NCS iConnect.NET 4.4 R3

Developer's Guide

5 Ang Mo Kio Street 62
NCS Hub
Singapore 569141
Phone: (65) 6556 - 8000
Fax: (65) 6556 - 7000
Web: www.ncs.com.sg
Email: iconnect-net@ncs.com.sg

April 25, 2013

Written by: Lu Xiao Feng, He Jiang Yan, Jiang Jin Nan, Lu Ya Ming, Wu Yong Qing, Chai Ruo Yu, He Jian Bo, Jin Zhi Hao

Copyright © NCS Pte Ltd. All rights reserved.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Preface xii
1. Scope xii
2. Objectives xiii
3. What's Covered? xiii
1. IoC 1
1.1. Introduction 1
1.2. Getting Started 1
1.3. `ServiceLocator` 3
1.4. `IServiceLocatableObject` 5
1.5. IoC Integrated with WCF 6
1.6. Interception 7
2. Application Context 9
2.1. Introduction 9
2.2. Getting Started 10
2.3. Get the Current ApplicationContext 14
2.4. Get/Set a Context Item 14
2.5. Get/Set the Value of Given Context Item 16
2.6. Configuring Application Context 16
2.7. WCF Extensions 18
2.7.1. Cross-Domain Context Propagation 18
2.7.2. Transaction ID Generation 19
3. Architecture 21
3.1. Tiers and Layers 21
3.1.1. 2-Tiers, 3-Tiers, and Distributed Architecture 21
3.1.2. 2-N-Layers 23
3.2. Data Access Layer 24
3.3. Business Component Layer 26
3.4. Service Interface Layer 27
3.5. Service Layer 27
3.5.1. Service in Non-Distributed Architecture 29
3.6. Service Proxy Layer 30
3.7. User Interface Layer 30
3.7.1. Service Proxy Invocation 31
4. Message Management 32
4.1. Introduction 32
4.2. Getting Started 32
4.3. Multiple Providers Design 35
4.4. Caching 36
4.5. InternalMessageEntry & MessageEntry 38
4.6. Message Format 39
4.7. Code Generation 40
4.7.1. Multiple Files Supporting 42
4.8. Localization 42
4.9. Message Maintenance 43

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

4.9.1. `MessageRoute` ... 43
4.9.2. Add New Message Entries ... 44
4.9.3. Retrieve Existing Message Entries ... 44
4.9.4. Modify/Delete Message Entries ... 45
4.9.5. Save Message Entries into an XML File ... 46
5. Code Table ... 47
5.1. Introduction ... 47
5.2. Getting Started ... 47
5.3. Code Effectiveness ... 51
5.4. Configuration ... 52
5.5. Code Maintenance ... 53
5.6. Code Binding ... 53
5.6.1. One-shot Binding ... 54
5.6.2. Two-Step Binding ... 55
5.7. Get ID/Value of Code ... 56
5.8. Code Authorization ... 56
5.9. Code Generation ... 58
6. Data Mapping ... 62
6.1. Introduction ... 62
6.2. Getting Started ... 62
6.3. NamingConverter ... 67
6.4. Specifying the Data Format ... 68
6.5. Specifying the Bound Property of Control ... 68
6.6. BindingMapping ... 69
7. Form Helper ... 70
7.1. Introduction ... 70
7.2. Data Binding ... 70
7.2.1. Data Binding Based on Given `BindingMappingCollection` ... 70
7.2.2. Data Binding Based on Automatically Built `BindingMappingCollection` ... 71
7.2.3. Control the Data Binding Behavior by Registering Event ... 72
7.3. Data Capture ... 73
7.4. Enabling/Disabling Controls ... 73
7.5. Reset Controls ... 74
7.6. Working with Menus ... 74
8. Page Helper ... 76
8.1. Introduction ... 76
8.2. Data Binding ... 76
8.2.1. Data Binding Based on Given `BindingMappingCollection` ... 76
8.2.2. Data Binding Based on Automatically Built BindingMappingCollection ... 77
8.2.3. Control the Data Binding Behavior by Registering Event ... 77
8.2.4. DataGrid/GridView based Data Binding ... 78
8.3. Data Capture ... 79

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

8.4. Enabling/Disabling Controls ... 80
8.5. Adding Hyperlinks for Validation Messages ... 80
9. Database Helper ... 81
9.1. Introduction ... 81
9.2. Loading Data into DataTable ... 81
9.3. Loading Data into DataSet ... 82
9.4. Updating Data into Database ... 84
9.5. ExecuteReader ... 88
9.6. ExecuteScalar ... 88
9.7. ExecuteNonQuery ... 89
9.8. ExecuteBulkCopy ... 89
9.9. Working with Events ... 90
9.10. Procedure Generation ... 90
10. Entity Framework ... 92
10.1. Introduction ... 92
10.2. Entity Data Model Transformation ... 92
10.3. ADO.NET POCO Self-Tracking Entity Generation ... 96
10.4. DbContext POCO Entity Generation ... 98
10.5. Repository Pattern ... 99
11. Transaction Management ... 103
11.1. Introduction ... 103
11.2. Getting Started ... 104
11.3. Transaction ... 107
11.4. `TransactionCallHandler` & `TransactionScopeCallHandler` ... 108
11.5. Transaction Programming in WCF ... 109
12. Exception Handling ... 110
12.1. Introduction ... 110
12.1.1. Exception Handler ... 110
12.1.2. Exception Policy ... 111
12.2. Getting Started ... 111
12.3. Error Reporting ... 113
12.3.1. FormExceptionHandler ... 113
12.3.2. MessageBoxExceptionHandler ... 114
12.3.3. AlertExceptionHandler ... 114
12.3.4. WebTextControlExceptionHandler ... 115
12.4. Wrapping & Replacement ... 115
12.4.1. WrapExceptionHandler ... 115
12.4.2. ReplaceExceptionHandler ... 116
12.5. Database Handlers ... 117
12.5.1. *SqlWrapExceptionHandler & OralceWrapExceptionHandler* ... 117
12.5.2. SqlReplaceExceptionHandler & OralceReplaceExceptionHandler ... 118
12.6. Filterable Handler ... 119
12.7. WCF Integration ... 122
13. Logging ... 126

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

13.1. Introduction ... 126
13.2. Getting Started ... 128
13.3. Writing Log Entries ... 131
13.3.1. LogEntry ... 131
13.3.2. The Logging Facade: Log ... 132
13.3.3. Messaging Integration ... 132
13.4. `FormattedDatabaseTraceListener` and `WcfTraceListener` ... 133
13.4.1. `FormattedDatabaseTraceListener` ... 133
13.4.2. `WcfTraceListener` ... 134
13.5. Viewing Log Entries ... 134
13.5.1. Configuration ... 135
13.5.2. `LogViewer` ... 136
13.6. Monitoring Instrumentation ... 137
13.6.1. Start New Activity ... 137
13.6.2. `MonitoringCallHandler` ... 139
13.6.3. Monitoring `TraceListener` ... 140
13.6.4. Monitoring Log Entry Retrieval ... 141
13.6.5. Configuration ... 142
14. Audit Trail ... 145
14.1. Introduction ... 145
14.2. Getting Started ... 145
14.3. Writing Audit Data into Log Storage ... 149
14.3.1. Single Table Audit V.S. Multiple Table Audit ... 150
14.3.2. Create an AuditLog ... 152
14.3.3. Writing Audit Data with Triggers ... 153
14.3.4. Writing Audit Data with SQLCDC ... 155
14.3.5. Writing Audit Data with DataSet ... 157
14.3.6. Writing Audit Data with DbCommand ... 157
14.3.7. Writing Audit Action ... 158
14.4. Querying Audit Data ... 158
14.5. Archiving Audit Data ... 159
15. Hierarchy ... 160
15.1. Introduction ... 160
15.2. Getting Started ... 160
15.3. `HierarchyNode` ... 162
15.4. StorageProvider and HierarchyManager ... 164
15.5. Hierarchy Maintenance ... 165
15.5.1. Add a New Hierarchy Node ... 166
15.5.2. Remove an Existing Node from Hierarchy Tree ... 167
15.5.3. Remove an Exiting Node to another Node ... 167
15.6. System Parameter ... 167
15.7. Organization Management ... 168
15.7.1. `OrganizationUser` and `OrganizationExtension` ... 169
15.7.2. `OrganizationManager` ... 169

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

16. States ... 171
16.1. Introduction ... 171
16.2. Getting Started ... 171
16.3. How to Define State Items ... 176
16.4. Expiration Policy ... 177
16.5. State Item Backing Storing ... 177
16.6. Code Generation ... 178
17. Session Management ... 181
17.1. Introduction ... 181
17.2. Getting Started ... 181
17.3. `SessionInfo` and `SessionInfoCollection` ... 186
17.4. SessionProvider ... 187
17.5. Starting Sessions ... 188
17.6. Concurrent Session Throttling ... 189
17.7. Session Expiration and Renewal ... 190
17.8. Terminating Sessions ... 192
17.9. Killing Sessions ... 192
17.10. Scavenging Sessions ... 192
17.11. Retrieve Session(s) ... 193
18. Security ... 195
18.1. Introduction ... 195
18.2. Membership ... 196
18.2.1. Introduction ... 196
18.2.2. Getting Started ... 197
18.2.3. Configuration ... 198
18.2.4. User Maintenance ... 200
18.2.5. User Validation ... 206
18.2.6. Change and Retrieve Password ... 206
18.3. Role Manager ... 207
18.3.1. Introduction ... 207
18.3.2. Getting Started ... 208
18.3.3. Configuration ... 209
18.3.4. Role Maintenance ... 210
18.3.5. Authorization ... 212
18.4. Authorization ... 215
18.4.1. Introduction ... 215
18.4.2. Getting Started ... 216
18.4.3. Configuration ... 217
18.4.4. Business Action Maintenance ... 218
18.4.5. Authorization via Business Action ... 220
18.4.6. `AuthorizeCallHandler` ... 222
18.5. Profile ... 224
18.5.1. Introduction ... 224
18.5.2. Getting Started ... 225

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.5.3. Configuration ... 227
18.5.4. Customize Profile ... 228
18.5.5. Profile setting and retrieval ... 230
18.5.6. Profile Maintenance ... 231
18.6. Site Map ... 232
18.6.1. Introduction ... 232
18.6.2. Getting Started ... 233
18.6.3. Site Map Data V.S. Menu Data ... 235
18.6.4. SiteMapDataSource based Binding ... 235
18.6.5. XmlDataSource based Binding? ... 236
18.6.6. SiteMap Structure Maintenance ... 237
18.6.7. Authorization ... 239
18.6.7.1. Role based Authorization ... 239
18.6.7.2. Business Action Authorization ... 240
18.6.8. Site Map Provider ... 240
18.7. SecureConnect ... 241
18.7.1. Introduction ... 241
18.7.2. Getting Started ... 242
18.7.3. Programming ... 244
18.7.4. Configuration ... 246
18.7.5. Best Practice ... 247
19. Batch Job ... 249
19.1. Introduction ... 249
19.2. Getting Started ... 249
19.3. Scheduling Service ... 252
19.4. ScheduleController ... 255
19.4.1. Database Schedule Controller ... 255
19.4.2. WCF Schedule Controller ... 256
19.5. Get Job Information ... 256
20. File Uploader ... 257
20.1. Introduction ... 257
20.2. Getting Started ... 257
20.3. How to manage the task ... 262
20.4. Inside File Uploader ... 263
20.5. Database File Uploader Uploader ... 264
20.6. WCF File Uploader Uploader ... 264
21. Data Transfer ... 265
21.1. Introduction ... 265
21.2. Getting Started ... 265
21.3. Programming ... 269
21.3.1. DataTransferManager ... 269
21.3.2. TransferComponent ... 269
21.3.3. DataTransferDbProvider ... 270
21.3.4. DataTransferWcfProvider ... 270

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

21.4. Design Overview ... 271
22. Controls ... 272
22.1. Introduction ... 272
22.2. Getting Started ... 273
22.3. RequiredFieldValidator ... 275
22.4. RegularExpressionValidator ... 275
22.5. RangeValidator ... 275
22.6. CompareValidator ... 276
22.7. CustomValidator ... 276
22.8. SGInfoValidator ... 277
23. Calendar ... 278
23.1. Introduction ... 278
23.2. Getting Started ... 278
23.3. StorageProvider ... 281
23.4. WorkdayCalendar ... 282
23.5. Extensibility ... 284
23.5.1. Step I: Create Custom CalendarTag ... 284
23.5.2. Step II: Create Custom CalendarBuilder ... 285
23.5.3. Step III: Create Custom Calendar ... 286
24. Caching ... 287
24.1. Introduction ... 287
24.2. Getting Started ... 287
24.2.1. SqlDependencyExpiration ... 288
24.2.2. AbsoluteTimeExpiration ... 289
25. Letter Management System (LMS) ... 291
25.1. Introduction ... 291
25.2. Terminologies ... 291
25.3. Getting Started ... 292
25.4. Document Processing Pipeline ... 301
25.5. Document Processing Job ... 302
25.5.1. Data Token ... 303
25.5.2. Document ... 304
25.5.3. Job and Job Item ... 304
25.5.4. Job Initializer & Job Item Initializer ... 305
25.5.5. Extensions ... 308
25.6. Processing Documents ... 309
25.6.1. Run the Job ... 309
25.6.2. Document Processing Context ... 310
25.6.3. Data Source ... 311
25.6.4. Document Handler ... 312
25.6.5. Convert File Format ... 313
25.6.6. Mail Merge ... 314
25.7. Delivering Documents ... 315
25.7.1. Document Store ... 315

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

25.7.2. Document Delivery Service ... 317
25.7.3. Document Deliverer ... 318
25.7.4. Document Delivery Listener ... 318
25.7.5. SmtpDocumentDeliveryListener ... 320
25.7.6. FileSaverDeliveryListener ... 322
25.7.7. Delivery Retry ... 323
25.7.8. Delivery Failure Handling ... 324
25.8. Document Processing Job Provider ... 324
25.8.1. Job Provider is an Extension ... 325
25.8.2. Job Maintenance ... 325
25.8.3. Job Item Maintenance ... 326
25.8.4. Template Maintenance ... 327
25.8.5. Data Source Maintenance ... 328
25.8.6. Mail Message Maintenance ... 329
25.8.7. DbDocumentProcessingJobProvider and WcfDocumentProcessingJobProvider ... 330
25.8.8. Run a Persisted Document Processing Job ... 331
25.9. Template Editor ... 332
25.9.1. Word Template Editor ... 332
25.9.1.1. Getting Started ... 332
25.9.1.2. Components ... 333
25.9.1.3. Create Template ... 334
25.9.2. Html mail merge and Html Template Editor ... 340
25.10. Mailing Service ... 344
25.10.1. How to Send Mail with HTML Template ... 345
26. Message Compression ... 348
26.1. Introduction ... 348
26.2. Programming ... 348
26.3. Extensions ... 349
27. ASP.NET MVC Extensions ... 350
27.1. Introduction ... 350
27.2. IoC ... 350
27.2.1. Getting Started ... 350
27.2.2. `ServiceLocatableControllerFactory` ... 352
27.3. Code Table ... 353
27.3.1. Getting Started ... 353
27.3.2. `ListControlAttribute` ... 357
27.3.3. `HtmlHelper` Extensions ... 358
27.3.3.1. DropDownList ... 359
27.3.3.2. ListBox ... 359
27.3.3.3. RadioButtonList ... 360
27.3.3.4. CheckBox & CheckBoxList ... 360
27.4. Exception Handling ... 361
27.4.1. Getting Started ... 361

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

27.4.2. `BaseController` ... 364
27.4.3. Set Exception Policy ... 365
27.4.4. How Exception Handling Works? ... 365
27.4.5. Get Error Information From ViewData ... 367
27.4.6. `ErrorMessageExceptionHandler` ... 367
27.5. Validation Enhancement ... 368
27.5.1. Getting Started ... 368
27.5.2. `BaseController` ... 371
27.5.3. ValidatorAttribute ... 371
27.5.3.1. Error Message Template ... 372
27.5.3.2. `RequiredValidatorAttribute` ... 373
27.5.3.3. `RegularExpressionValidatorAttribute` ... 373
27.5.3.4. `StringLengthValidatorAttribute` ... 373
27.5.3.5. `RangeValidatorAttribute` ... 374
27.5.3.6. `CompareValidatorAttribute` ... 374
27.5.3.7. `DataTypeValidatorAttribute` ... 375
27.5.3.8. `EnumDataTypeValidatorAttribute` ... 375
27.5.3.9. `NumericValidatorAttribute` ... 376
27.5.4. `DisplayTextAttribute` ... 376
27.5.5. ModelValidatorProvider ... 376
27.5.6. `ValidationRuleAttribute` ... 378
27.6. Localization ... 378
27.7. Authorization ... 379
27.7.1. `BusinessActionAttribute` & `AuthorizationRuleAttribute` ... 379
27.7.2. `ActionAuthorizeAttribute` ... 379
27.7.3. Role Assignment ... 381
27.8. FileUpload ... 382
28. Configuration Console ... 388
28.1. Introduction ... 388
28.2. The usage flow ... 389
28.2.1. iConnect audit trail settings ... 389
28.2.2. iConnect batch job settings ... 390
28.2.3. iConnect batch job service settings ... 390
28.2.4. iConnect calendar settings ... 391
28.2.5. iConnect code table settings ... 392
28.2.6. iConnect common settings ... 393
28.2.7. iConnect application context settings ... 393
28.2.8. iConnect message compression settings ... 394
28.2.9. iConnect data mapping settings ... 395
28.2.10. iConnect data access settings ... 396
28.2.11. iConnect hierarchy settings ... 397
28.2.12. iConnect LMS delivery settings ... 398
28.2.13. iConnect LMS manager settings ... 399
28.2.14. iConnect logging settings ... 400

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

28.2.15. iConnect instrumentation settings 401
28.2.16. iConnect message settings 402
28.2.17. iConnect security scheduler settings 403
28.2.18. iConnect session management settings 404
28.2.19. iConnect state settings 405
28.2.20. iConnect string resource settings 406

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Preface

In this information technology era, the business environment is extremely dynamic and competitive. Besides, the increased complexity of modern software construction and the ever-demanding project schedule have added immense pressure to the software development process. Time to market is one of the primary critical success factors, while cost is another. When translating these issues to the software development arena, it means a software development team must be able to roll out business software quickly, and within tight budgets, in order to stay competitive. The information technology is advancing at tremendously speed, with major innovations coming up every six months. However, it will take a novice developer at least six months to be productive in using certain new technology. Besides the various forms of formal and informal training, NCS has adopted additional ways to enhance our capabilities, such as knowledge management, discussion groups, centers of expertise, intellectual properties repositories, etc. In the area of intellectual properties (IP) development, NCS Chief Architect Office (CAO) has embarked on the vision of iConnect for developers. The iConnect umbrella achieves to provide enterprise services of portal, knowledge management, EAI and two major application framework - Java/J2EE and .NET technology. The functional architecture is depicted in the following diagram.

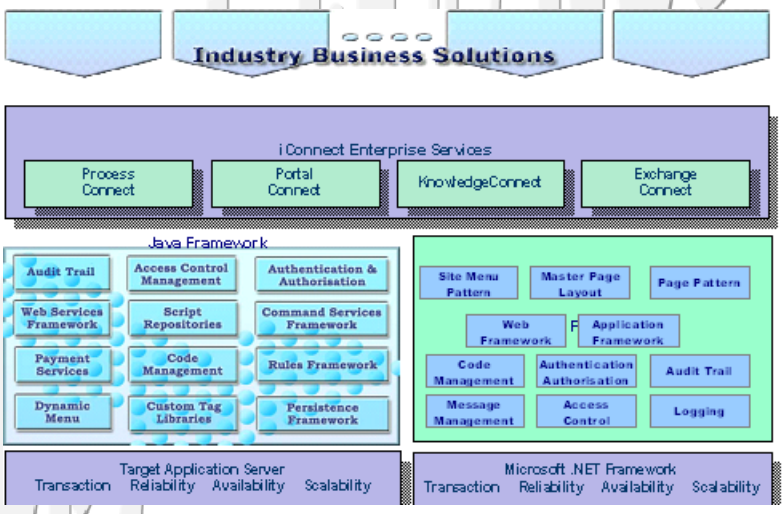


Figure 1. Industry Business Solutions

NCS CAO has already embarked on the development of Java Framework (iFrame) for Java/J2EE developers. Based on the concept of iFrame, a similar approach of IP development is available for the Microsoft .NET platform. Though the architectural elements of Java and .NET differ, the backend design and concepts is analogous.

1. Scope

The IP development for the iConnect.NET framework includes many essential building blocks (Microsoft called them Application Blocks). Because assembling building blocks into an application will also require time and the right knowledge, it is proposed that application

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

patterns be developed on top of these building blocks. Application patterns and application blocks together provide skeletal templates to the actual development of the application. They encapsulate best practices or guidelines of both architectural design and actual code implementation. They serve as concrete examples of how best practices or guidelines could be implemented in real practical sense, as opposed to pure theories only. The following section documents the functionalities provided by NCS iConnect.NET framework.

2. Objectives

The immediate objectives of the framework are as follows:

- To provide a set of application patterns and reusable components to speed up program design and development, and thus enhancing competitiveness.
- To harness development experiences and best practices in terms of concrete code implementations to enable re-use of knowledge, and thus maximizing the value of intellectual capital.
- To lay an architectural foundation upon which additional application patterns and application blocks could be built subsequently.

With iConnect.NET framework, development team will be able to start the initial program construction much earlier in the development life cycle. By embracing this premise, project costs can be cut and software quality can be improved as well.

3. What's Covered?

iConnect.NET is built on top of the .NET Framework and Enterprise Library. It contains the following major components, and provides additional features which are required by most of the projects:

- *Messaging*

  It provides the strongly typed message management solution and extends storage of messages.

- *Logging*

  The Logging Component enables easy and flexible logging. The logging output is customizable and can be configured to log to different destinations, such as a file, a database, event log or the MSMQ. A set of different filters can be specified in the configuration that specifies which messages that should be logged and to where.

- *Security*

  It provides the membership management, role management, authorization, menu and profile management. It utilizes the ASP.NET 2.0 security component and provider model.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- *Code Table*

  It provides simple and efficient read access to code tables, update access to system code table and several utility classes to bind codes to List Controls.

- *Audit Trail*

  Audit Trail can record down who has accessed a computer system and what operations he or she has performed during a given period of time.

- *Exception Handling*

  It provides a consistent strategy for processing exceptions that occur throughout the architectural layers of enterprise applications. iConnect.NET provides several useful additional exception handlers.

- *Transaction*

  It provides similar solution with the `System.Transactions` by using ADO.NET transaction. It provides a good balance between performance and easy of programming.

- *Application Context*

  It is used to store and retrieve content data in a transparent way. By using this module, the data can be passed from client to server without having to add additional parameters on your business functions, and these data can be passed between logical layers.

- *Batch Job*

  Batch job is used to schedule and run jobs at per-determined time in the scheduler server. The primary function is to manage jobs and run scheduling jobs. It will also support to view batch job status and history.

- *Calendar*

  Calendar is used for manages calendars. The primary function is managed calendar and filter.A calendar contains a set of days, and every day has a name attribute. Filters do some behavior on calendar's days, such as add days, remove days or change the day's name.

- *Controls*

  It provides a set of windows validators similar to ASP.NET validators.

- *Architecture*

  The proposed logical layers include Business Facade, Business Component, Data Access, Business Entity, Service Interface, Client Proxy and Client layers.

- *Helpers*

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

It includes helpers for Data Access, and helpers for processing data on UIs.

- *Session Management*

  It is used to track user's active status after login, and handle web and windows application(s).

- *LMS*

  LMS provides applications with a comprehensive solution of template based document generation and delivery.

- *ASP.NET MVC Extensions*

  ASP.NET MVC is an web application development framework of high extensibility, and a series of extensions are made to integrate with existing *iConnect.NET* components, such as IoC, exception handling, code table, messaging, etc. In addition, the ASP.NET MVC built-in validation is greatly enhanced.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 1. IoC

1.1. Introduction

The key to successfully building such applications is to achieve a decoupled or very loosely coupled design. Loosely coupled applications are more flexible and easier to maintain. They are also easier to test during development. You can mock up shims (lightweight mock implementations) of objects that have strong concrete dependencies; such as database connections, network connections, ERP connections, and rich user interface components. IoC (so-called dependency injection) is a prime technique for building loosely coupled applications. It provides ways to handle the dependencies between objects. For example, an object that processes customer information may depend on other objects that access the data store, validate the information, and check that the user is authorized to perform updates. Dependency injection techniques can ensure that the customer class correctly instantiates and populates all of these objects, especially where the dependencies may be abstract.

Unity is an open source IoC framework developed by Microsoft Patterns & Practices team. The latest *EntLib* is directly built on the basis of Unity. Leveraging a lightweight IoC container (Unity Container), Unity addresses the issues faced by developers engaged in component-based software engineering. Leveraging the interception mechanism provided by Unity, `<AOP>` is also achieved. From version iConnect.NET 4.0, IoC is introduced into iConnect.NET as a core component. iConnect.NET IoC, which extends Unity, serves the other functional modules. It can be also used as an independent component.

1.2. Getting Started

This section provides you with a sample to demonstrate which kinds of DI (Dependency Injection) and AOP features can be achieved. With your Visual Studio, you first create a new Console project with following assembly reference added:

- Microsoft.Practices.Unity;
- Microsoft.Practices.Unity.Configuration;
- Microsoft.Practices.Unity.Interception;
- NCS.IConnect.Core;
- NCS.IConnect.Caching.

Then create a new class file, in which the following four interfaces/classes are defined. The interface `ITimeService` has only one `GetCurrentTime` method, which is used to get the current time. `TimeService` implements this interface. However, it does not directly implement `ITimeService`'s `GetCurrentTime` method, which is implemented by invoke the `TimeProvider` property of `ITimeProvider` interface. The `TimeProvider` property is initialized in `TimeService`'s

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

constructor. In fact, the current time retrieval is ultimately implemented in `TimeProvider`, which implements `ITimeProvider` interface. Please note the `GetCurrentTime` method of `TimeProvider`, which is decorated with a special custom attribute: `CachingCallHandlerAttribute`. This attribute make it possible to cache the result based on the passed parameters when the target method is invoked, thus avoid duplicate method execution to improve performance.

```
using System;
using NCS.IConnect.Caching.CallHandlers;
using NCS.IConnect.Core.IoC;

namespace NCS.IConnect
{
    public interface ITimeService
    {
        DateTime GetCurrentTime();
    }
    public interface ITimeProvider
    {
        DateTime GetCurrentTime();
    }
    public class TimeService : ITimeService
    {
        public ITimeProvider TimeProvider { get; set; }
        public TimeService(ITimeProvider provider)
        {
            this.TimeProvider = provider;
        }
        public DateTime GetCurrentTime()
        {
            return this.TimeProvider.GetCurrentTime();
        }
    }
    public class TimeProvider : ITimeProvider
    {
        [CachingCallHandler]
        public DateTime GetCurrentTime()
        {
            return DateTime.Now;
        }
    }
}
```

Copy the following code snippet to override the Main method of the `Program` class.

```
static void Main(string[] args)
{
    IServiceLocator serviceLocator = ServiceLocatorFactory.GetServiceLocator();
    ITimeService timeService = serviceLocator.GetService<ITimeService>();
    for (int i = 0; i < 5; i++)
    {
        Console.WriteLine(timeService.GetCurrentTime());
        Thread.Sleep(1000);
    }
}
```

Finally, you need to add an App.config, and then copy the following configuration into this file. The "`NCS.IConnect.IoC`" in the code snippet indicates the assembly name.

```
<configuration>
  <configSections>
  <section name="unity"
           type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection,
                 Microsoft.Practices.Unity.Configuration"/>
  </configSections>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  <unity>
    <sectionExtension
type="Microsoft.Practices.Unity.InterceptionExtension.Configuration.InterceptionConfigurationExtension,
                            Microsoft.Practices.Unity.Interception.Configuration"/>
   <sectionExtension type="NCS.IConnect.Core.IoC.Configuration.AutoInterceptionConfigurationExtension,
                            NCS.IConnect.Core" />
    <container>
      <extension type="AutoInterception"/>
      <extension type="Interception"/>
     <register type="NCS.IConnect.ITimeService, NCS.IConnect.IoC" mapTo="NCS.IConnect.TimeService,
NCS.IConnect.IoC"/>
     <register type="NCS.IConnect.ITimeProvider, NCS.IConnect.IoC" mapTo="NCS.IConnect.TimeProvider,
NCS.IConnect.IoC"/>
      <autoInterception>
        <interceptor type="InterfaceInterceptor"/>
      </autoInterception>
    </container>
  </unity>
</configuration>
```

Run the console application, you will get the following output. Recall to the definition of Main, you will note that only one interface is specified, which is `ITimeService` interface. However, its implementer class (`TimeService`) can be resolved. `TimeService`'s construct accepts a parameter of `ITimeProvider` interface, which cannot be directly instantiated. However, its implementer class (`TimeProvider`) can be resolved. As with the output, we can see every result to invoke `GetCurrentTime` method is same. This proves the `CachingCallHandler` applied takes effects.

```
2/18/2011 4:07:35 PM
2/18/2011 4:07:35 PM
2/18/2011 4:07:35 PM
2/18/2011 4:07:35 PM
2/18/2011 4:07:35 PM
```

From the above demonstration, we can see the core of IoC is a special component: `ServiceLocator`, which is used to create a new instance specified type or get the existing instance of a given type, E.G. a singleton instance. We regard this process as service location. In the process of service location, the configured or dynamically registered dependency information can be resolved.

1.3. `ServiceLocator`

In iConnect.NET, the service location related functionalities are all implemented in a particular service locator. There is an `NCS.IConnect.Core.IoC.IServiceLocator` interface to define the service location functionalities. `IServiceLocator`'s definition is as below.

```
public interface IServiceLocator
{
    object GetService(IServiceLocationContext context);❶
    T GetService<T>(IServiceLocationContext context) where T : class;❷
    T GetService<T>(string key) where T : class;❸
    T GetService<T>() where T : class;❹

    IEnumerable<T> GetAllServices<T>();❺

    void Register<TInterface, TService>(string name);❻
    void Register<TInterface, TService>();❼
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    void Register(Type interfaceType, Type serviceType, string name);❽
}
```

❶ Create a new instance or get an existing instance based on given service location context information;

❷ Create a new instance of given type or get an existing instance of given based on given service location context information. The generic type stands for the type of returned instance;

❸ Create a new instance of given type or get an existing instance of given type based on given key of instance registration;

❹ Create a new instance of given type or get an existing instance of given based on the registered type;

❺ Create or get all instance registered as given type;

❻ Perform type registration with a given name;

❼ Perform type registration without a given name;

❽ Perform type registration with a given name.

The context information of service location is defined in the following `NCS.IConnect.Core.IoC.IServiceLocationContext` interface.

```
public interface IServiceLocationContext
{
    string Key { get; }
    Type Type{get;}
    IDictionary<string, object> ExtendedContexts { get; }
}
```

This interface only has one implementer class: `NCS.IConnect.Core.IoC.ServiceLocationContext`. You can directly instantiate a `ServiceLocationContext` instance by invoking its constructor. Two overloaded `Create` methods can also be used to create `ServiceLocationContext` object.

```
public class ServiceLocationContext : IServiceLocationContext
{
    public string Key { get; private set; }
    public Type Type { get; private set; }
    public IDictionary<string, object> ExtendedContexts { get; private set; }

    public ServiceLocationContext(string key, Type type)
    public static IServiceLocationContext Create<T>(string key) where T : class
    public static IServiceLocationContext Create<T>() where T : class
}
```

*i*Connect.NET IoC just provides only type of service locator: `NCS.IConnect.Core.IoC.UnityServiceLocator`. As its name suggests, `UnityServiceLocator` leverages Unity framework to conduct service location. Each `UnityServiceLocator` correspond to one configured `UnityContainer`. You can configure more one `UnityServiceLocator` specific to the same or different `UnityContainer`. The following code snippet is the demonstrative configuration.

```
<configuration>
  <configSections>
    <section name="unity"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection
                  Microsoft.Practices.Unity.Configuration"/>
    <section name="iConnect.serviceLocation"
            type="NCS.IConnect.Core.IoC.Configuration.ServiceLocationSettings,
                  NCS.IConnect.Core"/>
   </configSections>
  <unity xmlns="http://schemas.microsoft.com/practices/2010/unity">
    <container>
      ...
    </container>
    <container name="container1">
      ...
    </container>
    <container name="container2">
      ...
    </container>
  </unity>
  <iConnect.serviceLocation defaultServiceLocator="unityServiceLocator1"❶>
    <serviceLocators>
      <add name="unityServiceLocator1"
           containerName="container1"
           type="NCS.IConnect.Core.IoC.UnityServiceLocator,
                 NCS.IConnect.Core"/>❷
      <add name="unityServiceLocator2"
           containerName="container2"
           type="NCS.IConnect.Core.IoC.UnityServiceLocator,
                 NCS.IConnect.Core"/>❸
       </serviceLocators>
  </iConnect.serviceLocation>
</configuration>
```

❶ The name of default service locator. This is an optional configuration property, and the default value is empty;

❷ The first named configured `UnityServiceLocator` specific to the container1 `UnityContainer`;

❸ The second named configured `UnityServiceLocator` specific to the container2 `UnityContainer`;

In general, we use the NCS.IConnect.Core.IoC.ServiceLocatorFactory to create a given service locator. As illustrated in the following code snippet, there are two overloaded `GetServiceLocator` methods defined in `ServiceLocatorFactory` .

```
public static class ServiceLocatorFactory
{
    public static IServiceLocator GetServiceLocator(string locatorName)❶
    public static IServiceLocator GetServiceLocator()❷
}
```

❶ Create the service locator based on given configuration name;

❷ Create the default service locator. If *defaultServiceLocator* is not empty, the configured service locator will be created, otherwise, a `UnityServiceLocator` specific to the default `UnityContainer` is returned.

1.4. `IServiceLocatableObject`

In *iConnect.NET* Common, the following `NCS.IConnect.Common.IServiceLocatableObject` interface is defined. Only one read-only ServiceLocator property is defined. What is returned in this property is an instance of `IServiceLocator` type.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public interface IServiceLocatableObject
{
    IServiceLocator ServiceLocator { get; }
}
```

All of the layer specific base classes, including `DataAccessBase`, `BusinessComponentBase`, `ServiceBase`, `ServiceProxy` based and FormBase/PageBase, implement this interface. The default service locator is returned in this property. `ServiceLocatorFactory` give a easy-to-use manner to create the default/named service locator. In the classes which derive from the above base classes, you can directly use the `ServiceLocator` to get the default service locator.

1.5. IoC Integrated with WCF

IoC is ultimately used to create an instance of appropriate type, and the registered dependencies is automatically injected and resolved. According to iConnect.NET distributed architecture, the Service layer sits on the top-most of the service side layer stack. All of the service side layer specific components, `BusinessComonent`, `DataAccess`, and other related objects are directly or indirectly referred by a particular service instance. Applying IoC to the service side needs to use IoC container to create the service instance.

iConnect.NET IoC provides a WCF extension to integrate WCF with IoC. For detailed, such an extension uses the above mentioned service locator to create the service instance. This extension is implemented as a service behavior: `ServiceLocationBehavior`.

You have two ways to apply such a `ServiceLocationBehavior` to the target service: programmatic and administrative configuration. For the first one, you must need to apply the `NCS.IConnect.Core.IoC.WCF.ServiceLocationBehaviorAttribute` to the service class. There are two constructors for the attributes class, and they are defined in the following code snippet. If you use the default constructor without any parameter, the service instance will be created using the default service locator. The other constructor give you a choice to specify a given service locator.

```
public class ServiceLocationBehaviorAttribute: Attribute, IServiceBehavior, IEndpointBehavior
{
    //Others.
    public ServiceLocationBehaviorAttribute(string serviceLocatorName);
    public ServiceLocationBehaviorAttribute();
}
```

As a matter of fact, we do not adopt the above declaration based programming to apply service behavior in real project development. We usually adopt the administrative configuration manner. The following code snippet illustrates how to apply the `ServiceLocationBehavior` to the `OrderSerivice` in a configurable manner.

```
<configuration>
 <system.serviceModel>
  <behaviors>
   <serviceBehaviors>
    <behavior name="serviceLocation">❶
        <serviceLocation/>
    </behavior>
   </serviceBehaviors>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  </behaviors>
  <services>
   <service behaviorConfiguration="serviceLocation"❷
            name="NCS.IConnect.Demons.Service.OrdersService">
    <endpoint binding="wsHttpBinding"
              contract="NCS.IConnect.Demons.Service.Interface.IOrdersService"/>
   </service>
   </services>
    <extensions>
      <behaviorExtensions>
        <add name="serviceLocation"
             type="NCS.IConnect.Core.IoC.WCF.ServiceLocationBehaviorElement,
                   NCS.IConnect.Core"/>❸
      </behaviorExtensions>
    </extensions>
 </system.serviceModel>
</configuration>
```

❶ The named service behavior with `ServiceLocationBehavior`;

❷ Refer to the above configured named service behavior;

❸ The configuration element setting for the `ServiceLocationBehavior`.

1.6. Interception

Unity interception enables you to effectively capture calls to objects and add additional functionality to the target object. Interception is useful when you want to modify the behavior for individual objects but not the entire class, very much as you would do when using the Decorator pattern. It provides a flexible approach for adding new behaviors to an object at run time.

Policy injection by a combination of Unity and the patterns & practices Enterprise Library uses a set of call handlers and the equivalent call handler attributes in conjunction with the underlying Unity interception mechanism. Interception enables you to effectively capture calls to objects and provide additional functionality to the target object by using behaviors and call handlers in the pipeline to define and manage the results of the interception. In Enterprise Library, policy injection is just one implementation of a Unity interception behavior. The `PolicyInjectionBehavior` captures calls to objects you resolve through the container, and applies a policy that uses call handlers and matching rules inherited from Unity to define its policy injection behavior on a per-method basis.

If we directly use Unity for interception, we must explicitly apply the `PolicyInjectionBehavior` to the target interceptable type and specify the interceptor. The following code snippet demonstrates how to apply `PolicyInjectionBehavior` to the type of `IFoo` with the `InterfaceInterceptor` is used.

```
IUnityContainer container = new UnityContainer();
container.AddNewExtension<Interception>()
    .Configure<Interception>().SetInterceptorFor<IFoo>(new InterfaceInterceptor());
container.RegisterType<IFoo, Foo>(new InterceptionBehavior<PolicyInjectionBehavior>());
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

The `PolicyInjectionBehavior` and interceptor can also be applied to the target type in a configurable manner, The above programming take the same effect as the following configuration.

```
<configuration>
  <unity>
    <sectionExtension
type="Microsoft.Practices.Unity.InterceptionExtension.Configuration.InterceptionConfigurationExtension,
                            Microsoft.Practices.Unity.Interception.Configuration"/>
    <section name="iConnect.serviceLocation" />
    <container>
      <extension type="Interception" />
      <register type="IFoo" mapTo="Foo">
        <interceptor type="InterfaceInterceptor" />
        <policyInjection />
      </register>
    </container>
  </unity>
</configuration>
```

In order to avoid type specific interception registration, *iConnect.NET* provides us with an `UnityContainerExtension` of `AutoInterception`. As its name suggests, the `AutoInterception` explicitly apply the `PolicyInjectionBehavior` and default interceptor to the interceptable type. We can add the this extension and configure default interceptor in both programming and configuration manners. In the following code snippet, we add the `AutoInterception` extension to the given `UnityContainer` and specify the `InterfaceInterceptor` as the default interceptor. If no interceptor is explicitly specified, the `TransparentProxyInterceptor` will be used as the default one.

```
IUnityContainer container = new UnityContainer();
container.AddNewExtension<Interception>()
    .AddNewExtension<AutoInterception>()
    .Configure<AutoInterception>().RegisterDefaultInterceptor(new InterfaceInterceptor());
container.RegisterType<IFoo, Foo>(new InterceptionBehavior<PolicyInjectionBehavior>());
```

The above programming can be instead with the following configuration.

```
<configuration>
  <unity>
    <sectionExtension
type="Microsoft.Practices.Unity.InterceptionExtension.Configuration.InterceptionConfigurationExtension,
                            Microsoft.Practices.Unity.Interception.Configuration"/>
   <sectionExtension type="NCS.IConnect.Core.IoC.Configuration.AutoInterceptionConfigurationExtension,
                            NCS.IConnect.Core" />
    <container>
      <extension type="AutoInterception"/>
      <extension type="Interception"/>
      <register type="IFoo" mapTo="Foo"/>
      <autoInterception>
        <interceptor type="InterfaceInterceptor"/>
      </autoInterception>
    </container>
  </unity>
</configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 2. Application Context

2.1. Introduction

In a typical N-Layers based application, it is a common requirement that some contextual information should be shared across different layers. E.G. The current login user's profile information should be available in the whole application wide, regardless for user interface layer, data access layer or others. For a distributed application, the contextual information should also need to be propagated from client side to service side. iConnect.NET *Application Context* addresses the cross-layer contextual data sharing and cross-domain contextual data propagation.

iConnect.NET *Application Context*, can be considered as a global contextual information container, which is available in the whole application wide. Each context entry is stored in the container in the form of key-value pair. Leveraging the APIs provided by *Application Context*, you can get a particular context item based on the given key, and you can also add a new context item with a unique key into the container. `ApplicationContext` is one of the core components of iConnect.NET. Both iConnect.NET internal modules/components and application specific programs can directly use it as a global container to store contextual information.

Not only a simple contextual data container, the following are also achieved as well:

- *Cross-Thread Marshaling*

  Similar to *CallContext* classification, we classify all of application context items into two categories: *Logical context* and *Illogical context*. For asynchronous invocation, logical context will be carried along the execution path and illogical context will not. Illogical context is strictly kept in the scope of the current thread, and it will never be propagated or marshaled to the target thread for each asynchronous invocation. However, the cross-thread context propagation will automatically happen to logical context. We can initialize any context item in the form of logical context or illogical context. It is also supported to transfer a particular context item from logical/illogical to another.

- *Concurrency Control*

  Now that the logical context can be marshaled across different threads, multiple operations can be concurrently performed against one single context item. The concurrency control mechanism is applied to ensure synchronous writing to one single context item.

- *Read-only Context Item*

  In some situations, once the context item initialized, we want to keep it immutable and prevent anyone from changing it. In the latest version of iConnect.NET, not only can we directly create read-only application context item, but we can change the existing context item from writable to read-only.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- *3rd Party Client Compatibility*

  For most services exposed by iConnect.NET, the execution of the operations more or less dependents on some application context items, E.G. The current users ID. At application level, the service operation's implementation logic possibly uses more application context items, which are assumed to be automatically propagated from the consumer side. If we expect these context-bound services can be freely invoked by more consumers, including the consumers resides in non-.NET platform, we must make it possible to propagate application context in a platform independent manner. iConnect.NET enhances the existing WCF based application context propagation to achieve the pure 3rd party client compatibility.

- *Extended Properties*

  Each context item should not be limited to carry only one scale value. It will be more useful if it can also store more complementary information. An extended property collection, which is a dictionary with a key of string type, will be bound to each context item.

2.2. Getting Started

The following sample demonstrates you how to perform context propagation in a WCF based distributed application.

- *Step 1: Create a Distributed Solution*

  Create a new blank solution and add the following projects in it:

  - *Service.Interface:* This is a class library project to define the WCF service contract. The assembly reference to `System.ServiceMode` should be added;

  - *Service:* This is a class library project to define the WCF service, which implements the service contract defined in Service.Interface project. Assembly reference to `NCS.IConnect.Core` and project reference to Service.Interface project are added;

  - *Hosting*: This is a class library application project used to host the WCF service defined in the `Service` project. The project reference to `Service.Interface` and `Service` projects and assembly reference to `NCS.IConnect.Core` and `System.ServiceMode` are added. Then map project folder as a IIS virtual directory and give it a name: *helloworldservice*. Finally, do not forget to change build output directory from default \bin\debug to *\bin*;

  - *Client:* This is a Console application project as the consumer of the WCF service defined above. The assembly reference to `NCS.IConnect.Core` and `System.ServiceMode` and project reference to `Service.Interface` project are added.

    The following figure illustrates the structure of the whole demonstration solution:

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

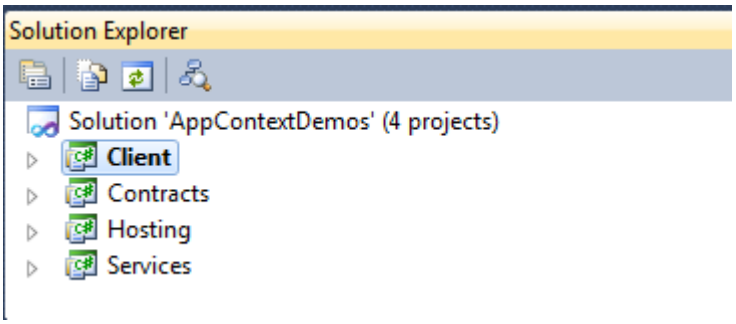


Figure 2.1. The structure of the Application Context Demonstration solution

- *Step 2: Create Service Contract*

  Define the WCF service contract in Service.Interface project: `IHelloWorldService`

```
using System.ServiceModel;
namespace AppContextDemos.Service.Interface
{
    [ServiceContract(Namespace = "http://www.ncs.com.sg/")]
    public interface IHelloWorldService
    {
        [OperationContract]
        string SayHello();
    }
}
```

- *Step 3: Create Service*

  Define the WCF service which implements `IHelloWorldService` contract in Contracts project: `HelloWorldService`. In the service operation `SayHello`, the current application context User with a "server" suffix is returned.

```
using System;
using AppContextDemos.Service.Interface;
using NCS.IConnect.Core.ApplicationContexts;

namespace AppContextDemos.Service
{
    public class HelloWorldService : IHelloWorldService
    {
       public string SayHello()
       {
           return ApplicationContext.Current.UserName + "server";
       }
    }
}
```

- *Step 4: Hosting Service*

  In the `Hosting` project, add a .svc file for the `HelloWorldService` to host. The *<%@ServicerHost%>* directive definition is as below.

```
<%@ ServiceHost language="C#" Service="AppContextDemos.Service.HelloWorldService" %>
```

  Add a web.config in the root directory, and add the service hosting configuration as below.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="iConnect.applicationContexts"
            type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                  NCS.IConnect.Core" />
  </configSections>
 <iConnect.applicationContexts defaultContextLocator="callContextLocator">❶
    <contextLocators>
      <add name="callContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.CallContextLocator,
                 NCS.IConnect.Core" />
    </contextLocators>
  </iConnect.applicationContexts>
  <system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior name="sampleServiceBehavior">❷
          <contextPropagation/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <services>
      <service behaviorConfiguration="sampleServiceBehavior"❸
               name="AppContextDemos.Service.HelloWorldService">
        <endpoint binding="wsHttpBinding"
                  name="HelloWorldService"
                  contract="AppContextDemos.Service.Interface.IHelloWorldService"/>
      </service>
    </services>
    <extensions>
      <behaviorExtensions>
        <add name="contextPropagation"
             type="NCS.IConnect.Core.ApplicationContexts.WCF.ContextPropagtionBehaviorElement,
                   NCS.IConnect.Core"/>❹
      </behaviorExtensions>
    </extensions>
  </system.serviceModel>
</configuration>
```

❶ The application context configuration section. The `CallContextLocator` is specified as the default context locator;

❷ Named service behavior with `ContextPropagtionBehavior`, which is used to implement cross-domain context propagation. This service behavior is applied to the hosted service;

❸ Refer to the above context-propagation enabled service behavior;

❹ The behavior extension configuration for the `ContextPropagtionBehavior`.

- *Step 5: Programming Client*

In the last step, we will complete the client side (service invocation) related program. Copy the following code into the Program.cs of Client project. In the program, the service proxy is first initialized via `WcfHelper`. Prior to invoke the `SayHello` service operation against the created service proxy, the application context item *UserName* is initialized as "Foo". Finally, the result of the service invocation is written to console.

```
using System;
using AppContextDemos.Service.Interface;
using NCS.IConnect.Core.ApplicationContexts;
using NCS.IConnect.Common;

namespace AppContextDemos.Client
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
{
    class Program
    {
        static void Main(string[] args)
        {
            var dummy = WcfHelper.GetObject<IHelloWorldService>("HelloWorldService");
            ApplicationContext.Current.UserName = "Foo";
            string serverName = dummy.SayHello();
            Console.WriteLine("Hello, {0}!", serverName);
            Console.Read();
        }
    }
}
```

Finally, add an App.config, and copy the following configuration into this file.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
   <configSections>
    <section name="iConnect.applicationContexts"
            type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                   NCS.IConnect.Core" />
  </configSections>
  <iConnect.applicationContexts defaultContextLocator="callContextLocator">
    <contextLocators>
      <add name="callContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.CallContextLocator,
                 NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>
  <system.serviceModel>
    <client>
      <endpoint name="HelloWorldService"
                address="http://localhost/helloworldservice/HelloWorld.svc"
                behaviorConfiguration="sampleEndpointBehavior"
                binding="wsHttpBinding"
                contract="AppContextDemos.Service.Interface.IHelloWorldService"/>
    </client>
    <behavior:
      <endpointBehaviors>
        <behavior name="sampleEndpointBehavior">
          <contextPropagation/>
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <extensions>
      <behaviorExtensions>
        <add name="contextPropagation"
             type="NCS.IConnect.Core.ApplicationContexts.WCF.ContextPropagtionBehaviorElement,
                   NCS.IConnect.Core"/>
      </behaviorExtensions>
    </extensions>
  </system.serviceModel>
</configuration>
```

Run the *Client* project, following running the *Hosting* project. You will get the following output at hosting console, which clearly approves the context (User Name: Foo) specified at client side is successfully propagated to server side.

```
Hello, Foo_server!
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2.3. Get the Current ApplicationContext

A `Current` static read only property is defined in the `ApplicationContext`. You can get the current `ApplicationContext` from it. The first access to the `Current` property will lead to current `ApplicationContext` initialization.

```
public class ApplicationContext
{
    //Others...
    public static ApplicationContext Current{get;}
}
```

2.4. Get/Set a Context Item

As we know, `ApplicationContext` can be regarded as a dictionary of context items, each of which is a instance of `NCS.IConnect.ApplicationContexts.ContextItem`. The following code snippet represents `ContextItem`'s definition.

```
[Serializable]
[DataContract(Name="ContextItem", Namespace=ApplicationContext.ContextHeaderNamespace)]
public class ContextItem
{
    public string Key { get; private set; }❶
    public object Value{get; set;}❷
    public bool IsLogical { get; private set; }❸
    public ExtendedPropertyCollection ExtendedProperties { get; private set; }❹
    public bool ReadOnly { get; set; }❺
    public bool IsLocal { get; private set; }❻
}
```

❶ The context item's key;

❷ The value of the context item;

❸ Indicates if the context item is logical or illogical. Logical context item is capable of cross-thread marshaling, whereas the illogical is kept in the scope of the current thread;

❹ A collection of extended property attached to the context item;

❺ Indicates if the context item is read-only or editable;

❻ Indicates if the context item is local or public. Just only public context item can be propagated from client side to service side.

Since `ApplicationContext` is a dictionary like container, you can get a particular context item based on its key. `ApplicationContext` gives you two ways to get a given context item. The first one is to use the `ApplicationContext`'s index as below.

```
public class ApplicationContext
{
    //Others...
    public ContextItem this[string key]{get;}
}
```

There is a core component in the `ApplicationContext` module: *ContextLocator* whose main responsibility is to get and set the current context item. All context locators implement the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`NCS.IConnect.ApplicationContexts.IContextLocator` interface. The following code snippet illustrates its definition. The `GetContextItem` method helps you to get a given context item.

```
public interface IContextLocator
{
    //Others...
    ContextItem GetContextItem(string key);
    void SetContextItem(ContextItem contextItem);
}
```

You can get the context locator via the `ContextLocator` property of `ApplicationContext`.

```
public class ApplicationContext
{
   //Others...
   public IContextLocator ContextLocator{get};
}
```

You can use the `ContextLocator` to set a given context item by invoking `SetContext` method. Such a `SetContext` method's logical is as below:

- If no context item of the same key exist, the new context item will be directly added into the current context item container;
- If a context item of the same key exists, and it is read only, an exception will be thrown;
- If a context item of the same key exists, but it is not read only, the exiting one will be overridden by the new one.

There is an more straightforward way to set a given context item by directly invoking the `SetContext<TValue>` method of `ApplicationContext` to add a new context item, or override the existing context item. The following code snippet illustrates the definition of the three `SetContext` methods of `ApplicationContext`.

```
public class ApplicationContext
{
   //Others...
   public void SetContext<TValue>(string key, TValue value);❶
   public void SetContext<TValue>(string key, TValue value, bool isLogical);❷
   public void SetContext<TValue>(string key, TValue value, bool isLogical, bool isLocal)❸
}
```

❶ If no context item of the same key exists, a new writable, logical and public (IsLocal = false) context with the specified key and value is added into current context item container. Otherwise the exiting context item's value is overridden;

❷ If no context item of the same key exists a new writable, and public (`IsLocal` = false) context with the specified key, value and isLogical indicator, is added into current context item container. Otherwise the exiting context item's `Value` and `IsLogical` are overridden;

❸ If no context item of the same key exists, a new writable context with the specified key, value, isLogical and isLocal indicator, is added into current context item container. Otherwise the exiting context item's Value, IsLogical and IsLocal are overridden;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

2.5. Get/Set the Value of Given Context Item

The most frequent usage is to get/set the value of a given context item. If you are in advanced know the context item's key, you can first get the corresponding context item. If the context item is here, you can get/set its entire properties, including its value.

As mentioned in the above section, you can invoke the `SetContext<TValue>` method of `ApplicationContext` to set the value (and other context item's properties) of given context item. `ApplicationContext` provides a `GetValue<TValue>` method to get the value of given context item. If no context item of the specified key exists, generic type specific default value is returned.

```
public class ApplicationContext
{
   //Others...
   public TValue GetValue<TValue>(string key)
}
```

For commonly used context items, such as UserId, UserName, TransactionId etc, there are corresponding properties defined in `ApplicationContext`. This makes it more straightforward to set/get their value.

```
public class ApplicationContext
{
   //Others...
   public string UserId{get;set;}
   public string UserName{get;set;}
   public string SessionId{get;set;}
   public string TransactionId{get;set;}
   public TimeZoneInfo TimeZone{get;set;}
   public CultureInfo Culture{get;set;}
   public CultureInfo UICulture{get;set;}
}
```

If you need to get/set other properties of the above built-in context items, you can use another set of properties, which return the corresponding `ContextItem` instance.

```
public class ApplicationContext
{
   //Others...
   public ContextItem UserIdContextItem
   public ContextItem UserNameContextItem{get;}
   public ContextItem SessionIdContextItem{get;}
   public ContextItem TransactionIdContextItem{get;}
   public ContextItem TimeZoneContextItem{get;}
   public ContextItem CultureContextItem{get;}
   public ContextItem UICultureContextItem{get;}
}
```

2.6. Configuring Application Context

In terms of context storage, there are two different mechanisms: *CallContext* and *HttpSessionState*. The two different mechanisms are implemented in two context locators: `CallContextLocator` and `HttpSessionStateContextLocator`. `HttpSessionStateContextLocator` is used for ASP.NET web application, and the others use

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`CallContextLocator`. The `ApplicationContext` based configuration is very simple, and we just need to specify appropriate context locator. The following code snippet is a demonstrative configuration.

```
<configuration>
 <configSections>
  <section name="iConnect.applicationContexts"
           type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                 NCS.IConnect.Core"/>
 </configSections>
<iConnect.applicationContexts defaultContextLocator="callContextLocator">
    <contextLocators>
      <add name="callContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.CallContextLocator
                 NCS.IConnect.Core"/>
     <add name="httpSessionStateContextLocator"
          type="NCS.IConnect.Core.ApplicationContexts.HttpSessionStateContextLocator,
                NCS.IConnect.Core"/>
    </contextLocators>
 </iConnect.applicationContexts>
```

As aforementioned, for all *CallContext* based context items, just only logical ones can be automatic marshaled to target thread in the asynchronous invocation. However, when we adopt `HttpSessionStateContextLocator`, the context items are all stored in the HTTP *SessionState* of current `HttpContext`. `HttpContext` is essentially *CallContext*, and it is illogical *CallContext*. So our `ApplicationContext` by default cannot support asynchronous invocation.

In order to resolve this problem, we use the `CallContextLocator` as the backup of `HttpSessionStateContextLocator` when current `HttpContext` is not available. In addition, we must achieve the consistence between `HttpSessionStateContextLocator` and its backup, and this is implemented a custom `HttpModule`: `NCS.IConnect.Core.ApplicationContexts.ContextHttpModule`. If your ASP.NET code involves asynchronous invocation which depends on a particular `ApplicationContext` item, you must register this `ContextHttpModule` in your web.config.

For IIS 7 (or later) with Integrated mode, custom `HttpModule` registration is configured in *<system.webServer>* node. The following is a demonstrative configuration.

```
<configuration>
  <system.webServer>
    <modules>
        <add name="contextHttpModule"
             type="NCS.IConnect.Core.ApplicationContexts.ContextHttpModule
                   NCS.IConnect.Core"/>
    </modules>
  </system.webServer>
</configuration>
```

If you use IIS 7 (or later) with classic mode or early edition to host your ASP.NET application, custom *HttpModule* registration is configured in *<system.web>/<httpModules>* node. The following is a demonstrative configuration.

```
<configuration>
  <system.web>
    <httpModules>
        <add name="contextHttpModule"
             type="NCS.IConnect.Core.ApplicationContexts.ContextHttpModule
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                    NCS.IConnect.Core"/>
    </httpModules>
  </system.web>
</configuration>
```

2.7. WCF Extensions

In distributed application with WCF, iConnect.NET provides an extension to achieve cross-domain context propagation. With this extension, the current application context at client side can be automatically sent to service side. In addition, another extension is provided to automatically generate transaction id for the transaction operations.

2.7.1. Cross-Domain Context Propagation

iConnect.NET achieve WCF based cross-domain context propagation via WCF extension, and a special endpoint behavior named `ContextPropagtionBehavior` is created, which can be both used at client and server side. At client side, a custom `ClientMessageInspector`, `ContextAttachMessageInspector` is used to wrap the current application context into a message header, and ultimately insert into the header collection of the outgoing message. At server side, a custom `CallContextInitializer`, `ContextReceiveCallContextInitializer`, is used to extract the context specific message header and set it as the current application context.

The WCF based context propagation configuration mainly focuses on how to apply the context propagation based endpoint behavior to the client and server runtime of WCF infrastructure. At service side, the `ContextPropagtionBehavior` is applied to the endpoint of the service. The following is a demonstrative configuration.

```
<system.serviceModel>
 <extensions>
  <behaviorExtensions>
   <add name="contextReceival"
        type="NCS.IConnect.Core.ApplicationContexts.WCF.ContextPropagtionBehaviorElement
              NCS.IConnect.Core"/>
  </behaviorExtensions>
 </extensions>
 <behaviors>
  <endpointBehaviors>
   <behavior name="contextReceival">
    <contextReceival />
   </behavior>
  </endpointBehaviors>
 </behaviors>
 <services>
  <service name="AppContextDemos.Service.HelloWorldService">
   <endpoint address="http://127.0.0.1:9999/helloworldservice"
             behaviorConfiguration="contextReceival"
             binding="ws2007HttpBinding"
             contract="AppContextDemos.Service.Interface.IHelloWorldService"/>
  </service>
 </services>
</system.serviceModel>
```

At client side, the same endpoint behavior is also needed to apply to the endpoint to invocate the service. As the following code snippet illustrates, it is very similar to the service side configuration.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<system.serviceModel>
 <behaviors>
 <extensions>
  <behaviorExtensions>
   <add name="contextReceival"
        type="NCS.IConnect.Core.ApplicationContexts.WCF.ContextPropagtionBehaviorElement
              NCS.IConnect.Core"/>
  </behaviorExtensions>
 </extensions>
  <endpointBehaviors>
   <behavior name="contextAttachment">
    <contextAttachment />
   </behavior>
  </endpointBehaviors>
 </behaviors>
 <client>
  <endpoint address="http://127.0.0.1:9999/helloworldservice"
            behaviorConfiguration = "contextAttachment"
            binding = "ws2007HttpBinding"
            contract = "AppContextDemos.Service.Interface.IHelloWorldService"
            name = "helloworldservice" />
 </client>
</system.serviceModel>
```

2.7.2. Transaction ID Generation

Transactional operation in WCF is the operation method to which the `OperationBehaviorAttribute` is applied and the TransactionScopeRequired is set true. Prior to invoking these operations, a new GUID will be generated annd attached to the current application context if it's empty. Such functionality for transaction initialization is implemented by `TransactionIdInitializeBehaviorAttribute`. As the following code snippet illustrates, `TransactionIdInitializeBehaviorAttribute` is defined as a custom attribute, and it is also a service behavior, endpoint behavior and operation behavior.

```
public class TransactionIdInitializeBehaviorAttribute :
    Attribute,
    IServiceBehavior,
    IEndpointBehavior,
    IOperationBehavior
{
    //...
}
```

`TransactionIdInitializeBehaviorAttribute` can directly be applied to service class as a service behavior, and can be applied to transactional operational operation method as a operation behavior. In addition, we can also leverage configuration to apply `TransactionIdInitializeBehaviorAttribute` to a particular endpoint as an endpoint behavior. The configuration structure is like this:

```
<system.serviceModel>
 <extensions>
  <behaviorExtensions>
   <add name="transactionIdInitialization"
        type="NCS.IConnect.Core.ApplicationContexts.WCF.TransactionIdInitializeBehaviorElement,
              NCS.IConnect.Core"/>
  </behaviorExtensions>
 </extensions>
 <behaviors>
  <endpointBehaviors>
   <behavior>
    <transactionIdInitialization />
   </behavior>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  </endpointBehaviors>
 </behaviors>
 </services>
</system.serviceModel>
```

Note

The functionality of transaction ID initialization is also implemented in the context propagation extension, so it is not necessary to reuse the `TransactionIdInitializeBehaviorAttribute` if `ContextPropagtionBehavior` is used.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 3. Architecture

3.1. Tiers and Layers

This guide uses the term *LAYER* to refer to component types and uses *TIER* to refer to physical distribution patterns.In other words, layers are logical and tiers are physical. This distinction is helpful, particularly when you want to talk about where you run your layers.

3.1.1. 2-Tiers, 3-Tiers, and Distributed Architecture

The following lists the 3 typical kinds of multiple tires based application architecture:

- *2-Tiers(C/S)*

  As the following figure illustrates, this is a typical traditional C/S based architecture. All of the code, including user interaction, business process and data access, is executed in the end user's client machine. The business data is persisted in a single database server.

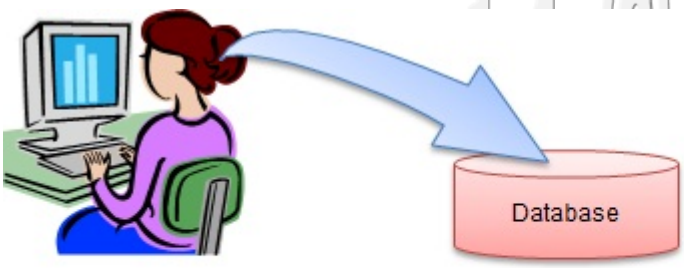


Figure 3.1. 2-Tier Architecture

- *3-Tiers(B/S)*

  This kind of architecture is adopted by simple web based application. As illustrated in the figure below, the end user uses the browser to consume the web application, which is hosted in the web server. The separate database server is also used to persist the business data.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

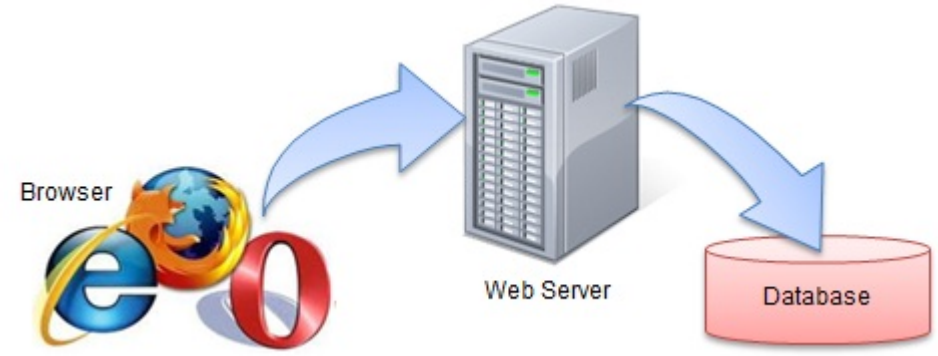


Figure 3.2. 3-Tier Architecture

- *Distributed Architecture*

  In the application of distributed architecture, business process, which is separated from user interaction, is exposed to the outer side in the form of a series of services. These reusable services are all hosted on one or more different servers, which is generally called application server. Distributed technologies, such as .NET Remoting, Web Service, WCF (Windows Communication Foundation) etc., are used as the communication means between client computer/web server and application server. The following figure illustrates the topology of Windows Form/Smart Client application based distributed application.

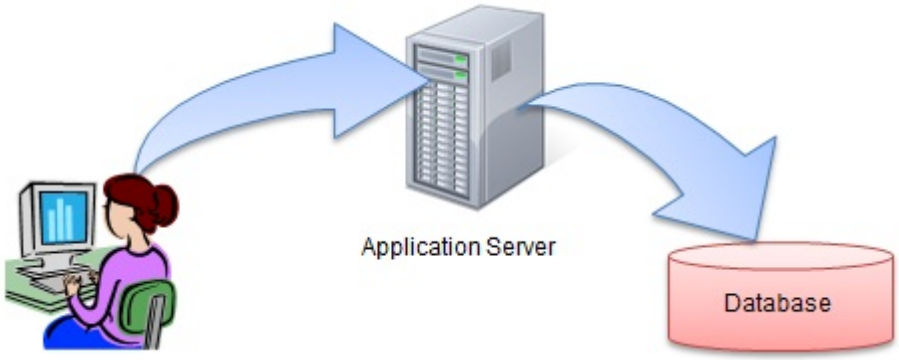


Figure 3.3. Distributed Architecture (Windows Form/Smart Client Application)

For web application with distributed architecture, the topology will be like the figure below.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

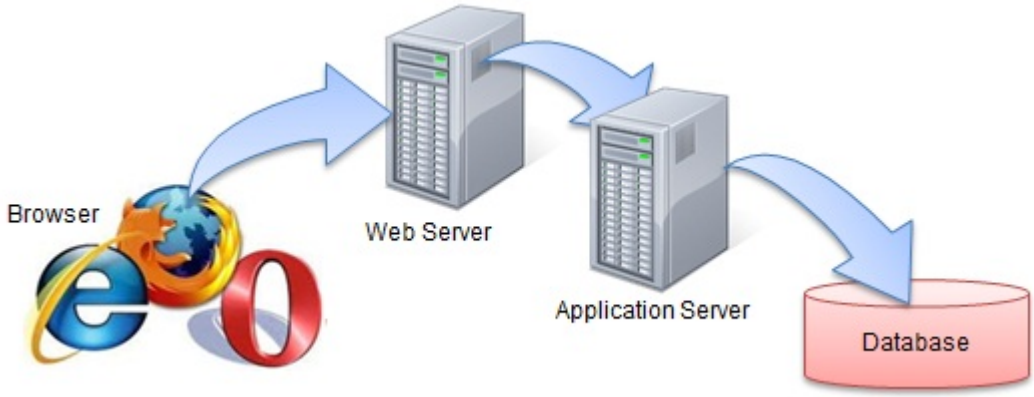


Figure 3.4. Distributed Architecture (Web Application)

The biggest benefit to adopt distributed application architecture is scalability. In real enterprise application deployment environment, the web server and application server tier are not one single physical server but a server cluster. Once bottlenecks happen to the two tiers, the application can easily scale out by adding more servers in the server cluster.

3.1.2. 2-N-Layers

Tier focus on which applications, components or services are deployed in which machines, and the physical communication between each other. *Layer* essentially stands for a kind of concern separation, and what it cares about is how to logically separate all of the logic into s different layers, and how to decouple the dependency between the layers each other. As a typical iConnect.NET based distributed application, its layer architecture is represented in the figure below.

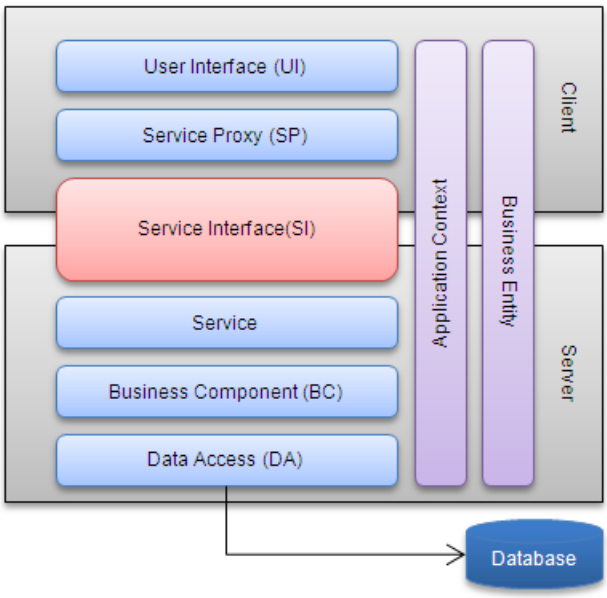


Figure 3.5. N-Layers for iConnect.NET Application

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- **Data Access:** Perform the pure data access operations. E.G. Execute a SQL statement to retrieve data from database; Execute a stored procedure to insert a new record, or update/delete an existing record;
- **Business Component:** Implement the main business logic. In the operation in business component layer, the data access components can be directly instantiated and invoked;
- **Service:** All of the related business logic are all encapsulated in variety of services. Currently, just only WCF service can be supported;
- **Service Interface:** Service interface can be regarded as the service contract, which abstracts the operations of service. Service implements the corresponding service interface, and the service proxy utilizes appropriate service interface to consume the service.
- **Service Proxy:** This is a proxy of the service on client behalf. In general, both service proxy and service implement the same service interface (service contract);
- **User Interface:** All of the user interface processing is implemented in the user interface layer, which can leverage service proxy to consume the corresponding service;
- **Business Entity:** All of the entity types are defined in business entity layer, which is available in the whole application. You can use strongly typed DataSet or custom entity types as business entities.
- **Application Context:** Application context is a container to store application-wide contextual data which need to be shared across layers, and even across client and server domain.

Each layer has its own responsibility, E.G. All of database access functionalities must be implemented in data access layer; transaction enlistment and audit logging are service layer specific responsibility; just only the user interface layer and service layer need to handle exception.

3.2. Data Access Layer

All of data access classes must be derived (directly or indirectly) from `NCS.IConnect.Architecture.Server.DataAccessBase`, which is defined in the `NCS.IConnect.Architecture.Server` assembly. The following code snippet represents the definition of `DataAccessBase`.

```
public class DataAccessBase : ApplicationContextAwareObject, IServiceLocatableObject
{
    //Others...
    public IServiceLocator ServiceLocator{get;}
    protected virtual DbHelper Helper{get;}
}
```

`DataAccessBase` implements the `NCS.IConnect.CommonAccessBase.IServiceLocatableObject` interface, which defines only one `ServiceLocator` property whose type is `NCS.IConnect.Core.IoC.IServiceLocator`. Service locator leverages the configured IoC

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

container (Unity Container) to create a new or locate an existing instance in a *dependency injection (DI)* manner. If the type of the instance to create is defined in the downstream layer, or another module, we strongly recommend you to create it in such a IoC pattern.

Another property `Helper` of the `DataAccessBase` return a `NCS.IConnect.Helpers.Data.DbHelper` instance, such a helper class can be used to perform most of database access functionalities.`DataAccessBase` directly inherits the `NCS.IConnect.Common.ApplicationContextAwareObject`, which defines an `ApplicationContext` property returning the current `ApplicationContext`.

```
public class ApplicationContextAwareObject : MarshalByRefObject, IApplicationContextAwareObject
{
    public ApplicationContext ApplicationContext { get; }
}
```

Generally speaking, data access layer just only contains the pure data access based logic, and the business process based code should not be defined in this layer.

If adopting ADO.NET Entity Framework as the means of database access, it is recommended to use repository pattern. iConnect.NET Entity Framework extensions provides two abstract base classes `RepositoryBase<TDbContext>` and `Repository<TDbContext, TEntity>`, in which the basic CRUD based method are defined. We can refer to the chapter of Entity Framework to see detail about how to program.

In order to isolate the data access based from the business logical, and make it possible to independently perform unit test against business logic, we generally make the very concrete repository implement an interface. Leveraging iConnect.NET service location based IoC, the business component can activate the repository instance via interface. Interface based repository activation depends on the mapping between repository type and its specific interface. We can manually register such a type mapping, and other necessary dependency injection behaviors in configuration file.

Instead of such a configurable type registration, we can also register all of repository type in a batch by calling `ServiceLocator`'s extension method "`RegisterRepositories`". Once this method is called, the `BuidManager` will be used to load all of the assemblies and get all repository types, which inherit `RepositoryBase<TDbContext>`, and finally these repository types will be registered to the interfaces they implement.

```
public static class ServiceLocatorExtensions
{
    public static void RegisterRepositories(this IServiceLocator serviceLocator, params Type[]
excludedInterfaces❶);
}
```

❶ The interface type list which will be ignored when performing type registration.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

3.3. Business Component Layer

Two abstract classes defined in `NCS.IConnect.Architecture.Server` assembly are used as the base class of all business components. The first one is `NCS.IConnect.Architecture.Server.BusinessComponentBase`, and its definition is represented in the code snippet below. It has only one `ServiceLocator` property used for IoC based instance creation or location.

```
public class BusinessComponentBase : ApplicationContextAwareObject, IServiceLocatableObject
{
    public IServiceLocator ServiceLocator{get;}
}
```

It cross-module invocation is involved at service side in your application, it is strongly recommended to perform this in business component layer. What is called should be the service other than its downstream business component or data access object. In order to improve loosely decouple between two different modules, the caller should consume the functionalities exposed by the callee via interface. Leveraging the type registration between interface and service type, we can use the `ServiceLocator` to perform interface based cross module invocation. About the usage of `ServiceLocator`, please refer to the chapter of *IoC and AOP*.

Besides `BusinessComponentBase`, there is another generic base class defined in the same namespace: `BusinessComponentBase<TDataAccess>`. The following code snippet illustrates its definition. The read only property `DataAccess` returns the data access instance of the generic type.

```
public class BusinessComponentBase<TDataAccess> : BusinessComponentBase where TDataAccess :
DataAccessBase
{
    public BusinessComponentBase(TDataAccess daObject);
    public TDataAccess DataAccess { get; private set; }
}
```

Generally speaking, all of business process related code should be defined in this layer. However, the transaction enlistment and exception is not in the scope of business component layer's responsibility set.

As mentioned above, the repository pattern is recommended to adopted if using ADO.NET Entity Framework, and business component classes leverage service locator to interact with repository via interface. In addition, the repository which wraps the `DbContext` object, must be disposed in time. The following code snippet represents the programming we should follow.

```
public class ProductBC: BusinessComponent
{
    public Product[] GetAllProducts()
    {
         using(IProductRepository reposity = this.ServiceLocator.GetService<IProductRepository>())
         {
             return repository.GetAllProducts();
         }
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

3.4. Service Interface Layer

In distributed applications, all of business process related functionalities are exposed to the outer side as one WCF service. The service contract is separately defined in this layer. We recommend you to specify the namespace of the service contract in the `ServiceContractAttribute`. The following code snippet is a simple demonstrative service interface.

```
[ServiceContract(Namespace = Constants.Namespace)]
public interface IOrdersService
{
    [OperationContract]
    void OrderProducts(Order order);
}
```

3.5. Service Layer

Like `BusinessComponentBase` and `BusinessComponentBase<TDataAccess>`, there are two similar service layer specific base classes: `ServiceBase` and `ServiceBase<TDataAccess>`. Their definition is represented in the code snippet below.

```
public class ServiceBase : ApplicationContextAwareObject, IServiceLocatableObject
{
    public IServiceLocator ServiceLocator{get;}
}
```

```
public class ServiceBase<TBusinessComponent> : ServiceBase where TBusinessComponent :
BusinessComponentBase
{
    public ServiceBase(TBusinessComponent bcObject){get;}
    public TBusinessComponent BusinessComponent { get; set; }
}
```

If you adopt IIS as the service host, you should add the service specific .svc file. To facilitate the framework to handle exception thrown from service operation execution, you should override the `Factory` directive attribute to adopt our own `ServiceHostFactory`: `NCS.IConnect.ExceptionHandling.WCF.ExceptionHandlingServiceHostFactory`. For such a `ServiceHostFactory`, you can refer to the WCF integration section of the Exception Handling chapter. The following is a demonstrative .svc.

```
<%@ ServiceHost Service="NCS.IConnect.QuickStarts.Orders.Service.OrdersService"
                Factory="NCS.IConnect.ExceptionHandling.WCF.ExceptionHandlingServiceHostFactory"%>
```

If you do not use such a custom `ServiceHostFactory`, you *MUST* manually add the type `NCS.IConnect.Common.ServiceExceptionDetail` as all operations' fault contract. The following is a demonstrative code snippet. To avoid adding such a `FaultContractAttribute` to all operation methods, we recommend you to use the custom `ServiceHostFactory` in the .svc file.

```
[ServiceContract(Namespace = Constants.Namespace)]
public interface IOrdersService
{
    [OperationContract]
    [FaultContract(typeof(ServiceExceptionDetail))]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    void OrderProducts(Order order);
}
```

For the operation with data modification involved, they should be enlisted into a single transaction. We recommend using WCF transaction programming, and you just need to apply the `OperationBehaviorAttribute` to the target operation method and set the `TransactionScopeRequired` property as true. The following is a demonstrative code snippet. About how to program transaction in WCF, please refer to the MSDN online documentation.

```
public class OrdersService : ServiceBase<OrdersBC>, IOrdersService
{
    //Other Members
    [OperationBehavior(TransactionScopeRequired = true)]
    public void OrderProducts(Order order)
    {
        //...
    }
}
```

For the service operations which need to be audited (for audit logging, please refer to Audit Trail section), you should add the custom attribute: `AuditTrailCallHandler`. The following is a demonstrative code snippet, and the specified parameter ("Order Products") is the function name to audit.

```
public class OrdersService : ServiceBase<OrdersBC>, IOrdersService
{
    //Other Members
    [OperationBehavior(TransactionScopeRequired = true)]
    [AuditTrailCallHandler("Order Products")]
    public void OrderProducts(Order order)
    {
        //Call Business Component Code
    }
}
```

As with service hosting configuration, there are three custom WCF service behaviors which *CANNOT* be missed.

- *ContextPropagtionBehavior*: It is used to receive the application context from the incoming message;
- *ServiceLocationBehavior*: It applies the Unity based IoC to service instance provision, and *AOP (PIAB)* is also depend on it;
- *ExceptionHandlingBehavior*: It implements the integration between WCF and *EnterLib Exception Handling Application Block*.

The above three service behaviors are all included in the configuration snippet as below.

```
<configuration>
 <configSections>
    <section name="exceptionHandling"
             type="Microsoft.Practices.EnterpriseLibrary.ExceptionHandling
                   .Configuration.ExceptionHandlingSettings,
                   Microsoft.Practices.EnterpriseLibrary.ExceptionHandling"/>
  </configSections>
 <system.serviceModel>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  <behaviors>
   <serviceBehaviors>❶
    <behavior name="serviceBehavior">
     <serviceLocation />
     <exceptionHandling exceptionPolicy="svcPolicy" />
     <contextPropagation />
    </behavior>
   </serviceBehaviors>
  </behaviors>
  <services>
   <service behaviorConfiguration="serviceBehavior"❷
            name="NCS.IConnect.QuickStarts.Orders.Service.OrdersService">
     <endpoint binding="wsHttpBinding"
               contract="NCS.IConnect.QuickStarts.Orders.Service.Interface.IOrdersService" />
   </service>
  </services>
    <extensions>
      <behaviorExtensions>❸
        <add name="contextPropagation"
             type="NCS.IConnect.Core.ApplicationContexts.WCF.ContextPropagtionBehaviorElement,
                   NCS.IConnect.Core"/>
        <add name="serviceLocation"
             type="NCS.IConnect.Core.IoC.WCF.ServiceLocationBehaviorElement
                   NCS.IConnect.Core"/>
        <add name="exceptionHandling"
             type="NCS.IConnect.ExceptionHandling.WCF.ExceptionHandlingBehaviorElement,
                   NCS.IConnect.ExceptionHandling.WCF"/>
     </behaviorExtensions>
    </extensions>
  </system.serviceModel>
  <exceptionHandling>
    <exceptionPolicies>
      <add name="svcPolicy">
        ...
      </add>
    </exceptionPolicies>
  </exceptionHandling>
</configuration>
```

❶ Named service behavior configuration, including the above mentioned three behaviors;

❷ Apply the above configured service behavior to the target service;

❸ The configuration of the behavior extension for the three service behaviors.

3.5.1. Service in Non-Distributed Architecture

Even when developing the applications without distributed architecture, it is recommended to keep the service layer. The service, which still implements an interface, is not a WCF service any more, and it is just a locally deployed service. iConnect.NET service location based IoC is also used to perform service consuming via interface. The mapping between service type and its specific interface and other necessary dependency injection behaviors can be manually configured.

If we do not want to perform type registration for each service type in such a configurable manner, we can call the `ServiceLocator`'s extension method `RegisterServices`. Once this method is invoked, all of the service types which inherit the `ServiceBase` will be automatically registered and mapped to the implemented interface. The following code snippet illustrates the definition of `RegisterServies` method.

```
public static class ServiceLocatorExtensions
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public static void RegisterServices(this IServiceLocator serviceLocator, params Type[]
excludedInterfaces);
}
```

3.6. Service Proxy Layer

In general, we create a single service proxy class for each service in the User Interface based project, other than creating a single project as the service proxy layer. Similar to other layers, this layer has its own base class: `NCS.IConnect.Architecture.Client.WcfProxyBase<Tchannel>`, whose generic type is the service contract. The following code snippet represents this class's definition.

```
public class WcfProxyBase<TChannel>: ApplicationContextAwareObject
{
    public IServiceLocator ServiceLocator{get;}
    protected TChannel Invoker { get; private set; }
    public WcfProxyBase(string endpointConfigurationName);
}
```

We *MUST* use the wrapped TChannel instance, which is returned by `Invoker` property, to perform service invocation, since it will help you close or abort the open channel and handle communication exception at client side. The following is the code snippet demonstrating how to invocate a service operation.

```
public class OrdersProxy : WcfProxyBase<IOrdersService>, IOrdersService
{
    public OrdersProxy():base(Constants.Endpoint)
    {}

    public void OrderProducts(Order order)
    {
        this.Invoker.OrderProducts(order);
    }
}
```

3.7. User Interface Layer

In the user interface base layers, there are two specific base classes: `NCS.IConnect.Architecture.Client.FromBase` and `NCS.IConnect.Architecture.Client.PageBase`. `FromBase` which is a sub-class of `System.Windows.Forms.Form`, is used in the windows forms application. It definition is illustrated in the code snippet below.

```
public class FormBase :
    System.Windows.Forms.Form,
    IApplicationContextAwareObject,
    IServiceLocatableObject
{
    public IServiceLocator ServiceLocator{get;}
    public ApplicationContext ApplicationContext{get;}
}
```

`PageBase` which is a sub-class of `System.Web.UI.Page` is used in the Web application. Its definition is illustrated in the code snippet below.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public class PageBase :
    System.Web.UI.Page,
     IApplicationContextAwareObject,
     IServiceLocatableObject
{
     public IServiceLocator ServiceLocator{get;}
     public ApplicationContext ApplicationContext{get;}
}
```

3.7.1. Service Proxy Invocation

There are two different program to perform service proxy invocation. The one is to directly instantiate the service proxy object and invoke its method, and the other is to use `ServiceLocator` to activate the service proxy instance based on service interface. As with the second one, the mapping between service proxy type and interface, with other related dependency injection behaviors will be configured in advanced. This kind of type registration can be achieved in a configurable manner, and we also provide the following extension method "`RegisterServiceProxies`" for `ServiceLocator` to automatically register all of the service proxy types.

```
public static class ServiceLocatorExtensions
{
    public static void RegisterServiceProxies(this IServiceLocator serviceLocator, params Type[]
excludedInterfaces);
}
```

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 4. Message Management

4.1. Introduction

Literal messages are widely used in all kinds of applications, regardless of the user interactive applications or a background service without UI. The appropriate messages should be shown to response the end user interaction, or the instrumentation information should be logged for the purpose of troubleshooting. iConnect.NET Messaging is a lightweight framework to manage message entries, which can be persisted in different storages, E.G XML file and database.

iConnect.NET Messaging provides a full set of application programming interface for message entry retrieval and maintenance. The code generation is shipped to make you can programming Messaging in a highly efficient manner. Avoiding frequently get access to the message source, XML file or database, multiple degrees caching mode is supported. In order to make iConnect.NET fit the multi-language based application, message localization is available.

4.2. Getting Started

The following demonstration application will guide you how to program iConnect.NET Messaging .Including how to define the T4 template for code generation, and how to format the message using the generated code.

Step I. Define Message Entries

Create a new Console application using Visual Studio, and add the assembly reference to `NCS.IConnect.Messaing.dll`, which is located in the /bin folder of the iConnect.NET installation directory. In the demonstration application, we use an XML file to store the message entries. We add a new blank XML file into the root directory of the project, and name it "Messages.xml". Then copy the following XML snippet into this file. In the XML file, we create three message entries.

```
<iConnect.messaging.messages>
  <message category="Validation"
           id="MissingManadatoryField"
           severity="error"
           value="The {0} is mandatory, pease specify!" />
  <message category="Validation"
           id="BetweenTwoValue"
           severity="error"
           value="The {0} must be between {1} and {2}!" />
  <message category="Validation"
           id="InDomain3"
           severity="error"
           value="The {0} must be {1}, {2} or {3}!"/>
</iConnect.messaging.messages>
```

In order to make the XML file automatically copied to the output directory when the project is built, we need set its "Copy to Output Directory" property "Copy always". For this, you just need to right click the XML file in the solution explorer, and choose the "*Properties*" item in

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

the context menu. You can get the "Copy to Output Directory" property in the Properties dialog box.

Step II. Create T4 Template

The code generation of iConnect.NET Messaging is also on T4, so we need to add a new T4 template file in the project. We right click the project in the solution explorer, and choose "Add/New item" context menu item. From the installed project templates in the "Add new item" dialog box, select the "General" template. Finally choose the "Text Template" from the general item list. Name the newly added text template file "Messages.tt", the copy the following code snippet to override the existing content in this file. Please not forget to revise the assembly references if iConnect.NET is not installed in the default directory (C:\Program Files).

```
<#@ template  language="C#" hostSpecific="true" debug="true" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Messaging.dll" #>
<#@ assembly
     name="C:\Program Files\NCS iConnect.NET\{version
no}\bin\Microsoft.Practices.EnterpriseLibrary.Caching.dll" #>
<#@ import namespace="NCS.IConnect.Messaging.CodeGeneration" #>
<#@ import namespace="NCS.IConnect.Core.CodeGeneration" #>
<#@ import namespace="System.CodeDom.Compiler" #>
<#@ import namespace="System.IO" #>
<#@ include file="T4Toolbox.tt" #>
<#@ output extension="cs" #>
<#
    string templateFileDirectory = Path.GetDirectoryName(this.Host.TemplateFile);
    var codeGenerator = new MessageGenerator(this.Host.ResolvePath("App.config"),
                                             Languages.CSharp,
                                             new CodeGeneratorOptions(),
                                             "NCS.IConnect.Messaging",
                                             templateFileDirectory);
   codeGenerator.Run();
#>
```

A configuration file (App.config), in which the iConnect.NET Messaging configuration are all defined, is referenced in the T4 template file. You need to add a new App.config in the project, and use the following XML snippet as the application configuration.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="iConnect.messaging"
             type="NCS.IConnect.Messaging.Configuration.MessageManagerSettings,
                   NCS.IConnect.Messaging"/>
  </configSections>
  <iConnect.messaging>
    <messageProviders>
      <add name="xmlProvider"
           type="NCS.IConnect.Messaging.XmlMessageProvider,
                 NCS.IConnect.Messaging"
           path="Messages.xml"
           cachingMode="None"/>
    </messageProviders>
  </iConnect.messaging>
</configuration>
```

Now, any attempt to save the template file (Messages.tt) will trigger the T4 engine to generate the code in the dependent file Messages.cs. Code generation can also be triggered by "Run Custom Tool". The template file specific "Custom Tool" will run when you right click the file

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

in the solution explorer and choose the "Run Custom Tool" item in the context menu. The following code snippet represents the C# code, which matches the message entries defined in the XML file.

```
//------------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.225
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------------

namespace NCS.IConnect.Messaging {
    using NCS.IConnect.Messaging;


    /// <summary>Define the generated message class.</summary>
    public sealed partial class Messages {

        private static NCS.IConnect.Messaging.MessageManagerBase manager;

        private static ValidationCategory validation;

        /// <summary>Define the message manager.</summary>
        public static NCS.IConnect.Messaging.MessageManagerBase Manager {
            get {
                if ((Messages.manager == null)) {
                    System.Threading.Monitor.Enter(typeof(Messages));
                    try {
                        if ((Messages.manager == null)) {
                           Messages.manager = NCS.IConnect.Messaging.MessageManagerFactory.Create();
                        }
                    }
                    finally {
                        System.Threading.Monitor.Exit(typeof(Messages));
                    }
                }
                return Messages.manager;
            }
            set {
                Messages.manager = value;
            }
        }

        /// <summary>Contains messages in the Validation category.</summary>
        public static ValidationCategory Validation {
            get {
                if ((Messages.validation == null)) {
                    Messages.validation = new ValidationCategory();
                }
                return Messages.validation;
            }
        }

        public sealed class ValidationCategory {

            /// <summary>Returns a message similar to: The {0} must be between {1} and {2}!</summary>
            public MessageEntry3 BetweenTwoValue {
                get {
                    return new MessageEntry3(Messages.Manager.GetMessage("Validation",
"BetweenTwoValue"));
                }
            }

            /// <summary>Returns a message similar to: The {0} must be {1}, {2} or {3}!</summary>
            public MessageEntry InDomain3 {
                get {
                    return new MessageEntry(Messages.Manager.GetMessage("Validation", "InDomain3"));
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                }
            }

            /// <summary>Returns a message similar to: The {0} is mandatory, pease specify!</summary>
            public MessageEntry1 MissingManadatoryField {
                get {
                    return new MessageEntry1(Messages.Manager.GetMessage("Validation",
"MissingManadatoryField"));
                }
            }
        }
    }
}
```

Step III: Retrieve and Format Message

Now that the generated code reflects the whole message entries, we can directly leverage the code to retrieve them. In addition, the message value defined in the XML file contains one or more placeholder, and we need to format the retrieved message entry by specifying the values corresponding to the placeholders, and finally generated a complete formatted literal message. The message entry retrieval and format programming is as below:

```
Console.WriteLine( Messages.Validation.MissingManadatoryField.Format("User Name"));
Console.WriteLine(Messages.Validation.BetweenTwoValue.Format("Age",18,15));
Console.WriteLine(Messages.Validation.InDomain3.Format("Language","Chinese","English", "Japanese"));
```

Output:

```
The User Name is mandatory, pease specify!
The Age must be between 18 and 15!
The Language must be Chinese, English or Japanese!
```

4.3. Multiple Providers Design

The provider design is applied in iConnect.NET Messaging, and each message storage or source has its specific `MessageProvider`. Unlike the general iConnect.NET component, in which always only one provider (the default provider or given named provider) is used, iConnect.NET Messaging can support multiple `MessageProvider`. As the following XML snippet illustrates, we can configure more than one `MessageProvider`, and the default `MessageProvider` cannot be specified.

```
<iConnect.messaging >
  <messageProviders>
    <add name="xmlProvider1"
         type="NCS.IConnect.Messaging.XmlMessageProvider,
               NCS.IConnect.Messaging"
          path="Message1.xml" cachingMode="None"/>
    <add name="xmlProvider2"
         type="NCS.IConnect.Messaging.XmlMessageProvider,
               NCS.IConnect.Messaging"
         path="Message2.xml" cachingMode="None"/>
    <add name="dbMessageProvider"
         type="NCS.IConnect.Messaging.DbMessageProvider,
               NCS.IConnect.Messaging.Database"
          database="iConnectDb"
          applicationName="iConnectDemo"
          cachingMode="None"/>
  </messageProviders>
</iConnect.messaging>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

When you attempt to get a particular message entry by specify the message ID, all of the configured `MessageProvider` will be invoked in the configured order until the given message entry is located. For example, in the above configuration, there are totally three configured `MessageProvider`. Among the three `MessageProvider`, the first two are `XmlMessageProvider` (xmlProvider1 and xmlProvider2) and the third is `DbMessageProvider` (*`dbMessageProvider`*). If we get a message entry by given the ID, xmlProvider1 will be be first invoked to try to get it from its specific XML file. If no message entry cannot found, the subsequent `MessageProvider` will be one by one executed until the target message entry is found.

iConnect.NET provides two built-in `MessageProvider`: `XmlMessateProvider` and `DbMessageProvider`. As their names suggest, they are separately specific to two kinds of message entry storage: XML file and database. In the distributed application, the `XmlMessageProvider` and `DbMessageProvider` are used at service side, and `WcfMessageProvider` is used at client. The following XML snippet illustrates how to configure `WcfMessageProvider`, whose configuration includes an endpoint attribute to refer to the endpoint to the iConnect.NET Messaging service.

```
<configuration>
  <system.serviceModel>
    <client>
      <endpoint name="Messages"
       behaviorConfiguration="contextPropagation"
       binding="wsHttpBinding"
       contract="NCS.IConnect.Messaging.IWcfMessage"
       address="http://localhost/ICSampleService/Iconnect/Messages.svc"/>
    </client>
    <behaviors>
      <endpointBehaviors>
        <behavior name="contextPropagation">
          <contextPropagation/>
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <extensions>
      <behaviorExtensions>
        <add name="contextPropagation"
             type="NCS.IConnect.Core.ApplicationContexts.WCF.ContextPropagtionBehaviorElement,
                   NCS.IConnect.Core" />
      </behaviorExtensions>
    </extensions>
  </system.serviceModel>
  <iConnect.messaging>
    <messageProviders>
      <add name="wcfProvider"
           type="NCS.IConnect.Messaging.WcfMessageProvider,
                 NCS.IConnect.Messaging"
       endpoint="Messages"
       cachingMode="None"/>
    </messageProviders>
  </iConnect.messaging>
</configuration>
```

4.4. Caching

In order to void frequently getting access to physic storage (XML file and database) and invoking WCF service, caching is applied to the three `MessageProvider`. iConnect.NET Messaging based caching directly leverages EntLib Caching Application Block. The three

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

aforementioned `MessageProvider` specific caching configuration is represented in the following XML snippet.

```
<iConnect.messaging >
  <messageProviders>
    <add name="xmlProvider1"
         type="NCS.IConnect.Messaging.XmlMessageProvider,
               NCS.IConnect.Messaging"
         path="Message1.xml"
         cachingMode="All"❶
         cacheManager="DefaultCacheManager"❷/>
    <add name="dbMessageProvider"
        type="NCS.IConnect.Messaging.DbMessageProvider,
              NCS.IConnect.Messaging.Database"
        database="iConnectDb"
        applicationName="iConnectDemo"
        cachingMode="All"
        cacheManager="DefaultCacheManager"
        expirationTime="00:30:00"❸/>
    <add name="wcfMessageProvider"
        endpoint ="Messages"
        type="NCS.IConnect.Messaging.WcfMessageProvider,
              NCS.IConnect.Messaging"
        cachingMode="All"
        cacheManager="DefaultCacheManager"
        expirationTime="00:30:00"/>
  </messageProviders>
</iConnect.messaging>
```

❶ The caching mode to control the cache granularity. The available options include `None`, `Single`, `Category` and `All`. If `None` caching mode is chosen, caching will be disabled;

❷ The configuration name of EnterLib `CacheManager`, which is used to cache message entries;

❸ The duration before the cached message entries expire.

Note

Both `DbMessageProvider` and `WcfMessageProvider` use absolute timeout based cache item expiration policy. However, the cache item expiration policy used in `XmlMessageProvier` is based on file dependency. The message entries cached by `XmlMessageProvider` will automatically expire once the source XML file is updated. So the duration configuration attribute only exists `DbMessageProvider` and `WcfMessageProvider` specific configuration element.

Generally speaking, we will force clear the cached message entries once we change the message source. In this situation, you can invoke the following two ClearCache method of MessageManager to clear the cached message entries.

```
public MessageManager: MessageManagerBase
{
    //...
}
public abstract class MessageManagerBase
{
    //...
    public virtual void ClearCache();❶
    public virtual void ClearCache(string providerName);❷
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ Clear the message entries cached by all `MessageProvider`;

❷ Clear the message entries cached by the given `MessageProvider`. What is specified is the configuration name of the `MessageProvider`.

In general, we do not manually instantiate the `MessageManager` object. iConnect.NET Messaging provides you with a static `MessageManagerFactory` to create the `MessageManager`. You just need to call the static Create method without any input parameters.

```
public class MessageManagerFactory
{
   //Others
   public static MessageManager Create();
}
```

Message caching by default happens to the first message retrieval attempt. In some situation, we need to pre-cache the message entries before the first end user's request reaches, and this can greatly improve the first visit specific responsiveness. Message entry pre-caching can be achieved by calling the `PreLoadMessages` methods of the `MessageManager`.

```
public MessageManager: MessageManagerBase
{
    //...
}
public abstract class MessageManagerBase
{
    //...
    public void PreLoadMessages();❶
    public void PreLoadMessages(CultureInfo culture);❷
    public void PreLoadMessages(string category);❸
    public void PreLoadMessages(string category, CultureInfo culture);❹
}
```

❶ Pre-load the current culture specific message entries;

❷ Pre-load the given culture specific message entries;

❸ Pre-load the current culture specific message entries of the given category;

❹ Pre-load the given culture specific message entries of the given category;

4.5. InternalMessageEntry & MessageEntry

In iConnect.NET Messaging application programming interface, the message entry is represented by an `InternalMessageEntry` object. The definition of `InternalMessageEntry` is represented in the following code snippet.

```
public class InternalMessageEntry : IComparable, ICloneable
{
    //Other Memebers
    public string Category { get;}❶
    public string CultureName { get; }  ❷
    public string Id { get; } ❸
    public MessageSeverity Severity { get;  }❹
    public MessageStatus Status { get;  }❺
    public string Value { get; }❻
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ The category of the message entry;

❷ The code of the culture of message entry is specific to;

❸ The ID uniquely identify a given message entry;

❹ The severity of the message entry. The available severity includes: Debug, Warning, Error, Critical, Confirmation and Information.

❺ The status of the message entry for update. The available status includes: Unchanged, Deleted, Modified, Added and Detached.

❻ A literal text representing the value of the message entry. This literal value can include placeholder.

iConnect.NET Messaging provides an extra type named `MessageEntry`. `MessageEntry` directly inherits from `InternalMessageEntry`, and defines two `Format` methods. The definition of the `MessageEntry` is represented in the following code snippet.

```
public sealed class MessageEntry : InternalMessageEntry
{
    //Others
    public string Format(params object[] args);
    public string Format(MessageFormat messageFormat, params object[] args);
}
```

When you get a `MessageEntry`, you can call the `Format` methods and specifying the values to replace the placeholder to get a formated literal message.

4.6. Message Format

From the definition of `InternalMessageEntry`, we can see there are a series of properties for a given message entry. By default, the formatted message only includes the Value of the message entry. As with `MessageEntry` mentioned in the above section, there are two overloaded `Format` methods defined. The second Format method accepts a `messageFormat` parameter, and its type is `MessageFormat`. By specifying such a parameter in the `Format` method, you can control the format of output literal text, which can include ID, Severity, Category and other message entry related property. `MessageFormat` is an enumeration, and the following code snippet represents its definition.

```
public enum MessageFormat
{
    Default,❶
    Value,❷
    IdValue,❸
    CategoryIdValue,❹
    CategoryIdSeverityValue❺
}
```

❶ If no message format is explicitly specified in the `Format` method, this option is automatically used. By default, this option specific formatting effect is the same as the Value;

❷ The output formatted message only includes the Value of the message entry;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❸ The output formatted message only includes the ID and Value of the message entry. The output format is by default like "*{ID}: {Value}*";

❹ The output formatted message only includes the Category, ID and Value of the message entry. The output format is by default like "*{Category} {ID}: {Value}*";

❺ The output formatted message only includes the Category, ID, Severity and Value of the message entry. The output format is by default like "*{Severity} {Category} {ID}: {Value}*";

By default, the above five `MessageFormat` options have their specific format. You can also leverage iConnect.NET Messaging configuration to override the default format settings.

There is a formatSpecifications configuration attribute in the *<iConnect.messaging>* section. You can use this configuration attribute to define the above five options specific message format templates. Each template can contain one or more predefined placeholders, each of which represents one a particular property of the message entry. The following table lists all of the placeholders.

Table 4.1. Message formatting placeholder

Placeholder	Description
{0}, {id}	The Id of the message.
{1}, {category}	The Category of the message.
{2}, {severity}	The Severity of the message.
{3}, {value}	The formatted Value of the message.

The formatSpecifications in the following configuration represents the default format templates of the five `MessageFormat` options. In other words, the format effects are the same if this configuration attribute is emitted.

```
<iConnect.messaging
   formatSpecifications="{value}
         ;{value}
         ;{id}: {value}
         ;{category} {id}: {value}
         ;{severity} {category} {id}: {value}">
  ...
</iConnect.messaging>
```

4.7. Code Generation

Although a series of methods are defined in iConnect.NET Messaging facade class `MessageManager`, it is strongly recommended to leverage code generation to retrieve all of the message entries and dynamically generate managed code. Programing the generated code in a strongly typed manner cannot not only improve productivity but avoid causing errors as well.

iConnect.NET Messaging code generation is based on T4. We just need to create a T4 template in your project, in which the code should be generated. All of the code generation related functionalities are all implemented in `MessageGenerator`. As the following code snippet

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

illustrates, you must need to instantiate the `MessageGenerator` object and invoke its Run method in the T4 template.

```
<#@ template  language="C#" hostSpecific="true" debug="true" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Messaging.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version
no}\bin\NCS.IConnect.Messaging.Database.dll" #>
<#@ import namespace="NCS.IConnect.Messaging.CodeGeneration" #>
<#@ import namespace="NCS.IConnect.Core.CodeGeneration" #>
<#@ import namespace="System.CodeDom.Compiler" #>
<#@ import namespace="System.IO" #>
<#@ include file="T4Toolbox.tt" #>
<#@ output extension="cs" #>
<#
    string templateFileDirectory = Path.GetDirectoryName(this.Host.TemplateFile);
    var codeGenerator = new MessageGenerator(this.Host.ResolvePath("App.config")❶,
           Languages.CSharp❷,
           new CodeGeneratorOptions()❸,
           "NCS.IConnect.Messaging"❹,
           templateFileDirectory)❺;
   codeGenerator.Run();
#>
```

❶ The path of the configuration file, in which the iConnect.NET Messaging related configuration is defined;

❷ The programming language of the generated code. VisualBasic and CSharp can be supported.

❸ The `CodeGeneratorOptions` object to control the style of the generated code;

❹ The namespace of the generated code;

❺ The directory in which the T4 template file is located. This is used to help revise the file path of the `XmlMessageProvider`.

From the code generated in the Getting Started sample, we can see the outermost of the generated code is a static Messages class. The static Messages class has a series of nested classes, each of which corresponds to a message category. The concrete message entry based code exists as a property of its category specific class.

For the message entry with one or more placeholders, we need to specify the parameters to replace these placeholders when formatting the message entry by calling the Format method. In order to make the programmer always provide matched parameters (the parameter count always equals the count of placeholder), iConnect.NET will generate different types of the MessageEntry based on the count of the placeholders. In iConnect.NET Messaging application programming interface, MessageEntry has four sub-classes: `MessageEntry0`, `MessageEntry1`, `MessageEntry2` and `MessageEntry3`. The suffix number stands for the placeholder count define in the value of message entry. As the following code snippet illustrates, they have its own Format methods, whose parameters match the placeholder count.

```
public sealed class MessageEntry0 : InternalMessageEntry
{
    public string Format();
    public string Format(MessageFormat messageFormat);
}
public sealed class MessageEntry1 : InternalMessageEntry
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public string Format(object arg0);
    public string Format(MessageFormat messageFormat, object arg0);
}
public sealed class MessageEntry2 : InternalMessageEntry
{
    public string Format(object arg0, object arg1);
    public string Format(MessageFormat messageFormat, object arg0, object arg1);
}
public sealed class MessageEntry3 : InternalMessageEntry
{
    public string Format(object arg0, object arg1, object arg2);
    public string Format(MessageFormat messageFormat, object arg0, object arg1, object arg2);
}
```

In most situations, projects prefer to use different XML files to maintain module specific message entries. *iConnect.NET* enables you to generate a partial class, and .cs/.vb files specific to these XML files contribute to this class. When conducting such a multiple files based code generation, there must be and only one master `MessageGenerator` among the ones instantiated in the T4 templates. `MessageGenerator` is by default master, the non-master `MessageGenerator` is instantiated in calling the following costrutor overload, and the last parameter of boolean type indicates to create master `MessageGenerator` or not.

```
...
<#
    string templateFileDirectory = Path.GetDirectoryName(this.Host.TemplateFile);
    var codeGenerator = new MessageGenerator(this.Host.ResolvePath("App.config"),
           Languages.CSharp,
           new CodeGeneratorOptions(),
           "NCS.IConnect.Messaging",
           templateFileDirectory,
           false);
   codeGenerator.Run();
#>
```

4.7.1. Multiple Files Supporting

In most situations, projects prefer to use different XML files to maintain module specific message entries

4.8. Localization

Message localization is the basic demand for the applications with multiple languages support. iConnect.NET messaging provides built-in location support, regardless which message persistent storage is adopted. If we use XML file to store message entries, iConnect.NET differentiates the culture based on the suffix of XML file name. In details, for the same set of message entries for different cultures, are separately stored in culture specific XML files. We name the culture specific message XML file by add a culture code suffix. For example, Xxx.en-US.xml and Xxx.zh-CN.xml stored the same set of message entries, one for Simplified Chinese and the other for American English. The XML file whose name includes no culture code suffix is used to stored culture neutral messages. The following figure illustrates the message files structure in a multi-languages application.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Figure 4.1. Culture specific XML Message Files in Multiple-Language Application

If the `DbMessageProvider`, which is based on database message entry storage, we use the CULTURE_NAME column of the T_IC_MESSAGE table to determine the culture to which each message entry is specific. The culture code is stored in the CULTURE_NAME column, the message entry is considered as culture neutral if no culture code is specified.

For the single message entry retrieval attempt, iConnect.NET always gets the appropriate message entry based on the current thread's CurrentUICulture (other than CurrentCulture) . If no message entry which matches the culture, the culture neutral based one will be retrieved instead.

4.9. Message Maintenance

The message maintenance related functionalities, including add new message entries or update/delete the existing message entries, can be achieved by invoking one single Update method of `MessageManager`.

```
public MessageManager: MessageManagerBase
{
    //...
}
public abstract class MessageManagerBase
{
    //...
    public abstract void Update(InternalMessageEntryCollection messages);
}
```

4.9.1. `MessageRoute`

Because iConnect.NET Messaging adopts multiple provider mode, and each `MessageProvider` is specific to a particular message source (XML file or database). When we submit a message entry to add, iConnect.NET must determine which `MessageProvider` should be used. A special flag is attached to a message entry to indicating which `MessageProvider` it is belong to. Such a flag is called *`MessageRoute`*, which is necessary for message maintenance, not only for adding new message entries but also for updating/deleting the existing message entries. In this situation, the configuration name of `MessageProvider` is used as the `MessageRoute`.

In distributed scenario, multiple `MessageProvider` can be used in both client side (E.G. One or more `WcfMessageProvider`) and service side (E.G. One or more `DbMessageProvider` or `XmlMessageProvider`). At client side, the `MessageRoute` help to determine which `MessageProvider` is used to deliver the message entry to service side. After the message entry is successfully received at service side, the `MessageRoute` also help to choose a appropriate `MessageProvider` to save the message entry to the target XML file or database. In this scenario,

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

the format of the `MessageRoute` is like "`<<ClientMessageProviderName>>@<<ServiceMessageProviderName>>`".

The `InternalMessageEntry` type has a read-only `MessageRoute` property, which is initialized in the constructor.

```
public class InternalMessageEntry
{
    //Others
    public string MessageRoute { get; }
    public InternalMessageEntry(string category, string id, string severity, string value, string
remark, string messageRoute, string cultureName);
}
```

4.9.2. Add New Message Entries

For adding new message entries, you need to create an `InternalMessageEntryCollection`, and add the newly created `InternalMessageEntry` in it, and then invoke the Update method to submit the newly added message entries into the appropriate message source. The target message source for each message entry is determined by the MessageRoute. The following code snippet is a demonstration.

```
var category        = "Validation";
var id              = "CreaterThan";
var severity        = MessageSeverity.Error;
var value           = "The {0} must be greater than {1}.";
var remarks         = string.Empty;
var messageRoute    = "xmlProvider";
var culture         = string.Empty;

InternalMessageEntry messageEntry = new InternalMessageEntry
(category, id, severity, value, remarks, messageRoute, culture);
InternalMessageEntryCollection messages = MessageManagerFactory.Create().GetAllCultureNeutralMessages();
messages.Add(messageEntry);
MessageManagerFactory.Create().Update(messages);
```

4.9.3. Retrieve Existing Message Entries

Regardless of adding new message entries or modifying/deleting existing message entries, what is invoked is always the `Update` method of `MessageManager`. As mentioned in the above section, there is a Status property defined in `InternalMessageEntry` type, and it helps `MessageManager` to determine which operation (Add, Modify or Delete) should be performed. When you add a newly created `InternalMessageEntry` into a given `InternalMessageEntryCollection`, the status is transfer Detached to Added, so the message entry can be successfully added into the target message source. If you need to modify or delete the existing message entries, you have to retrieve them from the message source. The facade `MessageManager` defines the following methods to perform message entry retrieval.

```
public MessageManager: MessageManagerBase
{
    //...
}
public abstract class MessageManagerBase
{
    public abstract InternalMessageEntryCollection GetAllCultureNeutralMessages();❶
    public InternalMessageEntryCollection GetAllMessages();❷
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public abstract InternalMessageEntryCollection GetAllMessages(CultureInfo culture);❸

    public virtual InternalMessageEntry GetMessage(string category, string id);❹
    public abstract InternalMessageEntry GetMessage(string category, string id, CultureInfo culture);❺

    public virtual bool TryGetMessage(string category, string id, out InternalMessageEntry message);❻
    public abstract bool TryGetMessage(string category, string id, CultureInfo culture, out
InternalMessageEntry message);❼
}
```

❶ Get all culture neutral message entries;

❷ Get all current culture specific message entries;

❸ Get all given culture specific message entries;

❹ Get the current culture specific message entry based on the given category and ID;

❹ Get the specified culture specific message entry based on the given category and ID;

❻ Try to get the current culture specific message entry based on the given category and ID;

❼ Try to get the specified culture specific message entry based on the given category and ID

Note

Regardless of `GetMessage` and `TryGetMessage`, the neutral message entry will retrieved if the culture specific one cannot be found. When the `GetMessage` is called, a `MessagingException` will be thrown if both message entries, culture specific and culture neutral, cannot be found.

4.9.4. Modify/Delete Message Entries

From the definition of `InternalMessageEntry`, we can see that the primary properties, such as ID, Category, Value, Severity, etc., are all read-only. We cannot modify a given InternalMessageEntry by directly setting these properties. `InternalMessageEntry` exposes a series of `Edit` methods make you can easily change the value of a particular property. The following code snippet illustrates the definition of these `Edit` methods. Once any one `Edit` method is invoked, the status of the message entry will be automatically transferred to Modified. When the edited message entries are delivered to the Update method of `MessageManager`, iConnect.NET Messaging is aware of the target operation is modification.

```
public class InternalMessageEntry : IComparable, ICloneable
{
    //Others...
    public void Edit(string value);
    public void Edit(MessageSeverity severity, string value);
    public void Edit(string value, string remark);
    public void Edit(MessageSeverity severity, string value, string remark);
    public void Edit(string category, string id, MessageSeverity severity, string value, string
remark, string cultureName);
}
```

As with message entry deleting, you need to invoke the `Delete` method, which will change the status to Deleted. Finally you deliver the deleted message entries to `MessageManager`'s Update method for physical delete.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

4.9.5. Save Message Entries into an XML File

Besides the basic message maintenance functionaries, `MessageManager` also provides the following three overloaded Save method to save message entries into an XML files.

```
public MessageManager: MessageManagerBase
{
    //Others
    public override void Save();❶
    public override void Save(IEnumerable<InternalMessageEntry> messages, string path);❷
    public override void Save(string path, CultureInfo culture);❸
}
public abstract class MessageManagerBase
{
    //Others
    public abstract void Save();
    public abstract void Save(IEnumerable<InternalMessageEntry> messages, string path);
    public abstract void Save(string path, CultureInfo culture);

}
```

❶ Get all culture neutral message entries and save them into an XML file, whose file name is calculated based on current date time. Such an XML file is stored in the current directory;

❷ Save the given message entries into an XML file, whose path is explicitly specified;

❸ Save all given culture specific message entries and save them into an XML file, whose path is explicitly specified.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 5. Code Table

5.1. Introduction

The word *Code* in the phrase *Code Table* stands for an item of a predefined data list, which is generated bound to a particular `ListControl`. For example, in a general registration form, there are usually a Gender `RadioButton` with two options (Male and Female), a Country `DropDownList` with a predefined country list. If your application is built on iConnect.NET, the gender and country list and other similar list data can be managed by Code Table. iConnect.NET Code Table does not only provide you with a full set of application programming interface (API) to perform code maintenance, but can help you to easily achieve code binding as well.

5.2. Getting Started

This section gives you a demonstrative sample application to guide you program Code Table, including how to create a new code and bind a category of codes into a given `ListControl`.

Step I: Create a Windows Form Application and Design the Form

Using your Visual Studio, you first create a new Windows Form project and add the reference to the `NCS.IConnect.CodeTable` assembly, which is located in the /bin folder of installation directory. In the automatically created Windows Form (Form1), drag and drop necessary `Control` into the form and design it according to the following screen shot.

There are two regions in the Windows form. The top region is used for code creation and the bottom region is used to demonstrate code binding. The three `TextBox` in the top region are separately named *txtBoxCategory* , *txtBoxCode* and *txtBoxSequence* , and the `ComboBox` and `ListBox` in the bottom region are separately *comboBoxCategory* and *listBoxCodes* .

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

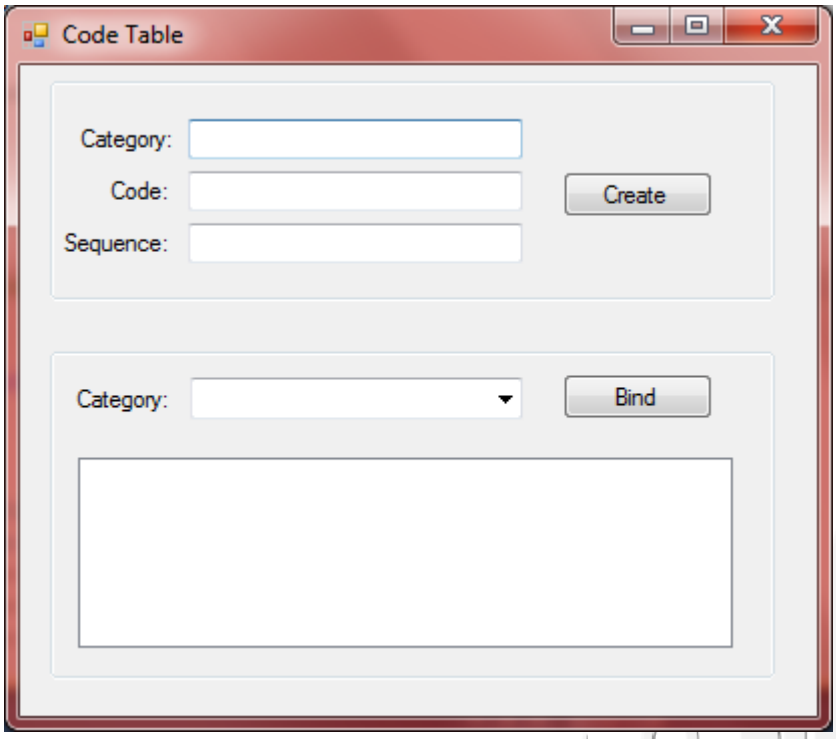


Figure 5.1. Code Table Demonstration From

Step II Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You firstly need to execute the following SQL script in this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*CodeTableDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3B9413F6-3644-4733-8CD6-DA08B203D751'
           ,'CodeTableDemo'
           ,'codetabledemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step III Configure Code Table

Add an App.config in this project, and copy the following XML snippet into the configuration file. Do not forge to revise the connection string accordingly.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="iConnect.codeTable" type="NCS.IConnect.CodeTable.Configuration.CodeTableSettings,
NCS.IConnect.CodeTable" />
  </configSections>

  <connectionStrings>
    <add name="iConnectDb" providerName="System.Data.SqlClient"
         connectionString="..."/>
  </connectionStrings>

  <iConnect.codeTable defaultProvider="dbCodeTableProvider" defaultBindingOption="defaultBindingOption">
    <codeTableProviders>
      <add name="dbCodeTableProvider"
           type="NCS.IConnect.CodeTable.DbCodeTableProvider, NCS.IConnect.CodeTable"
           applicationName="CodeTableDemo"
           database="iConnectDb" />
    </codeTableProviders>
    <controlBindings>
      <add name =" ComboBox"
           controlType="System.Windows.Forms.ListControl, System.Windows.Forms, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
           binderType="NCS.IConnect.CodeTable.CodeBinding.DefaultCodeBinder,NCS.IConnect.CodeTable"
           bindingOptionName="defaultBindingOption"/>
    </controlBindings>
    <bindingOptions>
      <add name="defaultBindingOption" displayTextFormat="{Code}"/>
    </bindingOptions>
  </iConnect.codeTable>
</configuration>
```

Step IV Program Code Table

In the form's load event, write the following code to get all code category list and bind them into the Category `ComboBox`.

```
private void Form1_Load(object sender, EventArgs e)
{
    this.comboBoxCategory.DataSource = CodeManager.GetAllCategories();
}
```

Program the Create button's click event.

```
private void buttonCreate_Click(object sender, EventArgs e)
{
    try
    {
        var codeSet = new CodeSet();
        var codeItem = new CodeItem();
        codeItem.Code = this.txtBoxCode.Text;
        codeItem.Category = this.txtBoxCategory.Text.Trim();
        codeItem.Sequence = int.Parse(this.txtBoxSequence.Text.Trim());
        codeItem.StartDate = DateTime.Today.AddYears(-1);
        codeItem.EndDate = DateTime.Today.AddYears(1);

        codeItem.AuditInfo.CreatedBy = "Sys";
        codeItem.AuditInfo.LastUpdatedBy = "Sys";

        codeSet.Add(codeItem);
        CodeAdmin.SaveCodes(codeSet);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        MessageBox.Show("Code is successfully created!");
        this.comboBoxCategory.DataSource = CodeManager.GetAllCategories();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

The following programing in the following code snippet is the click event for Bind button.

```
private void buttonBind_Click(object sender, EventArgs e)
{
    string category = this.comboBoxCategory.SelectedItem.ToString();
    if (!string.IsNullOrEmpty(category))
    {
        codeHelper.Bind(this.listBoxCodes, category);
    }
}
```

Once completing the above programming and configuration, you can directly run this demonstration application to create codes. Now, we create a list of codes to represent week days. The Category, Code and Sequence for these codes to create are:

- Week - Sunday - 1
- Week - Monday -2
- Week - Tuesday - 3
- Week - Wednesday -4
- Week - Thursday -5
- Week - Friday - 6
- Week - Saturday -7

When you create these codes, you can directly click the Bind button (with the Week category is selected in the Category `ComboBox`) to bind them in the `ListBox`. As the following figure illustrates, the seven week day codes are all listed in the `ListBox` in the Sequence specific order.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

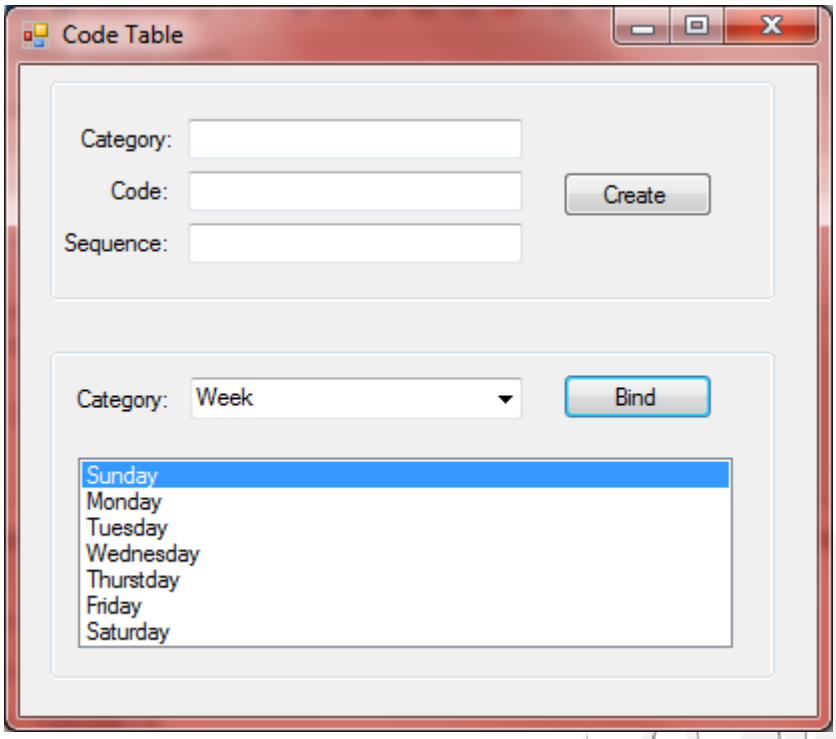


Figure 5.2. Code Table Binding

5.3. Code Effectiveness

Except for general properties, such as Code ID, Code, Code Value, and Sequence, a code has the following two effectiveness based properties:

- *Effective Start Date:* The date when the code take effect;
- *Effective End Date:* The date before which the code is effective.

Both the two properties determine the status of the code. There are totally three status:

- *NotYetEffective:* The current date is earlier than effective start date;
- *Effective:* The current date is between the effective start date and end date;
- *Expired:* The current date is later than effective end date.

The three kinds of code status are defined in the following `CodeStatus` enumeration.

```
[Serializable]
public enum CodeStatus
{
    Effective,
    NotYetEffective,
    Expired,
    UnCommitted,❶
    All
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ UnCommitted means a CodeItem is modified in memory but have not been committed to database.

5.4. Configuration

The following XML snippet represents the whole structure of Code Table configuration section.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="iConnect.codeTable" type="NCS.IConnect.CodeTable.Configuration.CodeTableSettings,
NCS.IConnect.CodeTable"/>
  </configSections>

  <iConnect.codeTable defaultProvider="dbCodeTableProvider" defaultBindingOption="defaultBindingOption">
    <codeTableProviders>❶
      <add name="dbCodeTableProvider"
           type="NCS.IConnect.CodeTable.DbCodeTableProvider, NCS.IConnect.CodeTable"
           applicationName="CodeTableDemo"
           database="iConnectDb"/>❷
      <add name="wcfCodeTableProvider"
           type="NCS.IConnect.CodeTable.WcfCodeTableProvider, NCS.IConnect.CodeTable"
           endpoint="CodeTable"/>❸
    </codeTableProviders>
    <controlBindings>
      <add name="ComboBox"
           controlType="System.Windows.Forms.ListControl, System.Windows.Forms, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
           binderType="NCS.IConnect.CodeTable.CodeBinding.DefaultCodeBinder,NCS.IConnect.CodeTable"
           bindingOptionName="defaultBindingOption"/>
    </controlBindings>
    <bindingOptions>❹
      <add name="defaultBindingOption" appendEmptyItem="true"
           emptyItemDisplayText="Please select an option" emptyItemValue=""
           displayTextFormat="{Value}" valueFormat="{ID}"/>
    </bindingOptions>
   <cache cacheManagerName="Default Cache Manager" enabled="true" expirationTime="00:30:00" cacheMode
="Category" />❺
    <nameConverters>❻
      <add name="default" type="NCS.IConnect.CodeTable.CodeGeneration.UnderscoreTrimNameConverter,
NCS.IConnect.CodeTable"/>
    </nameConverters>
  </iConnect.codeTable>
</configuration>
```

❶ More than one `CodeTableProvider` are configured in the <codeTableProviders/> element;

❷ The configuration of `DbCodeTableProvider`;

❸ The configuration of `WcfCodeTableProvider`;

❹ The <bindingOptions/> contains some settings to control the code binding behavior;

❺ Code caching based configuration, including a switch to enable or disable caching codes; the configuration name of `CacheManager`; and the expiration time;cacheMode can be set to "Category" or "All";

❻ The <nameConverters/> contains the configuration elements of `NameConverter`, which is used for code generation.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

5.5. Code Maintenance

You can use the static `Save` methods of `CodeAdmin` to maintain the codes, including create new codes, and update or delete the existing codes. As the following code snippet illustrates, there are two `Update` method overload. The second is used to maintain the given culture specific code data in multiple languages scenario.

```
public static class CodeManager
{
   //Others
   public static void SaveCodes(CodeSet codeSet);
   public static void SaveCodes(CodeSet codeSet, CultureInfo culture);
}
```

The following code snippet demonstrates how to create new code. You just need to create a new `CodeItem`, and specify all necessary code property specific columns in the newly added data row. Finally submit the newly created code into the database by calling the `Save` method of the `CodeAdmin`.

```
var codeSet = new CodeSet();
var codeItem = new CodeItem();
codeItem.Code = this.txtBoxCode.Text;
codeItem.Category = this.txtBoxCategory.Text.Trim();
codeItem.Sequence = int.Parse(this.txtBoxSequence.Text.Trim());
codeItem.StartDate = DateTime.Today.AddYears(-1);
codeItem.EndDate = DateTime.Today.AddYears(1);

codeItem.AuditInfo.CreatedBy = "Sys";
codeItem.AuditInfo.LastUpdatedBy = "Sys";

codeSet.Add(codeItem);
CodeAdmin.SaveCodes(codeSet);
```

If you need to update or delete the existing codes, you should first retrieve them in the form of `CodeSet`. And then modify or delete the code specific `CodeItem`. Finally submit all of the changes into database in a batch by calling the `Save` method of the `CodeAdmin`.

A series of overloaded SearchCodes methods are defined in the `CodeAdmin`. You can use them to retrieve the codes to maintain. The methods with a culture parameter is used to find the codes with the given culture specific description.

```
public static class CodeAdmin
{
    public static CodeSet SearchAllCodes(CultureInfo culture);
    public static CodeSet SearchCodes(string codeId);
    public static CodeSet SearchCodes(string codeId, CultureInfo culture);
    public static CodeSet SearchCodes(string category, string code, string value);
   public static CodeSet SearchCodes(string category, string code, string value, CultureInfo culture);

}
```

5.6. Code Binding

The most common usage of Code Table is code binding, which retrieve a particular category of codes and bind them to a given `ListControl`. With the help of `CodeHelper`, you can easily

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

perform code binding. You can achieve code binding in the following different manners by calling different methods of `CodeHelper` instance:

- One-shot binding: Directly complete code binding by specifying code category and other related binding behavior information;
- Two-step binding: The code list is first retrieved and then bound to a given `ListControl`.

5.6.1. One-shot Binding

For one-shot code binding, you can call the `Bind` method of `CodeHelper`. As the following code snippet illustrates, there are totally two overloaded Bind methods.

```
public static class CodeHelper
{
   ... ...
   public void Bind(object control, string category, CodeStatus codeStatus = CodeStatus.Effective);
   public void Bind(object control, string category, CodeBindingOption bindingOption, CodeStatus
codeStatus = CodeStatus.Effective❶);
}
```

❶ This parameter determines which status (`Effective`, `NotYetEffective`, `Expired` and `All`) of codes need to be bound to the given control. The effective codes are by default bound;

The `CodeBindingOption` is very important for code binding, as it controls the binding behaviour. The following code snippet represents the definition of `CodeBindingOption`.

```
public class CodeBindingOption
{
    public CodeBindingOption();
    public CodeBindingOption(BindingOptionData bindingOptionSetting);
    public CodeBindingOption(string displayTextFormat, string valueFormat);
    public CodeBindingOption(string displayTextFormat, string valueFormat, bool appendEmptyItem,
string emptyItemDisplayText, string emptyItemValue, BindingPropertyName sortByPropertyName,
SortDirection sortDirection);

    public bool AppendEmptyItem { get; set; }❶
    public string DisplayTextFormat { get; set; }❷
    public string EmptyItemDisplayText { get; set; }❸
    public string EmptyItemValue { get; set; }❹
    public BindingPropertyName SortByPropertyName { get; set; }❺
    public SortDirection SortDirection { get; set; }❻
    public string ValueFormat { get; set; }❼
}
```

❶ A flag indicating whether to add an empty item, like "please select one option.";

❷ Display text format that can consist of flat text and tokens;

❸ The display text for the added empty item;

❹ The value of the added empty item;

❺ Property name based on which the binding is sorted;

❻ Sorted by ascend or descend;

❼ Value format that can consist of flat text and tokens;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

For both the `DisplayTextFormat` and `ValueFormat` property of `CodeBindingOption`, they are a template based text, in which code property specific placeholders can be used. The following lists all of the predefined placeholder: The placeholder is not case sensitive.

- Code ID: {ID}
- Code (Name): {Code}
- Code Value: {Value}
- Code Category: {Category}

If you do not explicitly specify such a `CodeBindingOption` parameter when calling `CodeHelper`'s `Bind` method, the configured `CodeBindingOption` will be used. The following XML snippet represents the `CodeBindingOptions` related configuration.

```
<configuration>
  <iConnect.codeTable defaultBindingOption="defaultBindingOption" ...>
    ...
    <controlBindings>❶
      <add name="ComboBox"
           controlType="System.Windows.Forms.ListControl, System.Windows.Forms, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"
           binderType="NCS.IConnect.CodeTable.CodeBinding.DefaultCodeBinder,NCS.IConnect.CodeTable"
           bindingOptionName="defaultBindingOption"/>
    </controlBindings>
    <bindingOptions>❷
      <add name="defaultBindingOption" appendEmptyItem="true"
           emptyItemDisplayText="Please select an option" emptyItemValue=""
           displayTextFormat="{Value}" valueFormat="{ID}"/>
    </bindingOptions>
  </iConnect.codeTable>
</configuration>
```

❶ The mapped control type and binder type and binding option name;

❷ The binding option configured for formating codes before performing code binding.

When you bind a category of codes to a given `ListControl`, `CodeHelper` always try to get the configured control specific `CodeBindingOption` base on the control's type. If a matched `CodeBindingOption` can be found, it will be used.

5.6.2. Two-Step Binding

If you adopt two-step binding, you need in advance get the codes, and then call an appropriate `Bind` method to bind them to a given `ListControl`. As the following code snippet illustrates, there are two overloaded `Bind` method defined in `CodeHelper`.

```
public static class CodeHelper
{
    public void Bind(object control, CodeSet codeSet);
    public void Bind(object control, CodeSet codeSet, CodeBindingOption bindingOption);
}
```

As with code retrieval for such a two-step binding, you can call a appropriate `GetCode` or `GetCodes` method of `CodeManager`. As the following code snippets illustrates, `CodeManager`

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

provides you with a series of `GetCode` and `GetCodes` method to enable you to get codes based on different search criteria.

```
public static class CodeManager
{
    ... ...
    public static CodeSet GetCodes(ICollection<string> categories, CodeStatus codeStatus =
CodeStatus.Effective);
    public static CodeSet GetCodes(string category, CodeStatus codeStatus = CodeStatus.Effective);
    public static CodeSet GetCodes(string category, string code, CodeStatus codeStatus =
CodeStatus.Effective);
}
```

5.7. Get ID/Value of Code

Because of code effectiveness, the code property cannot uniquely identify a code entry. What is used to uniquely identity a particular code entry is Code ID. Generally speaking, Code ID is a GUID, and Code is a readable name. In order to given the programmer a convenient way to get the Code ID, the following `GetId` methods are defined in the `CodeManager`.

```
public static class CodeManager
{
    ... ...
    public static string GetCodeId(string category, string code);❶
}
```

❶ This method will first retrieve the effective code entry based on the given category and code. If no matched code entry can be found, null returns, and the ID of the code entry is returned otherwise. The code effectiveness is calculated based on database server;

Besides Code ID, Code, a code entry has also another property, Code Value. This property is also used as the Value of the code entry. Similar to get ID of the given code, `CodeManager` also provides the following `GetValue` methods.

```
public static class CodeManager
{
    ... ...
    public static string GetValue(string codeId);
    public static string GetValue(string category, string code);
}
```

5.8. Code Authorization

In general, the codes are public for all users. However, we need to apply access control policy to some codes, and make them available for the permitted users. Code authorization based application programming interfaces are mainly exposed by `CodeAdmin`. You can call the following `AssignRole` or `AssignRoles` to assign the specified role(s) to the given code.

```
public static class CodeAdmin
{
    public static void AssignRole(string codeId, string role);❶
    public static void AssignRole(string category, string code, string role);❷
    public static void AssignRoles(string category, string code, ICollection<string> roles);❸
    public static void AssignRoles(string codeId, ICollection<string> roles);❹
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ Assign one single role to the code of the given Code ID.

❷ `CodeAdmin` will first get the effective code entry based on the given category and code, and assign the specified role to it;

❸ Assign a set of roles to the code of the given code ID.

❹ This overload is similar to the second one, and only difference is that what is assigned to target code is a role list other than a single role.

Note

When assign one or more roles to the given code(s) by calling the above methods, an exception will be thrown if any role has been assigned to the target code.

By assigning a role to a given code, a role-code mapping is built in the database. If a particular mapping needs to be removed, you can call the following `RemoveRole`, `RemoveRoles` and `RemoveAllRoles` method of `CodeAdmin`.

```
public static class CodeAdmin
{
    ... ...
    public static void RemoveRole(string codeId, string role);❶
    public static void RemoveRole(string category, string code, string role);❷
    public static void RemoveRoles(string codeId, ICollection<string> roles);❸
    public static void RemoveRoles(string category, string code, ICollection<string> roles);❹

    public static void RemoveAllRoles(string codeId);❺
    public static void RemoveAllRoles(string category, string code);❻
}
```

❶ Remove the given role from the code of the specified Code ID;

❷ `CodeAdmin` will first get the effective code entry based on the given category and code, and locate the corresponding role-code mapping, and finally remove the mapping;

❸ Remove the given roles from the code of the specified Code ID;

❹ This method is similar to the second one, and only difference is that what is removed is a role list other than singe role;

❺ Remove all assigned roles from the code of the given Code ID;

❻ `CodeAdmin` will first get the effective code entry based on the given category and code, and then remove all assigned roles from it.

Note

When remove the assigned role(s) from the given code, an exception will be thrown if any one role is not assigned to the target code.

Except for the methods used for the two kinds of basic code authorization operations: assigning role(s) to a given code and removing role(s) from a given code, there are following two extra `GetRoles` methods defined in the `CodeAdmin`. The two `GetRoles` methods are used to get all roles which are assigned to the given code.

```
public static class CodeAdmin
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    ... ...
    public static List<string> GetRoles(string codeId);
    public static List<string> GetRoles(string category, string code);
}
```

Code authorization setting finally serve code binding. In order to bind the authorized codes into the target `ListControl`, you need to call the `BindAuthorizedCodes` method of `CodeHelper`. As the following code snippet illustrates, this method accept a *`userName`* as the last parameter. `CodeAdapter` will first get the authorized codes based on all roles assigned to the given user account.

```
public static class CodeAdapter
{
    ... ...
    public void BindAuthorizedCodes(object control, string category);
    public void BindAuthorizedCodes(object control, string category, string userName);
    public void BindAuthorizedCodes(object control, string category, CodeBindingOption bindingOption
                                    , userName, CodeStatus codeStatus = CodeStatus.Effective);
}
```

The `BindAuthorizedCodes` method gives you a convenient way to only bind the authorized codes in an one-shot manner. If you need to adopt two-step binding, you should first manually get the authorized codes by calling the following `GetCodesByRole` , `GetCodesByRoles` methods. These methods enable you to get the authorized codes based on the given role(s).

```
public static class CodeManager
{
    ... ...
    public static CodeSet GetCodesByRole(string category, string role, CodeStatus codeStatus =
CodeStatus.Effective);
    public static CodeSet GetCodesByRoles(string category, ICollection<string> roles, CodeStatus
codeStatus = CodeStatus.Effective);
}
```

5.9. Code Generation

Code binding is generally based on code category, so we have to manually specify the category of the codes to bind. In addition, `GetId` and `GetValue` methods are defined in `CodeManager` to get the ID and Value of the given code entry, however, we has to manually specify the code of the code entry. For these two typical programming, although we usually define all the code category names and code name as constants, it is not convenient. We can leverage code generation to resolve this.

Code Table based code generator can automatically retrieve all of the codes from database, and generate a C# or VB.NET file. With the help of the generated classes, you can easily get a particular code in a strongly typed manner. The following code snippet represents what the generated source is like.

```
public partial class Codes {

    private static GenderCodeCategory gender = new GenderCodeCategory("Gender");
    private static WeekCodeCategory week = new WeekCodeCategory("Week");

    public static GenderCodeCategory Gender { get { return Codes.gender; } }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public static WeekCodeCategory Week { get { return Codes.week; } }
}

public sealed class GenderCodeCategory : NCS.IConnect.CodeTable.CodeGeneration.CodeCategory {

    public GenderCodeCategory(string name) :  base(name) {}

    public NCS.IConnect.CodeTable.CodeGeneration.CodeEntry Male {
       get { return NCS.IConnect.CodeTable.CodeGeneration.CodeEntry.GetCodeEntry("Gender", "Male"); }
    }

    public NCS.IConnect.CodeTable.CodeGeneration.CodeEntry Female {
       get { return NCS.IConnect.CodeTable.CodeGeneration.CodeEntry.GetCodeEntry("Gender", "Female");}
    }
}

public sealed class WeekCodeCategory : NCS.IConnect.CodeTable.CodeGeneration.CodeCategory {
    public WeekCodeCategory(string name) : base(name) { }

    public NCS.IConnect.CodeTable.CodeGeneration.CodeEntry Sunday {
       get { return NCS.IConnect.CodeTable.CodeGeneration.CodeEntry.GetCodeEntry("Week", "Sunday"); }
    }

    public NCS.IConnect.CodeTable.CodeGeneration.CodeEntry Monday {
       get { return NCS.IConnect.CodeTable.CodeGeneration.CodeEntry.GetCodeEntry("Week", "Monday"); }
    }

    public NCS.IConnect.CodeTable.CodeGeneration.CodeEntry Tuesday {
       get { return NCS.IConnect.CodeTable.CodeGeneration.CodeEntry.GetCodeEntry("Week", "Tuesday"); }
    }
}
```

Code category and code specific classes separately inherit `CodeCategory` and `CodeEntry`. They both are defined in Code Table. The following code snippet represents the definition of `CodeCategory`. The read-only property Name represents the category name, and the series of overloaded `Fill` methods are used to bind the current category of codes to the given Control.

```
public class CodeCategory
{
    public CodeCategory(string name);
    public virtual void Fill(object control, CodeStatus codeStatus = 1);

    public string Name { get; private set; }
}
```

The following code snippet represents the definition of `CodeEntry`. All forementioned code specific property can be access from a `CodeEntry` instance.

```
public class CodeEntry
{
    public CodeEntry();
    public static CodeEntry GetCodeEntry(string category, string code);

    public string Code { get; private set; }
    public DateTime EffectiveEndDate { get; private set; }
    public DateTime EffectiveStartDate { get; private set; }
    public string Id { get; private set; }
    public int Sequence { get; set; }
    public string Value { get; private set; }
}
```

Code generation used in Code Table is based on T4. If you want to generate the code based on your code data stored in database, you just need to add a T4 template file in the target project. The following code snippet is a demonstrative T4 template file, in which a

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`CodeTableGenerator` is created and its `Run` method is called to get code data and generate the code.

```
<#@ template  language="C#" hostSpecific="true" debug="true" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Common.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.CodeTable.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version
no}\bin\Microsoft.Practices.EnterpriseLibrary.Caching.dll" #>

<#@ import namespace="NCS.IConnect.CodeTable.CodeGeneration" #>
<#@ import namespace="System.Diagnostics" #>
<#@ import namespace="System.CodeDom.Compiler" #>
<#@ import namespace="NCS.IConnect.Common" #>
<#@ import namespace="NCS.IConnect.Core.CodeGeneration" #>
<#@ include file="T4Toolbox.tt" #>
<#@ output extension="cs" #>
<#
   CodeTableGenerator generator = new CodeTableGenerator(this.Host.ResolvePath("CodeTable.config"❶),
                                   Languages.CSharp❷, new CodeGeneratorOptions()❸,
                                   "NCS.IConnect.CodeTable.CodeGeneration"❹);
   generator.Run();
#>
```

❶ The Code Table configuration file's path, which is relative to the current template file;

❷ The parameter determines the programming language of the generated code;

❸ The `CodeGeneratorOptions` parameter is used to control the style of the generated code;

❹ The namespace under which the generated classes is.

For Code Table based code generation, there is an issue about name conversion. Because different naming conventions are separately used in database design and managed code programming. We have to convert the code name and category name stored in database to the class name or property name. In iConnect.NET Code Table, such a name conversion is implemented in a `NameConverter`. All of the `NamerConverter` implement the following `INamerConverter` interface. The only method `GetClassName` is called to perform name conversion from real value to class name.

```
public interface INameConverter
{
    string GetClassName(string codeCategory);
}
```

The `NamerConverter` is specified in the configuration file. The following XML snippet is a demonstrative Code Table configuration, which is only contain the `NameConverter` related configuration. The `DefaultNameConverter` is a built-in `NameConverter` defined in iConnect.NET.

```
<configuration>
  ...
  <iConnect.codeTable defaultProvider="dbProvider"
                      defaultNameConverter="defaultNameConverter"❶>
   ...
    <nameConverters>❷
      <add name="defaultNameConverter"
           type="NCS.IConnect.CodeTable.CodeGeneration.DefaultNameConverter,
                 NCS.IConnect.CodeTable"/>
     <add .../>
    </nameConverters>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  </iConnect.codeTable>
</configuration>
```

❶ The configuration name of default `NameConverter`;

❷ The named `NameConverter` configuration element list.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 6. Data Mapping

6.1. Introduction

For most of applications, especially for data centric applications, there are two common operations: data binding and data capture. In general, the business data is firstly retrieved base on the search criteria in the form of business entity instance, for example DataSet or Entity collection. The related data items encapsulated in the business entity will be bound to the corresponding controls in the current windows application or web application.

Once the source data is completely displayed, the end user can enter new value in these input controls. Prior to submitting, all of the new values in these input controls will be captured and become the new properties of the newly created or existing business entity instance.

Regardless of data binding and data capture, they can built on such a basis that a mapping between the control and its corresponding property of the business entity type. *iConnect.NET Data Mapping* gives you an easy-to-use way to build such a mapping. Furthermore, for the applications using strongly `DataSet` as the business entity, the mapping is used to associate the control with a column of a given `DataTable`, and for the applications use Entity collection as the business entity, the mapping is used to associate the control with entity property.

Generally speaking, Data Mapping is a building block in iConnect.NET. The developers do not directly program with it. The above mentioned two functionalities, data binding and data capture, which are both exposed in two helper classes: `NCS.IConnect.Helpers.WinForm.FormHelper` and `NCS.IConnect.Helpers.Web.PageHelper`, which are both built on the basis of data mapping. As their names suggest, `FormHelper` and `PageHelper` are separately used in windows application and web application.

With the help of data mapping, the two helpers can find the corresponding control specific to a given column of a `DataRow` (or a property of the entity) when performing data binding. On the other hand, they leverage the data mapping to determine which column of the `DataRow` (or which property of the entity) should be overridden with value entered in a given input control when conducting data capture.

6.2. Getting Started

This section will give you a demonstrative ASP.NET web application which guides you how to configure data mapping, how to name your web control according to the mapping rule, and how to use `PageHelper` to perform data binding and data capture.

Step I Create ASP.NET Application and Design the Demonstration Page

We fist create a ASP.NET web application using your Visual Studio, and add the following two assembly reference:

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- NCS.IConnect.DataMappings.dll
- NCS.IConnect.Helpers.Web.dll

After that you can design your web page. You can directly use the default page, which is automatically added when the project is created. The following figure represents the screen shot that the page is shown in the browse.

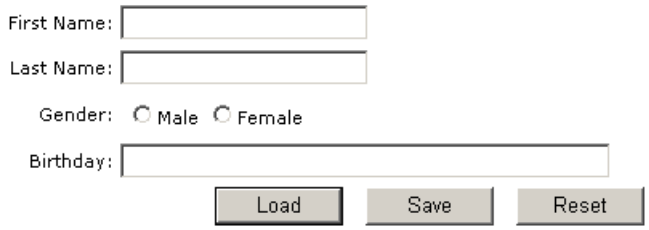


Figure 6.1. Data Mapping form screen shot

If you do not want to manually design the above web page. You can directly copy the following HTML to override the one of existing default page.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="DataMappingDemo.Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server">
        <title>Data Mapping Demonstration</title>
        <style type="text/css">
            body{font-family:Verdana; font-size:11px}
            .longtxt{width:300px;}
            .btn{width:80px; margin-right:10px;}
        </style>
    </head>
    <body>
    <form id="form1" runat="server">
    <div>
      <table width="100%">
        <tr>
            <td style="width:20%; text-align:right">First Name:</td>
            <td><asp:TextBox ID="FirstName" runat="server"></asp:TextBox></td>
        </tr>
         <tr>
            <td style="width:20%; text-align:right">Last Name:</td>
            <td><asp:TextBox ID="LastName" runat="server"/></td>
        </tr>
        <tr>
            <td style="width:20%; text-align:right">Gender:</td>
            <td><asp:RadioButtonList runat ="server" ID="Gender" RepeatDirection ="Horizontal">
                <asp:ListItem Text="Male" Value="M" />
                <asp:ListItem Text="Female" Value="F" />
            </asp:RadioButtonList></td>
        </tr>
        <tr>
            <td style="width:20%; text-align:right">Birthday:</td>
            <td><asp:TextBox ID="Birthday" runat="server" CssClass="longtxt" /></td>
        </tr>
        <tr>
            <td colspan="2" style="text-align:center">
                <asp:Button runat="server" ID="btnLoad" Text="Load" onclick="btnLoad_Click"
CssClass="btn" />
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                <asp:Button runat="server" ID="btnSave" Text="Save" onclick="btnSave_Click"
CssClass="btn"/>
                <asp:Button runat="server" ID="btnReset" Text="Reset" CssClass="btn"
onclick="btnReset_Click" />
            </td>
        </tr>
      </table>
    </div>
    </form>
</body>
</html>
```

Step II: Configuring DataMapping and `PageHelper`

After the project is created, the Web.config will be automatically added. You can directly copy the following configuration to override the existing content of the configuration file.

```
<?xml version="1.0"?>
<configuration>
 <configSections>
  <section name="iConnect.dataMapping"
           type="NCS.IConnect.DataMappings.Configuration.MappingSettings,
                 NCS.IConnect.DataMappings"/>
 </configSections>
 <iConnect.dataMapping>
  <namingConverter type="NCS.IConnect.DataMappings.AspNetNamingConverter,
                        NCS.IConnect.DataMappings"/>
  <sourceSettings>
   <add name="Decimal" type="System.Decimal" formatString="#.#00.00"/>
   <add name="DateTime" type="System.DateTime" formatString="dd-MM-yyyy"/>
  </sourceSettings>
  <controlSettings>
   <add name="DropDownList"
    type="System.Web.UI.WebControls.DropDownList,
          System.Web"
    bindingProperty="SelectedValue"/>
   <add name="CheckBox"
    type="System.Web.UI.WebControls.CheckBox,
          System.Web"
    bindingProperty="Checked"/>
   <add name="RadioButtonList"
    type="System.Web.UI.WebControls.RadioButtonList,
          System.Web"
    bindingProperty="SelectedValue"/>
  </controlSettings>
 </iConnect.dataMapping>
 <system.web>
</configuration>
```

Step III Programming Data Binding and Data Capture

The following code snippet represents the entire default page specific code behind. In which a manually created ADO.NET `DataTable` is used as the data source to bind to controls by calling the `FillPage` method of `PageHelper`. Data binding is performed by clicking the Load button. Once the Save button clicked, the `UpdateData` of `PageHelper` is invoked to capture the data entered into the mapped control and finally to save into a `DataRow` object.

```
using System;
using System.Data;
using NCS.IConnect.Helpers.Web;
namespace DataMappingDemo
{
    public partial class Default : System.Web.UI.Page
    {
        static PageHelper helper = new PageHelper();
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        private DataTable Person
        {
            get { return (DataTable)this.ViewState["Person"]; }
            set { this.ViewState["Person"] = value; }
        }
        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack)
            {
                this.Person = CreateDataTable();
            }
        }
        protected void btnLoad_Click(object sender, EventArgs e)
        {
            helper.FillPage(this.Person.Rows[0], this);
        }
        static DataTable CreateDataTable()
        {
            DataTable table = new DataTable();
            table.Columns.Add("FirstName", typeof(string));
            table.Columns.Add("LastName",  typeof(string));
            table.Columns.Add("Gender",    typeof(string));
            table.Columns.Add("BirthDay",  typeof(DateTime));
            table.Rows.Add("Jin Nan", "Jiang", "M", new DateTime(1981, 8, 24));
            return table;
        }
        protected void btnSave_Click(object sender, EventArgs e)
        {
            helper.UpdateData(this.Person.Rows[0], this);
        }
        protected void btnReset_Click(object sender, EventArgs e)
        {
            this.FirstName.Text       = string.Empty;
            this.LastName.Text        = string.Empty;
            this.Birthday.Text        = string.Empty;
            this.Gender.SelectedValue = "M";
        }
    }
}
```

Run this application and show the default page in the browser. You can click the Load button to bind the data into the four controls in the page. After entering the new values in these input controls can press Save button, the new value will be stored in a `DataRow` instance, whose container `DataTable` is stored as a view state item. After clicking the Reset to clear all input control and press `Load` button, you will see what are bound are the new values.

Step IV Use Entity Object as Data Source

So far, what is used as the data source is a `DataSet`. Finally we will demonstrate how to use a simple entity object as the data source. We first need to create an entity class: Person. As illustrated in the following code snippet, all of the property to bind must be decorated with a `DataPropertyAttribute`, which is defined in the `NCS.IConnect.EntityFramework` assembly, so you must add such an assembly reference.

```
public class Person
{
   [DataProperty("LastName")]
   public string LastName{get;set;}

   [DataProperty("FirstName")]
   public string FirstName{get;set;}

   [DataProperty("Birthday")]
   public DateTime Birthday{get;set;}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
   [DataProperty("Gender")]
   public string Gender{get;set;}
}
```

Then we use the following code behind to override the exiting one.

```
using System;
using System.Data;
using System.Runtime.Serialization;
using NCS.IConnect.EntityFramework;
using NCS.IConnect.Helpers.Web;
namespace DataMappingDemo
{
    public partial class Default : System.Web.UI.Page
    {
        static PageHelper helper = new PageHelper();
        private List<Person> Person
        {
            get { return (List<PersonDetail>)this.Session["Person"]; }
            set { this.Session["Person"] = value; }
        }

        static List<Person> CreateData()
        {
            List<Person> list = new List<Person>();
            var personDetail = new PersonDetail();
            personDetail.FirstName = "Tomhe";
            personDetail.LastName = "Adison";
            personDetail.BirthDate = new DateTime(1981, 8, 24);
            personDetail.Gender = "M";
            list.Add(personDetail);
            return list;
        }

        protected void Page_Load(object sender, EventArgs e)
        {
            if (!this.IsPostBack)
            {
                this.Person = CreateData();
            }
        }

        protected void btnLoad_Click(object sender, EventArgs e)
        {
            helper.FillPage(this.Person[0], this);
        }
        protected void btnSave_Click(object sender, EventArgs e)
        {
            helper.UpdateData(this.Person[0], this);
        }
        protected void btnReset_Click(object sender, EventArgs e)
        {
            this.FirstName.Text        = string.Empty;
            this.LastName.Text         = string.Empty;
            this.BirthDate.Text        = string.Empty;
            this.Gender.SelectedValue  = "M";
        }
    }
}
```

Data mapping is essentially a series of name mapping between control and `DataColumn` (or Entity property). In order to bind the data source to the target controls, the control names needs to be resolved based on the name mapping. On the other hand, the column name (or Entity property name) need to be resolve when you save back the new values captured from the controls back into the data source. The two kinds of functionalities are implemented in a special component: *NamingConverter*.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

6.3. NamingConverter

As its name suggests, *NamingConverter* is used to convert the name of `DataColumn` (or the name of Entity property) to the control name, and vice versa. All of `NamingConverter` types are directly or indirectly implement an interface: `NCS.IConnect.DataMappings.INamingConverter`. The following code snippet represents its definition.

```
public interface INamingConverter
{
    string GetControlName(string dataColumnName);
    string GetDataColumnName(string controlName);
}
```

There is a base *NamingConverter* `NamingConverterBase`, which implements the above interface. The default mapping rule is defined in `NamingConverterBase`. Let's give you an exception to explain what the default name mapping is like. Assuming the column name is "ABC_DEF", the name of the mapped control should be "*AbcDef*". That means for the given column name, the following name conversion rule will be applied to get the mapped control name:

- The column name will be split multiple string parts if it includes underscore ("_"), otherwise there is only one string part as the same as the raw column name. E.G. ABC_DEF=>ABC, DEF;
- For each part, capitalize the first letter and convert the rest to low case. E.G. ABC=>Abc, DEF=>Def;
- Concatenate the above all parts as a string string, which is used as the mapped control name. E.G. Abc+Def=>AbcDef.

The above demonstration uses another `NamingConverter`: NCS.IConnect.DataMappings.AspNetNamingConverter, which derives from `NamingConverterBase` . As its name suggests, it is used in ASP.NET web application for building web control specific data mapping. You can also create custom `NamingConverter` if you need to apply different mapping rule. You can let your custom `NamingConverter` inherit from `NamingConverterBase`, or directly implement `INamingConverter` interface.`NamingConverterBase`, `AspNetNamingConverter` and custom `NamingConverter` should be configured. In following XML snippet, the `NamingConverterBase` is configured.

```
<?xml version="1.0"?>
<configuration>
 <iConnect.dataMapping>
  <namingConverter type="NCS.IConnect.DataMappings.AspNetNamingConverter,
                        NCS.IConnect.DataMappings"/>
  ...
</configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

6.4. Specifying the Data Format

When you perform data binding, you need to consider the value is displayed in appropriate format. For example, different countries/regions have different date. For a decimal or double number, different precision is needed in different scenario. Thanks to the data mapping configuration system, you can freely set any data type specific display format. In the following code snippet, we set the format string for the following three data types: `Decimal`, `Double` and `DateTime`.

```
<Configuration>
   <iConnect.dataMapping>
      <sourceSettings>
         <add name="Decimal" type="System.Decimal" formatString="$#.#00.00" />
         <add name="Double" type="System.Double" formatString="#.#0.00" />
         <add name="DateTime" type="System.DateTime" formatString="dd-MM-yyyy" />
      </sourceSettings>
   </iConnect.dataMapping>
</Configuration>
```

6.5. Specifying the Bound Property of Control

Besides the special data type specific display format, there is another consideration related to data binding: Which property of the control is the value is bound to? The bound property of the control is by default Text. This suits for most general text control, such as TextBox, Label, etc.

However, there are many controls which cannot adopt such a default data binding. E.G. When we bind the data source to a `DropDownList` control, the bound property should be `SelectedText`, other than Text. For `RadioButtonList` control in the above demonstration, the bound property is `SelectedValue`, other than Text. The control type specific bound property can be configured in the *<iConnect.dataMapping>/<controlSettings>*. In the following demonstrative configuration, we set the bound properties specific the following three control type: `DropDownList`, `CheckBox` and `RadioButtonList`.

```
<Configuration>
   <iConnect.dataMapping>
      <controlSettings>
         <add name="DropDownList"
              type="System.Web.UI.WebControls.DropDownList,
                    System.Web"
              bindingProperty="SelectedValue"/>
         <add name="CheckBox"
              type="System.Web.UI.WebControls.CheckBox,
                    System.Web"
          bindingProperty="Checked"/>
         <add name="RadioButtonList"
              type="System.Web.UI.WebControls.RadioButtonList,
                    System.Web"
          bindingProperty="SelectedValue"/>
      </controlSettings>
   </iConnect.dataMapping>
</Configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

6.6. BindingMapping

`DataMapping` builds a relationship between controls and its data source. With the help of the mapping information, `FormHelper` and `PageHelper` can easily perform the data binding and data capture. The data mapping is essentially the name mapping between the control and `DataColumn` (or Entity property). Such a mapping can be represented in a `NCS.IConnect.DataMappings.BindingMapping` object. The following code snippet represents the definition of `BindingMapping`. Not only are BindingName and target control (`Control`) defined, there are also some other properties which are related to behavior of data binding and data capture.

```
public class BindingMapping
{
    public BindingMapping(string bindingName, object control, string bindingProperty,
        Type dataType, string format);
    public BindingMapping(string bindingName, object control, string bindingProperty,
        Type dataType, string format, bool allowUpdate, bool allowFill);

    public bool   AllowUpdate { get; set; }❶
    public bool   AllowFill { get; set; }❷
    public object Control { get; set; }❸
    public string BindingProperty { get; set; }❹
    public string BindingName { get; set; }❺
    public Type   DataType { get; set; }❻
    public string FormatString { get; set; }❼
}
```

❶ Indicates whether the value should be captured and saved when invoking the `UpdateData` method of PageHelper/FormHelper;

❷ Indicates whether the control should be bound when invoking the `FillPage` method of PageHelper/FormHelper;

❸ The web control or windows form control of the mapping;

❹ The property name of control, which the data source will be bound to;

❺ The name of the `DataColumn` (or Entity property);

❻ The data type of the `DataColumn` (or Entity property);

❼ The format string to control how the value should be displayed;

In the same namespace, `BindingMapping`'s collection type `BindingMappingCollection` is also defined. `BindingMappingCollection` directly derives from `List<BindingMapping>`, and its definition is represented in the following code snippet. The Container property, which indicates a container control, defines the mapping collection's scope.

```
public class BindingMappingCollection : List<BindingMapping>
{
    public BindingMappingCollection(object container);
    public object Container {get;  private set; }
    public BindingMapping this[object control] { get; }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 7. Form Helper

7.1. Introduction

As its name suggests, `FormHelper` is a helper class used in Windows application. This helper class is defined in the `NCS.IConnect.Helpers.WinForm` assembly, whose name also indicates its namespace. There are two main functionalities achieved by this helper class: data binding and data capture, which are both built on the basis of data mapping. Thanks to the mapping between control and `DataColumn`/Entity class, `FormHelper` can easily bind a `DataRow` to the target controls. With the help of such a mapping, the value entered in the control can also easily be saved into the corresponding column of a given `DataRow`.

Besides the above basic functionalities, other additional ones are provided by `FormHelper`. E.G Enable or disable all of the controls nested in a given container control in a batch.

There are a series of methods and events defined in `FormHelper`. You can invoke these methods to perform data binding, data capture, etc. The events can also be registered to control the behavior of data binding and data capture. This section will provide you with the detailed introduction to these application programming interfaces.

7.2. Data Binding

As the following code snippets illustrated, there are three `FillForm` overloaded methods defined in `FormHelper`. Which gives you different manners to bind a data object to the target controls in the given container control. There are two methods can address your most data binding requirements in variety of scenarios.

```
public class FormHelper
{
    public virtual void FillForm(Object entity, BindingMappingCollection mappings)
    public virtual void FillForm(Object entity, Control container, string mappingPrefix = "")
    //Others
}
```

7.2.1. Data Binding Based on Given `BindingMappingCollection`

As you know, data binding is complete built on the basis of data mapping. We can manually build the data mapping set, and then deliver it to `FormHelper` for conducting customized data mapping. This kind of data binding makes it possible to fully control the binding behavior. E.G.You can manually ignore a particular control in the give container control if it should not be bound. You can also manually specify a column specific format if it needs to be different from the configuration. In this scenario, you can invoke the `FillForm` method with a `BindingMappingCollection` parameter.

`ForHelper` gives you am easy-to-use manner to build a `BindingMappingCollection` based on the given data source object and container control. You just need to invoke the following

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`BuildMapping` methods. Once the method is invoked, `FormHelper` will build a `BindingMapping` collection based on the given `DataTable`'s column set or entity type's data properties and container control's sub controls, including nested sub-controls.

```
public class FormHelper
{
    public BindingMappingCollection BuildMappings(Object entity, Control container, string
mappingPrefix = "")
    //Others
}
```

The method builds the `BindingMapping` according to the `NamingConverter` specific default mapping rule. If you do not familiar with `NamingConverter`, please refer to the chapter of Data Mapping. The optional *mappingPrefix* parameter is used in such a scenario that you need to bind a single column to more than one control. For example, there are one Label and one TextBox, which are both bound to the *USER_NAME* column. According to the default mapping rule, the target control's name must be UserName. However, there are two target controls, which cannot share the same name. In this situation, you need to add an appropriate prefix to name the controls.

If the ID of the Label and the `TextBox` are UserName, and *txtUserName*, you can use the following two different programming to separately bind the data source (*USER_NAME* column) to them.

```
//1. UserName Label
BindingMappingCollection mapping1 = helper.BuildMapping(dataTable, this);
helper.FillForm(dataTable.Rows[0], mapping);

//2. txtUserName TextBox
BindingMappingCollection mapping2 = helper.BuildMapping(dataTable, this, "txt");
helper.FillForm(dataTable.Rows[0], mapping);
```

If there is a control (E.G. *txtCustName* `TextBox`) in the given container control that you need not to bind, you can build the container specific mapping collection and set the control specific `BindingMapping`'s `AllowFill` as false prior to binding. The following is the demonstrative programming.

```
BindingMappingCollection mapping = helper.BuildMapping(dataTable, this);
mapping["txtCustName"].AllowFill = false;
helper.FillForm(dataTable.Rows[0], mapping);
```

7.2.2. Data Binding Based on Automatically Built `BindingMappingCollection`

You can directly invoke follow two `FillForm` methods with container parameter to bind a given data source object to all of matched sub controls in the given container control. Before binding, the container specific data mapping will be automatically built, which ultimately serve the final data mapping.

```
public class FormHelper
{
    public virtual void FillForm(Object entity, Control container, string mappingPrefix = "")
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    //Others
}
```

7.2.3. Control the Data Binding Behavior by Registering Event

Once the `FillForm` method is invoked, a `DataBinding` event will be raised when every control is attempt bound. At other hand, the `DataBound` event will be raised after data binding is completed. You can register such two events to change the default data binding behavior. The following code snippet represents the `DataBinding` and `DataBound` event definition.

```
public class FormHelper
{
    public event EventHandler<DataBindingEventArgs> DataBinding;
    public event EventHandler<DataBindingEventArgs> DataBound;
    //Others
}
```

This event specific `EventArgs` type is NCS.IConnect.Helpers.WinForm.DataBindingEventArgs. The following code snippet illustrates its definition.

```
public sealed class DataBindingEventArgs : EventArgs
{
    public Control       BindingControl {get;}❶
    public object        BindingValue {get; set;}❷
    public BindingMapping Mapping {get;}❸
    public Object        Entity {get;}❹
}
```

❶ The control to which the data source is bound;

❷ The value which is bound to the target control;

❸ The currently used `BindingMapping`;

❹ The data source object.

Except for the `BindingValue`, the rest properties are all read-only. If the value to bind need to be customized, you can register this event and set an appropriate value by set the BindingValue property. Following code shows how to bind data into a `CheckBox` control. The `DataTable` includes a `DataColumn` named IsExpired. The value of this column can be Y or N. From the configuration, the value will be bound to Checked property. This property only accepts `Boolean` value.

```
FormHelper helper = new FormHelper();
helper.DataBinding += new DataBindingEventHandler(DataMapping_DataMappingBinding);

void DataMapping_DataMappingBinding(object sender, DataBindingEventArgs e)
{
  if (sender is CheckBox)
     {
        // BindingValue means the value will be assigned to control
        e.BindingValue = e.BindingValue.ToString () == "Y”? true: false;
      }
 }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

7.3. Data Capture

Generally, user will click a button to submit the data changes made on UI. Developers will write some codes to capture the values from controls and assign these values into business entity one by one. If the naming conventions meet the requirement, you can use `UpdateData` method to commit the changes into business entity by batch, instead one by one.

```
public class FormHelper
{
    public virtual void UpdateData(Object entity, BindingMappingCollection mappings) ;
    public virtual void UpdateData(Object entity, Control container, string mappingPrefix = "")
    //Others
}
```

Data capture, which also depends on data mapping, is the opposite operation of data binding. Data binding leverages the data mapping to get the target control based on column, whereas data capture uses the data mapping to get the source column name based on the column name. From the definition of the three `UpdateData` methods of `FormHelper` represented in the above code snippet, you can find they are similar to the definition of `FillForm` methods.

You can perform data capture based on the pre-built `BindingMappingCollection` by invoke the first overloaded method. You can also invoke the other overloaded methods to capture the data based on the automatically build data mapping, even prefix based data mapping.

Similar to data binding, there is one pair of pre-data-capture and post-data-capture events: `DataUpdating` and `DataUpdated`. As the following code snippet illustrates, they two have the same declaration as the aforementioned `DataBinding` and `DataBound` events.

```
public class FormHelper
{
    public event EventHandler<DataBindingEventArgs> DataUpdating;
    public event EventHandler<DataBindingEventArgs> DataUpdated;
    //...
}
```

7.4. Enabling/Disabling Controls

Beside the two frequently used functionalities, data binding and data capture, it is also a common programming to enable or disable a batch of controls, which are usually located in a given container control. You can invoke the following three `SetEnabled` methods to achieve such functionality.

```
public class FormHelper
{
    public void SetEnabled(Control container❶, bool enabled❷);
    public void SetEnabled(Control container, bool enabled, bool fixedStatus❸);
    public virtual void SetEnabled(Control container, bool enabled, bool fixedStatus,
        params Control[] excludedChildControls❹);
    //Others
}
```

❶ The container control in which the sub controls need to be enabled or disabled;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❷ Indicate whether to enable or disable the target controls;

❸ Indicating whether to prevent to change the enabled property.

❹ The sub control list which should be ignored to enable or disable.

Considering accessibility, the disabled TextBox cannot be focused, so it will become read only other than disabled by invoking this method with enabled parameter equals false.

7.5. Reset Controls

`SetEnabled` method give to a way to enable or disable multiple controls in a batch. If you need to reset value of multiple controls in batch, you can invoke the following two `ResetControl` methods.

```
public class FormHelper
{
    public void ResetControl(Control container❶);
    public void ResetControl(Control container, params Control[] excludedChildControls❷);
    //Others
}
```

❶ The container control in which the sub controls need to be reset;

❹ The sub control list which should be ignored to reset.

For `TextBox` control, the Text property will be reset as an empty string. The Checked property will be reset as false when a `CheckBox` control is reset. When `ListControl` is reset, its SelectedIndex will be specified as -1.

7.6. Working with Menus

The method `GenerateMenuItem` can generate the Menu items on UI. You may use SiteMap functions to retrieve menu items first before calling this method.

```
// get the menus for authenticated user.
MenuDataSet ds = Menus.GetMenus(userName);

// bind the menu to the MenuStript using method in helper class
helper.MenuItemClick += new EventHandler(MenuItemClick);
helper.GenerateMenuItem(this.menuStripMain, ds.T_IC_MENUS);
```

Following code shows how to handle user click for Menu item.

```
private void MenuItemClick(object sender, EventArgs e)
{
   ToolStripMenuItem menuItem = (ToolStripMenuItem)sender;
   if (!string.IsNullOrEmpty(menuItem.Tag.ToString()))
     {
        Type type = Type.GetType(menuItem.Tag.ToString());
        if (type == null)
        {
        // configuration error.
        return;
        }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        // create the form instance using reflection.
        Form form = (Form)Activator.CreateInstance(type);
        form.MdiParent = this;
        form.Show();
    }
}
```

Note that the Tag property of `TollStripMenuItem` is used here. For windows application, it'll contain the full type name of the Form, and was assigned when calling `GenerateMenuItem`. For example, the Tag value can be the Form name NCS.Sample.WinUI.AccessControl. If the Form is in another assembly, you need to include the assembly name too.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 8. Page Helper

8.1. Introduction

Page Helper provides utility methods working with ASP.NET web pages to load data from business entity and update data to business entity in batch, thus a lot of tedious code can be saved. And several events are provided for customization. At the same time, additional methods to set control status in batch are provided too.

8.2. Data Binding

As the following code snippets illustrated, there are three `FillPage` overloaded methods defined in `PageHelper`. Which gives you different manners to bind a data object to the target controls in the given container control. There are two methods can address your most data binding requirements in variety of scenarios.

```
public partial class PageHelper
{
    public virtual void FillPage(Object entity, BindingMappingCollection mappings);
    public virtual void FillPage(Object entity, Control container, string mappingPrefix = "");
    //Others
}
```

8.2.1. Data Binding Based on Given `BindingMappingCollection`

As you know, data binding is complete built on the basis of data mapping. We can manually build the data mapping set, and then deliver it to `FormHelper` for conducting customized data mapping. This kind of data binding makes it possible to fully control the binding behavior. E.G.You can manually ignore a particular control in the give container control if it should not be bound. You can also manually specify a column specific format if it needs to be different from the configuration. In this scenario, you can invoke the `FillPage` method with a `BindingMappingCollection` parameter.

`PageHelper` gives you am easy-to-use manner to build a `BindingMappingCollection` based on the given data source object and container control. You just need to invoke the following `BuildMapping` methods. Once this method is invoked, `PageHelper` will build a `BindingMapping` collection based on the given `DataTable`'s column set or entity type's data properties and container control's sub controls, including nested sub-controls.

```
public class PageHelper
{
    public BindingMappingCollection BuildMappings(Object entity, Control container, string
mappingPrefix = "")
    //Others
}
```

The method builds the `BindingMapping` according to the `NamingConverter` specific default mapping rule. If you do not familiar with `NamingConverter`, please refer to the chapter of Data Mapping. The optional *mappingPrefix* parameter is used in such a scenario that you need to

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

bind a single column to more than one control. For example, there are one `Label` and one `TextBox`, which are both bound to the *USER_NAME* column. According to the default mapping rule, the target control's name must be *UserName*. However, there are two target controls, which cannot share the same name. In this situation, you need to add an appropriate prefix to name the controls.

If the ID of the Label and the `TextBox` are UserName, and *txtUserName*, you can use the following two different programming to separately bind the data source (*USER_NAME* column) to them.

```
//1. UserName Label
var mapping1 = helper.BuildMapping(dataTable, this);
helper.FillPage(dataTable.Rows[0], mapping);

//2. txtUserName TextBox
var mapping2 = helper.BuildMapping(dataTable, this, "txt");
helper.FillPage(dataTable.Rows[0], mapping);
```

8.2.2. Data Binding Based on Automatically Built BindingMappingCollection

You can directly invoke follow two `FillPage` methods with container parameter to bind a given data source object to all of matched sub controls in the given container control. Before binding, the container specific data mapping will be automatically built, which ultimately serve the final data mapping.

```
public class FormHelper
{
    public virtual void FillForm(Object entity, Control container, string mappingPrefix = "")
    //Others
}
```

8.2.3. Control the Data Binding Behavior by Registering Event

Once the `FillPage` method is invoked, a `DataBinding` event will be raised when every control is attempt bound. At other hand, the `DataBound` event will be raised after data binding is completed. You can register such two events to change the default data binding behavior. The following code snippet represents the `DataBinding` and `DataBound` event definition.

```
public class PageHelper
{
    public event EventHandler<DataBindingEventArgs> DataBinding;
    public event EventHandler<DataBindingEventArgs> DataBound;
    //Others
}
```

This event specific `EventArgs` type is `NCS.IConnect.Helpers.Web.DataBindingEventArgs`. The following code snippet illustrates its definition.

```
public sealed class DataBindingEventArgs : EventArgs
{
    public Control        BindingControl {get;}❶
    public object         BindingValue {get; set;}❷
    public BindingMapping Mapping {get;}❸
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public Object         Entity {get;}❹
}
```

❶ The control to which the data source is bound;

❷ The value which is bound to the target control;

❸ The currently used `BindingMapping`;

❹ The data source object.

Except for the `BindingValue`, the rest properties are all read-only. If the value to bind need to be customized, you can register this event and set an appropriate value by set the BindingValue property. Following code shows how to bind data into a `CheckBox` control. The `DataTable` includes a `DataColumn` named IsExpired. The value of this column can be Yor N. From the configuration, the value will be bound to Checked property. This property only accepts Boolean value.

```
PageHelper helper = new FPageHelper();
helper.DataBinding += new DataBindingEventHandler(DataMapping_DataMappingBinding);

void DataMapping_DataMappingBinding(object sender, DataBindingEventArgs e)
{
  if (sender is CheckBox)
     {
        // BindingValue means the value will be assigned to control
        e.BindingValue = e.BindingValue.ToString () == "Y”? true: false;
      }
 }
```

8.2.4. DataGrid/GridView based Data Binding

The aforementioned data binding scenarios are all single record specific, `PageHelper` provides you with the following four methods to bind a collection to a `DataGrid` and `GridView`. As the following code snippet illustrates, the data source can be an `IEnumerable<Object>` or `DataView`.

```
public partial class PageHelper
{
    public  void FillDataGrid(IEnumerable<Object> datasource, DataGrid dataGrid);
    public  void FillDataGrid(DataView dataView, DataGrid dataGrid);
    public  void FillGridView(IEnumerable<Object> datasource, GridView gridView);
    public  void FillGridView(DataView dataView, GridView gridView);
    //Others
}
```

There are two pairs of pre-data-binding and post-data-binding events for DataGrid/GridView based data binding: DataGridBinding/DataGridBound, DataGridItemBinding/DataGridItemBound, GridViewBinding/GridViewBound, GridViewRowBinding/GridViewRowBound. The DataGridBinding/DataGridBound and GridViewBinding/GridViewBound are specific to the whole DataGird/GridView, and the reset are specific to `GridItem` or `GridViewRow`.

```
public partial class PageHelper
{
    public event EventHandler<DataGridBindingEventArgs> DataGridBinding;
    public event EventHandler<DataGridBindingEventArgs> DataGridBound;
    public event EventHandler<GridItemBindingEventArgs> DataGridItemBinding;
    public event EventHandler<GridItemBindingEventArgs> DataGridItemBound;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public event EventHandler<GridViewBindingEventArgs> GridViewBinding;
    public event EventHandler<GridViewBindingEventArgs> GridViewBound;
    public event EventHandler<GridViewRowBindingEventArgs> GridViewRowBinding;
    public event EventHandler<GridViewRowBindingEventArgs> GridViewRowBound;
    //Others
}
```

You can register appropriate events to change the default data binding behavior. The following code snippet represents the definition of the four EventArgs: `DataGridBindingEventArgs`, `GridItemBindingEventArgs`, `GridViewBindingEventArgs` and `GridViewRowBindingEventArgs`.

```
public sealed class DataGridBindingEventArgs : EventArgs
{
    public DataGrid DataGrid { get;}
    public Object DataSource { get;}
}

public sealed class GridViewBindingEventArgs : EventArgs
{
    public GridView GridView { get;}
    public Object DataSource { get;}
}

public sealed class GridItemBindingEventArgs : EventArgs
{
    public DataGridItem Item { get;}
    public Object Entity { get;}
    public BindingMappingCollection Mapping { get;}
}

public sealed class GridViewRowBindingEventArgs : EventArgs
{
    public GridViewRow GridViewRow { get;}
    public Object Entity { get;}
    public BindingMappingCollection Mapping { get;}
}
```

8.3. Data Capture

Generally, user will click a button to submit the data changes made on UI. Developers will write some codes to capture the values from controls and assign these values into business entity one by one. If the naming conventions meet the requirement, you can use `UpdateData` method to commit the changes into business entity by batch, instead one by one.

```
public class PageHelper
{
    public virtual void UpdateData(Object entity, BindingMappingCollection mappings);
    public virtual void UpdateData(Object entity, Control container, string mappingPrefix = "");
    //Others
}
```

Data capture, which also depends on data mapping, is the opposite operation of data binding. Data binding leverages the data mapping to get the target control based on `DataTable`'s column or entity's property, whereas data capture uses the data mapping to get the`DataTable`'s column or entity's property based on the control's ID. From the definition of the three `UpdateData` methods of `PageHelper` represented in the above code snippet, you can find they are similar to the definition of `Fill`Page methods.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

You can perform data capture based on the pre-built `BindingMappingCollection` by invoking the first overloaded method. You can also invoke the other method to capture the data based on the automatically built data mapping, even prefix based data mapping.

Similar to data binding, there is one pair of pre-data-dapture and post-data-capture events: `DataUpdating` and `DataUpdated`. As the following code snippet illustrates, they two have the same declaration as the aforementioned `DataBinding` and `DataBound` events.

```
public class PageHelper
{
    public event EventHandler<DataBindingEventArgs> DataUpdating;
    public event EventHandler<DataBindingEventArgs> DataUpdated;
    //Others
}
```

8.4. Enabling/Disabling Controls

Beside the two frequently used functionalities, data binding and data capture, it is also a common programming to enable or disable a batch of controls, which are usually located in a given container control. You can invoke the following three `SetEnabled` method to achieve such functionality.

```
public class PageHelper
{
    public void SetEnabled(WebControl container, bool enabled);
    //Others
}
```

8.5. Adding Hyperlinks for Validation Messages

Validator controls are widely used for input validation, and the error messages is displayed in the `ValidationSummary` control. By default, each message entry displayed in `ValidationSummary` is just only a literal string. The following `AddErrorMessageLink` method make you can change the literal message entries to linked ones, and the focus will automatically be set to the validated control when the linked message is clicked.

```
public class PageHelper
{
    public void AddErrorMessageLink(Page page);
    public void AddErrorMessageLink(Page page, string cssName);
    //Others
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 9. Database Helper

9.1. Introduction

The Database helper is a wrapper around the functionality of Microsoft Enterprise Library's Data Access Application Block. It wraps commonly methods from Data Access Application Block, and provides additional functions, especially when working with DataSet and stored procedures:

- Control database connection
- Handle transaction
- Create `DbCommand`
- Assign parameter values
- Check concurrency through version number
- Maintain the auditing purpose system fields
- Update the table by order through table relation
- Use naming conventions to map the `DataTable` in DataSet to CRUD stored procedures and map the parameters to `DataColumn`
- Provide a code generator so that it can use strongly typed classes access stored procedures and SQL statements
- Provide to set timeout property of `DbCommand` in configuration file
- Fulfill quick copy data to the destination table in the database

9.2. Loading Data into DataTable

Following code shows the simplest way to load data into `DataTable` with stored procedure and parameters.

```
DbHelper helper = new DbHelper();
helper.Fill(dataset.Tables[0], storedProcedure, value1, value2, ...);
```

Or you can take another way to fill data into `DataTable` using the parameters as `IDictionary<string, object>`.

```
DbHelper helper = new DbHelper();
IDictionary<string, object> parameters = ...;
helper.Fill(dataset.Tables[0], storedProcedure, parameters);
```

If the stored procedure contains output parameters, you should use another overload.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
DbCommand command = helper.BuildDbCommand(storedProcedure);
helper.AssignParameterValues(command, value1, value2, ...);
helper.Fill(dataset.Tables[0], command);

// Get the output parameter values
object result = command.Patameters[“output”].Value;
```

Note

Here, "void `AssignParameterValues` (`DbCommand` *`dbCommand`*, params object[] *`parameterValues`*);" is used to assign the parameters to the stored procedure. A point of worthy of note is that the parameters have to be provided in order. To avoid such a matter, we can pick up the overload "void `AssignParameterValues` (DbCommand command, `IDictionary<string, object>` parameters)" in place of that. For such an approach, we need to set the stored procedure parameter list as `IDictionary<string, object>`, but we don't need to care the order of parameter assignment.

9.3. Loading Data into DataSet

Loading data into DataSet means loading data into multiple tables in a batch. Generally, these `DataTable` will use different stored procedures to get data from database. And it will use the table name to navigate the relative stored procedure to fill `DataTable`. And it will use the instance of `CommandList` that will store table name as key and the relative stored procedure as value to fill DataSet.

```
DbHelper helper = new DbHelper();

Dictionary<string, string> storedProcedures = new Dictionary<string, string>();
storedProcedures.Add(dataset.Tables[0].TableName, storedProcedure0);
storedProcedures.Add(dataset.Tables[1].TableName, storedProcedure1);

CommandList commands = helper.BuildFillCommandList(dataset, storedProcedures);
BbCommand command = commands[dataset.Tables[0].TableName].SelectCommand;
helper.AssignParameterValues(command, value1, value2, ...);

command = commands[dataset.Tables[1].TableName].SelectCommand;
helper.AssignParameterValues(command, value1, value2, ...);

helper.Fill(ds, commands);
```

Note

The instance *`storedProcedures`* must use the table name in the dataset as the key, and the stored procedure name should be the value.

Actually, you can load data into `DataTable` one by one. For example,

```
DbHelper helper = new DbHelper();
helper.Fill(dataset.T_ORDER,"P_ORDER_GET",orderId,...);
helper.Fill(dataset.T_ORDER_LINE,"P_ORDER_LINE_GET",orderId,...);
...
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Another way is to use a single stored procedure, but this stored procedure will return multiple results.

```
DbHelper helper = new DbHelper();
string tables[] = new string[] {“T_ORDER”, “T_ORDER_LINE”};
helper.Fill(dataset, tables, "P_ORDER_GET", productName);

IF EXISTS (SELECT *
           FROM   dbo.sysobjects
           WHERE  id = OBJECT_ID(N'P_ORDER_GET')
                  AND OBJECTPROPERTY(ID,N'IsProcedure') = 1)
  DROP PROCEDURE P_ORDER_GET

GO

CREATE PROCEDURE P_ORDER_GET
 @p_product_name VARCHAR(256)

AS

BEGIN
  SELECT o.[ORDER_ID]
      ,o.[ORDER_DATE]
      ,o.[ORDER_TOTAL_PRICE]
      ,o.[ORDER_STATUS]
      ,o.[ORDERED_BY]
      ,o.[VERSION_NO]
      ,o.[TRANSACTION_ID]
      ,o.[CREATED_BY]
      ,o.[CREATED_TIME]
      ,o.[LAST_UPDATED_BY]
      ,o.[LAST_UPDATED_TIME]
  FROM T_ORDER o INNER JOIN T_ORDER_LINE ol ON ol.[ORDER_ID]=o.[ORDER_ID]
                 INNER JOIN T_PRODUCT p ON p.[PRODUCT_ID]=ol.[PRODUCT_ID]
  WHERE p.PRODUCT_NAME LIKE @p_product_name

  SELECT o.[ORDER_ID]
      ,o.[LINE_NO]
      ,o.[LINE_UNIT_PRICE]
      ,o.[LINE_QTY]
      ,o.[PRODUCT_ID]
      ,o.[VERSION_NO]
      ,o.[TRANSACTION_ID]
      ,o.[CREATED_BY]
      ,o.[CREATED_TIME]
      ,o.[LAST_UPDATED_BY]
      ,o.[LAST_UPDATED_TIME]
  FROM T_ORDER_LINE o INNER JOIN T_PRODUCT p ON p.[PRODUCT_ID]=o.[PRODUCT_ID]
  WHERE p.PRODUCT_NAME LIKE @p_product_name

END
```

Of course, for above sample, we can also make use of another `Fill` overload method to load data into multiple `DataTable` in DataSet, the result will be the same. Obviously, the difference is that we assigned stored procedure parameters as `IDictionary<string, object>` instead of object[]. The benefit of that is that we could assign the parameters out of order when the stored procedure needs many parameters, it's implemented better.

```
DbHelper helper = new DbHelper();
string tables[] = new string[] {“T_ORDER”, “T_ORDER_LINE”};
IDictionary<string, object> parameters = ...;
helper.Fill(dataset, tables, "P_ORDER_GET", parameters);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

9.4. Updating Data into Database

A `DataTable` may include added, modified or deleted records. The `Update` method will commit these changes into database. It will use the row state and the naming conversions defined in the configuration to find the relative stored procedures instead of writing many codes to define the relations between the stored procedure and DataSet. For example, using `DataDbAdapter` to update DataSet to database will be:

```
private void Save()
{
    SqlDataAdapter ad = new SqlDataAdapter();
    PrepareCommands(ad,conn);

    //Save to database
    ad.Update(dt);
}

private void PrepareCommands(SqlDataAdapter ad,SqlConnection conn)
{
    SqlCommand cmd = new SqlCommand();
    cmd.Connection = conn;
    cmd.CommandText = @"P_ORDER_I";
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.Parameters.Add("@p", SqlDbType.VarChar, 50, "ORDER_ID");
    cmd.Parameters.Add("@p_order_date", SqlDbType.DateTime, 50, "ORDER_DATE");
    ...
    ad.InsertCommand = cmd;

    ad.UpdateCommand = new SqlCommand();
    ad.UpdateCommand.Connection = conn;
    ad.UpdateCommand.CommandText =@"P_ORDER_U";
    ad.UpdateCommand.Parameters.Add("@o_order_id", SqlDbType.VarChar, 64, "ORDER_ID");
    ad.UpdateCommand.Parameters["@o_order_id"].SourceVersion = DataRowVersion.Original;
    ad.UpdateCommand.Parameters.Add("@p_order_date", SqlDbType.DateTime, 50, "ORDER_DATE");
    ...

    ad.DeleteCommand = new SqlCommand();
    ad.DeleteCommand.Connection = conn;
    ad.DeleteCommand.CommandText = "P_ORDER_D";
    ad.DeleteCommand.Parameters.Add("@o_order_id", SqlDbType.VarChar, 50, "ORDER_ID");
    ad.UpdateCommand.Parameters["@o_order_id"].SourceVersion = DataRowVersion.Original;
    ad.DeleteCommand.Parameters.Add("@o_version_no", SqlDbType.Int, 9, "VERSION_NO");
    ...
}
```

From the code, we can see, if the table has many fields, we have to code many lines. But it just is one table, if we have to do these for many tables, it will be tedious. But now, we can simply use the component to update DataSet like this:

```
DbHelper helper = new DbHelper();
// update the dataset directly
helper.Update(dataset);
```

And define the naming conversions once in the configuration file:

```
<configuration>
  <configSections>
    <section name="iConnect.dataaccess"
            type="NCS.IConnect.Helpers.Data.Configuration.DataAccessSettings,
                  NCS.IConnect.Helpers.Data"/>
    <section name="dataConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
                   Microsoft.Practices.EnterpriseLibrary.Data" />
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    <section name="iConnect.applicationcontexts"
             type="NCS.IConnect.ApplicationContexts.Configuration.ApplicationContextData,
                   NCS.IConnect.ApplicationContexts" />
  </configSections>
  <dataConfiguration defaultDatabase="..." />
  <connectionStrings>

  <iConnect.applicationcontexts defaultContextLocator="callContextLocator">
    <contextLocators>
      <add name="callContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.CallContextLocator,
                 NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>

 <iConnect.dataaccess commandTimeout="00:00:30" >
    <systemFields>
      <add name="VersionNo" fieldName="VERSION_NO" />
      <add name="TransactionId" fieldName="TRANSACTION_ID" />
      <add name="CreatedBy" fieldName="CREATED_BY" />
      <add name="CreatedTime" fieldName="CREATED_TIME" />
      <add name="LastUpdatedBy" fieldName="LAST_UPDATED_BY" />
      <add name="LastUpdatedTime" fieldName="LAST_UPDATED_TIME" />
    </systemFields>
    <commands>
      <add name="Select" format="P_{TableName}_S" />
      <add name="Delete" format="P_{TableName}_D" />
      <add name="Update" format="P_{TableName}_U" />
      <add name="Insert" format="P_{TableName}_I" />
    </commands>
    <parameters>
      <add name="Current" format="p_{DataColumnName}" />
      <add name="Original" format="o_{DataColumnName}" />
    </parameters>
    <schemaMappings>
      <add name="dataTableName" format="T_{TableName}" />
      <add name="dataColumnName" format="{DataColumnName}" />
    </schemaMappings>
  </iconnect.dataaccess>

</configuration>
```

Note

The component depends on the components of `ApplicationContext`, Transaction and Enterprise Library Data. So we have to add these references and place their configuration here if they have configuration settings. Please refer to the relative chapters and the document of Enterprise Library for more details.

These configuration items are designed to provide a rule to map the `DataTable` to CRUD stored procedures, and map the parameters with the `DataColumn`. By using these naming conventions, you will be released from writing tedious data access code.

- `schemaMappings`: Defines the format of table and fields.

  By default, the component *assumes* the name of `DataTable` in DataSet is same as the table name in database.

```
<schemaMappings>
 <add name="dataTableName" format="T_{TableName}" />
 <add name="dataColumnName" format="{DataColumnName}" />
</schemaMappings>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Here, the name of `DataTable` is prefixed with T_, and the format of `DataColumn` is not special, nether prefix nor suffix.

If project standard is that table name is prefixed with TB_, just modify *dataTableName = "T_{TableName}"* to *dataTableName = "TB_{TableName}"*, or no prefix required, *dataTableName = "{TableName}"*.

It will be used into configuration item `commands`. For example, in DataSet, there is a `DataTable` named "T_ORDER", then *`{TableName}`* will be "ORDER" and the relative stored procedures' name will be P_ORDER_S, P_ORDER_D, P_ORDER_U, P_ORDER_I.

Note

{TableName} and *{DataColumnName}* are predefined placeholder, you can not change them.

- `commands`: Defines the format of CRUD stored procedures.

  From the schemaMappings section, we can see `commands` will define the rule of mapping table name to the relative stored procedures.

```
<commands>
   <add name="Select" format="P_{TableName}_S" />
   <add name="Delete" format="P_{TableName}_D" />
   <add name="Update" format="P_{TableName}_U" />
   <add name="Insert" format="P_{TableName}_I" />
</commands>
```

  Here, the format is all stored procedures are prefixed with P_, and suffix with _I, _S, _U and _D to indicate the CRUD stored procedures. These stored procedures will be used for `DbDataAdapter` when calling `Fill` and `Update` method.

  But it just is general rule and it will work on the methods that does not point out the stored procedures, like `Update(DataSet)`, `Update(DataTable)`. If there are some tables and their stored procedures which do not match the rule, we will have to directly call Enterprise Library's `Database` instance to update.

```
//Prepare commands...
...
helper.Database.UpdateDataSet(dataSet,"table
name",insertCommand,updateCommand,deleteCommand,UpdateBehavior.Standard);
```

- commandTimeout: Defines the command timeout when excuting the command.

- `parameters`

  Defines the format of parameters of stored procedure. `Current` means the current version of `DataRow`, `Original` means the original version of `DataRow`.

```
<parameters>
 <add name="Current" format="p_{DataColumnName}" />
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
 <add name="Original" format="o_{DataColumnName}" />
</parameters>
```

As shown above, the parameters are normally prefixed with p_, and for those need to get the value of original version from DataSet are prefixed with o_. Generally, for primary key and version number, we should always get the original value. For example, if the input parameter is for primary key cust_id, it should be named o_cust_id, the original value will be passed in when performing updating.

Warning

- These two items must be different, otherwise system can't generate the correct `CommandList`.
- `Original`'s format can *NOT* be equal to `{DataColumnName}`; but `Current`'s can.

- `systemFields`: Defines the name of System Fields done for concurrency control and audit trail.

```
<systemFields>
 <add name="VersionNo" FieldName="VERSION_NO" />
 <add name="TransactionId" FieldName="TRANSACTION_ID" />
 <add name="CreatedBy" FieldName="CREATED_BY" />
 <add name="CreatedTime" FieldName="CREATED_TIME" />
 <add name="LastUpdatedBy" FieldName="LAST_UPDATED_BY" />
 <add name="LastUpdatedTime" FieldName="LAST_UPDATED_TIME" />
</systemFields>
```

iConnect.NET predefines 6 system fields, and each table should contain these 6 fields.

- VERSION_NO: Used to check the version when updating and deleting. It will be very useful to control optimistic concurrency;
- TRANSACTION_ID: Used to indicate the related records in a transaction. Audit purpose;
- CREATED_BY: User who created the record. Audit purpose;
- CREATED_TIME: Date and time the record created. Audit purpose;
- LAST_UPDATED_BY: Last user who updated the record. Audit purpose;
- LAST_UPDATED_TIME: Date and time the record last updated. Audit purpose.

Tip

In most of the situations, you will not need to modify this setting. But if your tables do not contain these fields, you can remove the items do not exist from here. The helper class will ignore them. It also can update the DataSet with a correct order of tables.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
// gererate a CommandList, and customize it, then update.
CommandList commands = helper.BuildUpdateCommandList(dataset);
Commands["T_ORDER"].UpdateCommand.Parameters.Add("p_additional",...);
helper.Update(dataset, commands);
```

We can use the iConnect Studio->SQL Generator tool to generate the draft stored procedures.

9.5. ExecuteReader

`ExecuteReader` will execute the command and return the result as `IDataReader` by the query.

```
DbHelper helper = new DbHelper();

// Execute the command
IDataReader result = helper.ExecuteReader(command);
```

Except for above basic method, there are other two overload to handle the query and return the result as `IDataReader`, just as the following methods:

```
// Execute the stored procedure and corresponding parameters, the parameters should be set in turn.
IDataReader ExecuteReader(string procedureName, params object[] parameters);❶

// Execute the stored procedure and corresponding parameters.
IDataReader ExecuteReader(string procedureName, IDictionary<string, object> parameters);❷
```

❶ It's used to retrieve the query result as `IDataReader` through the stored procedure name and corresponding parameters, especially note that the parameters have to be set in turn, otherwise, it'll result in the exception or unexpected result.

❷ The overload is similar with above one, the difference with that is about setting stored procedure parameters, that is, it could be not set in turn.

9.6. ExecuteScalar

`ExecuteScalar` will execute the command and returns the first column of the first row in the result set returned by the query. Extra columns or rows are ignored.

```
DbHelper helper = new DbHelper();

// Execute the stored procedure and parameters
object result = helper.ExecuteScalar(storedProcedureName, value1, value2, ...);
```

Another overload will accept a DbCommand object, you can use `BuildDbCommand` method to create a DbCommand object, and assign the parameter values.

```
// Get a DbCommand for stored procedure
DbCommand command = helper.BuildDbCommand(storedProcedure);
helper.AssignParameterValues(command, value1, value2, ...);

// Execute the command
object result = helper.ExecuteScalar(command);

// Get the parameter values
result = command.Patameters["output"].Value;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Another overload will work with Generics. It'll avoid casting for the results.

```
int i = helper.ExecuteScalar<int>(storedProcedureName, value1, value2, ...);
```

For all of above methods, there are corresponding overload to support setting stored procedure parameters out of order.

```
// The overload is related to "object ExecuteScalar(string storedProcedureName, params object[]
parameterValues);".
object ExecuteScalar(string procedureName, IDictionary<string, object> parameters);

// The overload is related to "T ExecuteScalar<T>(string storedProcedureName, params object[]
parameterValues);".
T ExecuteScalar<T>(string procedureName, IDictionary<string, object> parameters);
```

9.7. ExecuteNonQuery

`ExecuteNonQuery` will execute the command and returns the number of rows affected.

```
DbHelper helper = new DbHelper();

// Execute a stored procedure with parameters
int count = helper.ExecuteNonQuery(storedProcedureName, value1, value2, ...);

// Get a DbCommand for stored procedure
DbCommand command = helper.BuildDbCommand(storedProcedure);
helper.AssignParameterValues(command, value1, value2, ...);

// Execute this command
int count = helper.ExecuteNonQuery(command);

// Get the parameter values
object result = command.Patameters[“output”].Value;
```

The overload of `ExecuteNonQuery` support setting stored parameters out of order.

```
DbHelper helper = new DbHelper();

// Execute a stored procedure with out-of-order parameters
IDictionary<string, object> parameters = ...;
int count = helper.ExecuteNonQuery(storedProcedureName, parameters);

// Get a DbCommand for stored procedure
DbCommand command = helper.BuildDbCommand(storedProcedure);
IDictionary<string, object> parameters = ...;
helper.AssignParameterValues(command, parameters);

// Execute this command
int count = helper.ExecuteNonQuery(command);

// Get the parameter values
object result = command.Patameters[“output”].Value;
```

9.8. ExecuteBulkCopy

Copies all rows in the supplied System.Data.DataTable to a destination table specified by the System.Data.SqlClient.SqlBulkCopy.DestinationTableName property of the System.Data.SqlClient.SqlBulkCopy object..

```
DbHelper helper = new DbHelper();
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
// Execute a stored procedure with out-of-order parameters
helper.ExecuteBulkCopy(dataTable, destinationTable,commandTimeOut, columnMapping,bulkCopyOption);
```

9.9. Working with Events

Here are several events exposed by DbHelper.

- DataAdapterCreated
- ConnectionCreated
- CommandListCreated
- DataUpdating
- DataUpdated

These events provide the capability to customize the helper class behavior.

9.10. Procedure Generation

1. Install T4 toolbox, if you have installed the iConnect.NET, you can find it in the "All Programs\NCS iConnect.NET 4.3\Tools".

2. Open VS2008/2010, add a new item, the template is "Text Template". The following code shows a typical procedure generation by using T4 engine, then save this template, the t4 engine will generated the code automatically.

```
<#@ template  language="C#" hostSpecific="true" debug="true" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>❶
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Common.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Helpers.Data.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.SqlGeneration.dll"
#>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version
no}bin\Microsoft.Practices.EnterpriseLibrary.Common.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version
no}\bin\Microsoft.Practices.EnterpriseLibrary.Caching.dll" #>
<#@ import namespace="NCS.IConnect.SqlGeneration" #>❷
<#@ include file="T4Toolbox.tt" #>❸
<#
   new NCS.IConnect.SqlGeneration.SqlServer.ProcedureGenerator(this.Host.ResolvePath("App.config"),❹
   "iConnectDB"❺, "dbo"❻, "T_PRODUCT", "T_ORDER","T_ORDER_LINE"❼).Run();❽
   new
NCS.IConnect.SqlGeneration.Oracle.ProcedureGenerator(this.Host.ResolvePath("App.config"),"iConnectOracle","ic4gy"❾
   , "T_IC_BATCH_JOB_PARAM", "T_IC_MESSAGES").Run();
#>
```

❶ The full path of assembly references.

❷ The import namespaces.

❸ Template is defined in T4Toolbox.tt and referenced by including T4Toolbox.tt.

❹ The configuration file's name.

❺ The connect string's name.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❻ The destination DB schema.

❼ The table names which want to generate the audit table.

❽ Using the template generation to generate the code.

❾ The oracle DB owner.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 10. Entity Framework

10.1. Introduction

Entity Framework provides a strongly-typed LINQ data access experience over relational databases. It enables developers to work with data in the form of domain-specific objects and properties, without having to concern themselves with the underlying database tables and columns where this data is stored. Leveraging the designer provided in Visual Studio, we can easily create an entity data model based on the database table. With the help of code generation mechanism, the entity specific classes can be automatically generated. By default, each automatically generated entity class can by itself keep the state (Unchanged, Added, Modified and Deleted). Such a state tracking finally help determine what kind of CUD (Create, Update and Delete) should be performed when the `DataContext` is called to submit the data change into database.

However, the built-in state tracking cannot work in distributed application as the state cannot be serialized. In addition, there is a limitation if we adopt this programing mode that we has to use generated entity classes. It is better if we can use the exiting "plain-old" CLR objects (POCO) as the entity. As the matter of factor, POCO is supported by Entity Framework, which maps entities class to the entity data model. In addition, Microsoft provides a "ADO.NET Self-Tracking Entity Generator" to generate POCO based entity classes and related classes to achieve self or manual state tracking.

What does iConnect.NET do for you? In fact, what are provided are following two code generators, which are both integrated with Visual Studio

- Entity Data Model Transformation Generator: This code generator transform the source .edmx model, directly generated by designer, to new .edmx model with the stored procedure based C/S mapping. This process is called Entity Data Model transformation;
- Extended *POCO Self-Tracking* Generator: This code generator generate POCO based entity classes and `DataContext` class based on the .edmx model. This process is called POCO Self-Tracking classes generation.

10.2. Entity Data Model Transformation

By default, Microsoft Entity Framework dynamically generates SQL based on the entity data model and state, and leverages ADO.NET to execute this SQL to submit the changed data into target database. However, we have to use standard stored procedures instead of automatically generated literal SQL for some reasons. E.G. iConnect.NET Audit Trail depends on the standard CUD stored procedures to fire the audit triggers, otherwise the deletion specific audit logging cannot work. You can manually conduct entity specific stored procedure mapping using entity data model designer. However, this is a boring and error-pone tasks. Entity Data Model Transformation Generator help to relief from such a boring and error-pone tasks.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Using the entity data model designer, you first design your .edmx model by directly selecting related database tables. Once this work completed, you just need to right click in the designer and select "Add Code Generation Item" in the pop-up context menu.

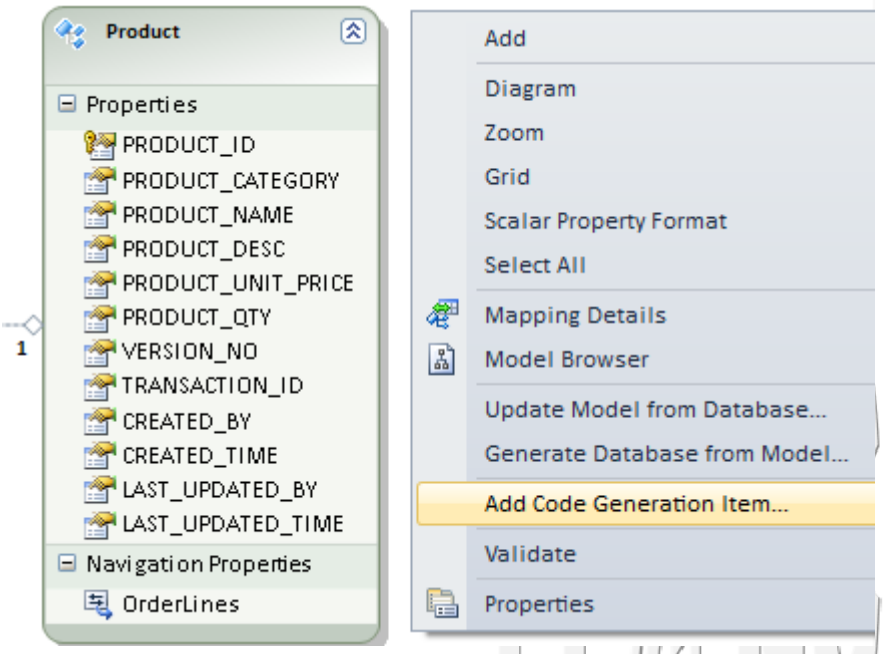


Figure 10.1. Add Code Generation Item

Then you can see two new item templates, "iConnect.NET ADO.NET POCO Self-Tracking Entity Generator" and "iConnect.NET Entity Model Transformation Generator". You should select the second one.

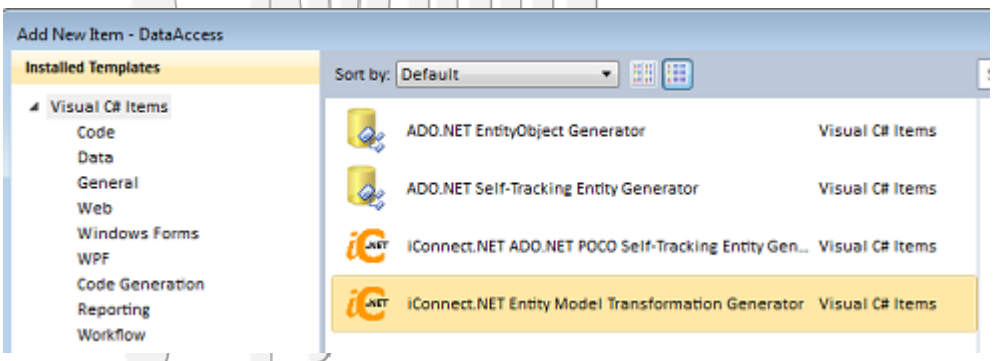


Figure 10.2. Code Generator Item Templates in Entity Framework ToolBox

The the following "Model Transformation Item" form is shown for customizing the mapping rule, and default rule is specified.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

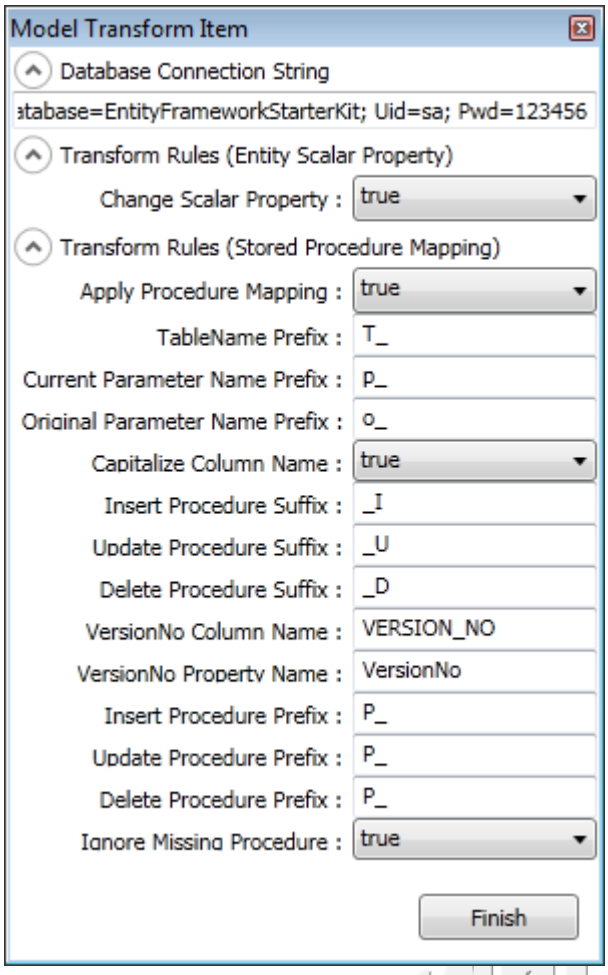


Figure 10.3. Model Transform Item Wizard

Following is the description for each rules:

- Database Connection String: The connection string to connect the target database;
- Change Scalar Property: Whether to transform entity property names. When we use Visual Studio "ADO.NET Entity Data Model" to generate a entity model, the property name of entity type is default the same with database column name, so if we want to transform property name "USER_NAME" to "UserName", just set this option to "true";
- Apply Procedure Mapping: This option is used to decide whether to import specified database stored procedures to entity model as Functions. Other options of "Transform Rules (Stored Procedure Mapping)" will be disabled if this option set to "false";
- Table Name Prefix : Table name prefix in database;
- Current Parameter Name Prefix: For update Function, there is a `Checkbox` option in "Mapping Details" view for a property, to indicate whether to "Use Original Value". Here we specify the parameter name prefix for stored procedure's parameters, if matched, will use current value of property to pass back to stored procedure, that is the same to uncheck "Use Original Value" in "Mapping Details" view.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Original Parameter Name Prefix: Specify the original parameter naming conversion for stored procedure, if matched, will use original value of the property to pass back to stored procedure, that is the same to check "Use Original Value" in "Mapping Details" view.
- Capitalize Column Name: Whether to capitalize column name;
- Insert Procedure Suffix: Specify "Insert Stored Procedure" suffix;
- Update Procedure Suffix: Specify "Update Stored Procedure" suffix;
- Delete Procedure Suffix: Specify "Delete Stored Procedure" suffix;
- VersionNo Column Name : Version Number column name in database table;
- VersionNo Property Name : Version Number property of entity;
- Insert Procedure Prefix: Specify "Insert Stored Procedure" prefix;
- Update Procedure Prefix: Specify "Update Stored Procedure" prefix;
- Delete Procedure Prefix: Specify "Delete Stored Procedure" prefix;
- Ignore Missing Procedure : Whether to ignore missing procedure when import "Insert/Update/Delete" Functions for the entity, if set to "false", will throw exception by T4 Code Generation Engine when encounter missing procedures.

After all of the mapping rule is appropriately specified and then clicking Finish button, a T4 template file will be generated to transform the source data entity model to a new model following the mapping rules you set. As the following screen shot illustrates, the original entity data model (*ShopModel.edmx*) is created in the DataAccess project. After we add the "iConnect.NET Entity Model Transformation Generator" code generation item based on the original model following the above steps, the *ShopModelTransform.tt* is generated, which generates a new model (*ShopModelTransform.edmx*).

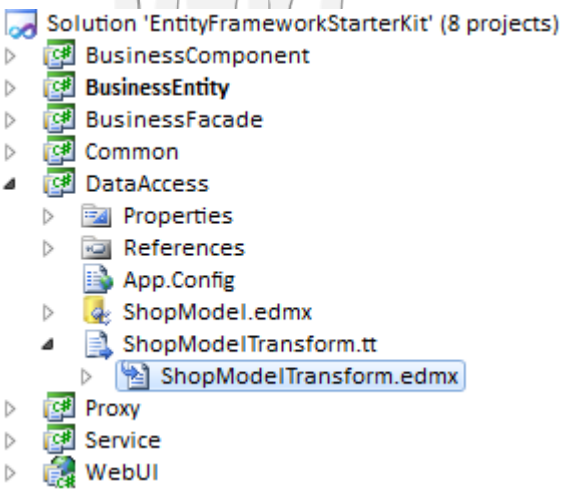


Figure 10.4. iConnect.NET Entity Model Transformation Generator Output

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

10.3. ADO.NET POCO Self-Tracking Entity Generation

After the new .edmx model is generated by Entity Data Model Transformation Generator, we can leverage another code generator, Extended ADO.NET POCO Self-Tracking Entity Generator to generate all entity classes and `DataContext` class against the new .edmx model. The generated classes have the following features:

- The six system fields, CreatedBy, CreatedTime, LastUpdatedBy, LastUpdatedTime, Version and Transaction can be automatically evaluated;
- iConnect.NET Data Mapping component can build mapping against the entity type and use FormHelper/PageHelper to perform batch data binding and data capture;

Open entity data model designer, and right click in the designer and select "*Add Code Generation Item*" in the pop-up context menu. In the "Add New Item" windows, select the "iConnect.NET ADO.NET POCO Self-Tracking Entity Generator" item. The following "POCO Self-Tracking Generator" dialog will be shown. As the follow figure illustrates, you can select a project to which the generated classes should be placed. This option just changes the namespace of the generated classes, and will really add the generated class files into the project you specify. You need to manually remove the generated files to the target project or add a file link in the target project.

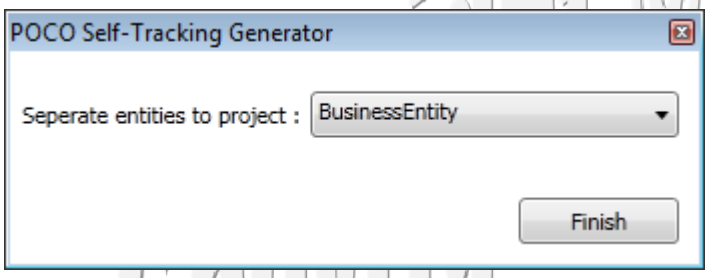


Figure 10.5. POCO Self-Tracking Generator Wizard

Two T4 template files will be generated after you click the Finish button. One of the two is used for entity classes generation, and the other is used for `DataContext` class generation. As the following screen shot illustrates, after we add "iConnect.NET ADO.NET POCO Self-Tracking Entity Generator" code generation item, the two T4 template files, ShopModel.Entities.tt and ShopModel.Context.tt, are generated. ShopModel.Entities.tt will generate all entity specific classes, each of which is defined a single .cs file (Order.cs, OrderLine.cs and Product.cs). The ShopModel.Context.tt generates the ShopModel.Context.cs file, including the definition of `DataContext` class.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

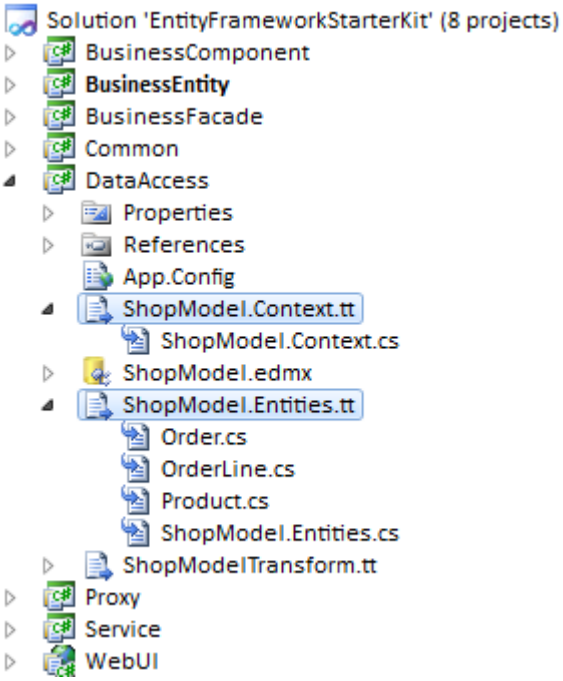


Figure 10.6. POCO Self-Tracking Entity Output

As we have specify the project that we hope to separate the entities to, the entities generated here have changed its namespace to default namespace of the selected project in the wizard. We can manually add file link in "Business Entity" to make sure the entities are built to a separate assembly.

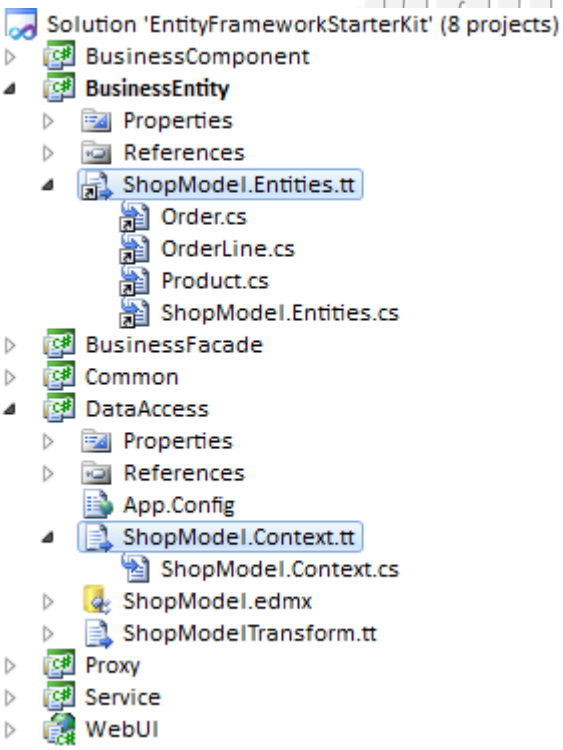


Figure 10.7. Manually Separate the *.Entities.tt to Business Entity Project

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

So far all of the related classes have been generated. You can directly use the entity types which are capable of state tracking carry business entities. The following is demonstrative programming to retrieve data. The `ShopContext` is generated `ObjectContext` type.

```
public List<Product>
    GetProductsByIds(List<String> listProductId) { using (var context =
    new ShopContext()) { var query = from p in context.Products where
    listProductId.Contains(p.ProductId) select p; return query.ToList(); }
    }
```

If you need to submit the changed entities (Added, Modified and Deleted) into database, you should follow the following programming.

```
public void UpdateProduct(Product product) { using (var
    context = new ShopModelContainer()) {
    context.Products.ApplyChanges(product); context.SaveChanges(); }
    }
```

For more information about ADO.NET Entity Framework programming, please refer to *MSDN* online documentation.

Note

The original entity data model is created, a connection string will be added in the configuration. Because the metadata files in this connection string is based on the original model, when we deploy, we must revise the connection string to use the transformed model based metadata files.

```
<add name="ShopModelContainer"
      providerName="System.Data.EntityClient"

connectionString="metadata=res://*/ShopModelTransform.csdl|res://*/ShopModelTransform.ssdl|res://*/ShopModelTransform.msl;
      provider=System.Data.SqlClient;provider connection string='Data
      Source=...;Initial Catalog=EntityFrameworkStarterKit;User
      ID=sa;Password=123456'" />
```

10.4. DbContext POCO Entity Generation

Version 4.1 of the Entity Framework contains both the Code First approach and the new DbContext API. This API provides a more productive surface for working with the Entity Framework and can be used with the Code First, Database First, and Model First approaches.

Similar to "ADO.NET POCO Self-Tracking Entity Generator" in previous section, iConnect.NET also provide a "DbContext POCO Entity Generator" to generate the DbContext based Context class and POCO-Self Tracking entities.

Open entity data model designer, and right click in the designer and select "*Add Code Generation Item*" in the pop-up context menu. In the "Add New Item" windows, select the "iConnect.NET DbContext POCO Entity Generator" item. The following "DbContext POCO Generator" dialog will be shown. As the follow figure illustrates, you can select a project to which the generated classes should be placed. This option just changes the namespace of the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

generated classes, and will really add the generated class files into the project you specify. You need to manually remove the generated files to the target project or add a file link in the target project.

Figure 10.8. DbContext POCO Generator Wizard

The generated files are simalar to previous section, you can refer to the previous section and process the files according to your project. If you want to use the generated DbContext based class with Repository Pattern, you can continue to refer to the next section.

10.5. Repository Pattern

It is recommended to adopt repository pattern to perform Entity Framework based data access. There are two abstract classes, `Repository<TEntity, TDbContext>`and `Repository<TEntity,TDbContext>`, which are used as the base classes of the concrete repository type. As illustrated in the following code snippet, the generic parameter of `RepositoryBase<TDbContext>` is subclass of `DbConext`. `Repository<TEntity, TDbContext>`, which is entity specific repository, inherits `Repository<TEntity, TDbContext>`, and the generic parameter `TEntity` represents the entity type.

```
public abstract class RepositoryBase<TDbContext> :
    IDisposable where TDbContext: DbContext { //Others public
    RepositoryBase(TDbContext dbContext, bool lazyLoadingEnabled = false);
    public TDbContext DbContext { get; private set; } } public abstract class
    Repository<TEntity, TDbContext> : RepositoryBase<TDbContext>
    where TEntity: class where TDbContext: DbContext { // Others public
    Repository(TDbContext dbContext); }
```

The two entity modes, *CLR POCO* and *POCO Self-Tracking* are supported by the two base repository types. `CLR POCO` entities and `DbContext` based classes can be generated by the built-in code generator (T4), and the `POCO Self-Tracking` entities based classes is generated in the way mentioned in the previous section. Because we can get the `DbContext` via the `DbContext` property, all kinds of data access operations can be performed by using it. In order to make it easy to perform the basic CURD operations, the following methods are defined in `Repository<TEntity, TDbContext>`and `Repository<TEntity,TDbContext>`.

```
public abstract class RepositoryBase<TDbContext> :
    IDisposable where TDbContext: DbContext { //Add, Update, Delete & Find
    public TEntity Add<TEntity>(TEntity entity) where TEntity: class;❶ public TEntity
Update<TEntity>(TEntity entity) where
    TEntity: class;❷ public void Delete<TEntity>(TEntity
    entity) where TEntity: class; ❸ public TEntity
    Find<TEntity>(object[] keyValues) where TEntity: class;❹ //Get entities based on given filter
condition. public
    IEnumerable<TEntity>
    Get<TEntity>(Expression<Func<TEntity, bool>>
    predicate❺) where TEntity: class; //Get entities based on
    given filter condition with include path. public
    IEnumerable<TEntity> Get<TEntity, TIncludeProperty>(
    Expression<Func<TEntity, bool>> predicate,
    Expression<Func<TEntity, TIncludeProperty>> includePath❻) where TEntity: class; public
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
IEnumerable<TEntity>
    Get<TEntity, TIncluePropertyl, TIncluePropertyl>(
    Expression<Func<TEntity, bool>> predicate,
    Expression<Func<TEntity, TIncluePropertyl>> includePathl, ❼ Expression<Func<TEntity,
TIncluePropertyl>>
    includePathl❽) where TEntity: class; public
    IEnumerable<TEntity> Get<TEntity, TIncluePropertyl,
    TIncluePropertyl, TIncluePropertyl>( Expression<Func<TEntity,
    bool>> predicate, Expression<Func<TEntity,
    TIncluePropertyl>> includePathl, Expression<Func<TEntity,
    TIncluePropertyl>> includePathl, Expression<Func<TEntity,
    TIncluePropertyl>> includePathl❾) where TEntity:
    class; //Get entities based on given filter condition with include path
    (paging & sorting). public IEnumerable<TEntity> Get<TEntity,
    TSortProperty>( Expression<Func<TEntity, bool>> predicate,
    int pageSize, ❿ int pageIndex, ⓫
   Expression<Func<TEntity, TSortProperty>> sortKeySelector, ⓬ bool isAsc⓭) where TEntity: class;
public
    IEnumerable<TEntity> Get<TEntity, TSortProperty,
    TIncludeProperty>( Expression<Func<TEntity, bool>>
    predicate, int pageSize, int pageIndex, Expression<Func<TEntity,
    TSortProperty>> sortKeySelector, bool isAsc,
    Expression<Func<TEntity, TIncludeProperty>> includePath) where
    TEntity: class; public IEnumerable<TEntity> Get<TEntity,
    TSortProperty, TIncludeProperty1, TIncludeProperty2>(
    Expression<Func<TEntity, bool>> predicate, int pageSize, int
    pageIndex, Expression<Func<TEntity, TSortProperty>>
    sortKeySelector, bool isAsc, Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty2>> includePath2) where TEntity: class; public
    IEnumerable<TEntity> Get<TEntity, TSortProperty,
    TIncludeProperty1, TIncludeProperty2, TIncludeProperty3>(
    Expression<Func<TEntity, bool>> predicate, int pageSize, int
    pageIndex, Expression<Func<TEntity, TSortProperty>>
    sortKeySelector, bool isAsc, Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty1>> includePath2, Expression<Func<TEntity,
    TIncludeProperty1>> includePath3) where TEntity: class; //Get All
    entities (paging & sorting) public IEnumerable<TEntity>
    GetAll<TEntity>() where TEntity: class; public
    IEnumerable<TEntity> GetAll<TEntity, TIncludeProperty>(
    Expression<Func<TEntity, TIncludeProperty>> includePath) where
    TEntity: class; public IEnumerable<TEntity> GetAll<TEntity,
    TIncludeProperty1, TIncludeProperty2>( Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty2>> includePath2) where TEntity: class; public
    IEnumerable<TEntity> GetAll<TEntity, TIncludeProperty1,
    TIncludeProperty2, TIncludeProperty3>( Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty2>> includePath2, Expression<Func<TEntity,
    TIncludeProperty3>> includePath3) where TEntity: class;
    }
```

❶ Add a new entity.

❷ Update the existing entity.

❸ Delete the existing entity.

❹ Find the entity based on the given key values.

❺ The expression representing the filter condition.

❻ The expression representing the include path.

❼ The expression representing the first include path.

❽ The expression representing the second include path.

❾ The expression representing the third include path.

❿ The record count for each page.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

11 The index of the retrieved page.

12 The expression representing accessor of soring key.

13 A boolean value indicating whether to sort the record in ascend order.

Note

The Add and Delete methods are only support CLR POCO mode. For CLR POCO Self-Tracking mode, Update method can be used for CUD operations since the entity is stateful.

Similar methods are also defined in `Repository<TEntity,TDbContext>`. Because `Repository<TEntity,TDbContext>` is specific to a particular concrete entity, the methods' generic parameter `TEntity` is omitted.

```
public abstract class Repository<TEntity,
    TDbContext> : RepositoryBase<TDbContext> where TEntity: class
    where TDbContext: DbContext { // Methods public TEntity Find(object[]
    keyValues); public IEnumerable<TEntity>
    Get(Expression<Func<TEntity, bool>> predicate); public
    IEnumerable<TEntity> Get<TIncludeProperty>(
    Expression<Func<TEntity, bool>> predicate,
    Expression<Func<TEntity, TIncludeProperty>> includePath);
    public IEnumerable<TEntity> Get<TIncludeProperty1,
    TIncludeProperty2>( Expression<Func<TEntity, bool>>
    predicate, Expression<Func<TEntity, TIncludeProperty1>>
    includePath1, Expression<Func<TEntity, TIncludeProperty2>>
    includePath2); public IEnumerable<TEntity> Get<TIncludeProperty1,
    TIncludeProperty2, TIncludeProperty3>(Expression<Func<TEntity,
    bool>> predicate, Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty2>> includePath2, Expression<Func<TEntity,
    TIncludeProperty3>> includePath3); public IEnumerable<TEntity>
    Get<TSortProperty>( Expression<Func<TEntity, bool>>
    predicate, int pageSize, int pageIndex, Expression<Func<TEntity,
    TSortProperty>> sortKeySelector, bool isAsc); public
    IEnumerable<TEntity> Get<TSortProperty, TIncludeProperty>(
    Expression<Func<TEntity, bool>> predicate, int pageSize, int
    pageIndex, Expression<Func<TEntity, TSortProperty>>
    sortKeySelector, bool isAsc, Expression<Func<TEntity,
    TIncludeProperty>> includePath); public IEnumerable<TEntity>
    Get<TSortProperty, TIncludeProperty1, TIncludeProperty2>(
    Expression<Func<TEntity, bool>> predicate, int pageSize, int
    pageIndex, Expression<Func<TEntity, TSortProperty>>
    sortKeySelector, bool isAsc, Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty2>> includePath2); public IEnumerable<TEntity>
    Get<TSortProperty, TIncludeProperty1, TIncludeProperty2,
    TIncludeProperty3>(Expression<Func<TEntity, bool>>
    predicate, int pageSize, int pageIndex, Expression<Func<TEntity,
    TSortProperty>> sortKeySelector, bool isAsc,
    Expression<Func<TEntity, TIncludeProperty1>> includePath1,
    Expression<Func<TEntity, TIncludeProperty1>> includePath2,
    Expression<Func<TEntity, TIncludeProperty1>> includePath3);
    public IEnumerable<TEntity> GetAll(); public
    IEnumerable<TEntity>
    GetAll<TIncludeProperty>(Expression<Func<TEntity,
    TIncludeProperty>> includePath); public IEnumerable<TEntity>
    GetAll<TIncludeProperty1, TIncludeProperty2>(
    Expression<Func<TEntity, TIncludeProperty1>> includePath1,
    Expression<Func<TEntity, TIncludeProperty2>> includePath2);
    public IEnumerable<TEntity> GetAll<TIncludeProperty1,
    TIncludeProperty2, TIncludeProperty3>( Expression<Func<TEntity,
    TIncludeProperty1>> includePath1, Expression<Func<TEntity,
    TIncludeProperty2>> includePath2, Expression<Func<TEntity,
    TIncludeProperty3>> includePath3); }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Warning

The status, which determines the operation (Insert, Update, and Delete) to perform, can be kept in the *POCO Self -Tracking* entity itself, so only the `Update` method is available if we adopt *POCO Self -Tracking* mode. As matter of fact, the `NotImplementedException` exception will be thrown when passing a *POCO Self -Tracking* entity to the `Add` and `Delete` method.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 11. Transaction Management

11.1. Introduction

A transaction is a sequence of operations performed as a single logical unit of work. A logical unit of work must exhibit four properties, called the atomicity, consistency, isolation, and durability (ACID) properties, to be qualified as a transaction.

- *Atomicity*: A transaction must be an atomic unit of work; either all of its data modifications are performed, or none of them is performed;
- *Consistency*: When completed, a transaction must leave all data in a consistent state. In a relational database, all rules must be applied to the transaction's modifications to maintain all data integrity. All internal data structures, such as B-tree indexes or doubly-linked lists, must be correct at the end of the transaction;
- *Isolation*: Modifications made by concurrent transactions must be isolated from the modifications made by any other concurrent transactions. A transaction either recognizes data in the state it was in before another concurrent transaction modified it, or it recognizes the data after the second transaction has completed, but it does not recognize an intermediate state. This is referred to as serializability because it results in the ability to reload the starting data and replay a series of transactions to end up with the data in the same state it was in after the original transactions were performed;
- *Durability*: After a transaction has completed, its effects are permanently in place in the system. The modifications persist even in the event of a system failure.

In most scenarios, the transaction we use is database transaction. A database transaction comprises a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. Transactions in a database environment have two main purposes:

- To provide reliable units of work that allow correct recovery from failures and keep a database consistent even in cases of system failure, when execution stops (completely or partially) and many operations upon a database remain uncompleted, with unclear status;
- To provide isolation between programs accessing a database concurrently. Without isolation the program's outcomes are possibly erroneous.

In terms of ADO.NET, the `System.Data.Common.DbTransaction` can be used to enlist more than one database access operations into one single transaction. `DbTransaction` can only be used in the local transaction scenario, and all of the operation enlisted in one transaction must be based one single database connection. If you transaction involves more than one operations which access to more than one databases, event multiple servers, the local transaction will be promoted to distributed transaction. There is a `TransactionScope` type, which is defined in

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

the `System.Transactions` namespace, make it easy to program distributed transaction. As with the usage of `DbTransaction` and `TransactionScope`, you can refer to MSDN online documentation.

In order to make it easy to program transaction in iConnect.NET based N-Layer application, the following work is made:

- Create a `NCS.IConnect.Transactions.Transaction` type, which wrap around the ADO.NET `DbTransaction`. This type it possible to program transaction in a `TransactionScope` like manner;
- Create two call handlers, NCS.IConnect.Transactions.Transaction specific `TransactionCallHandler` and `TransactionScope` specific `TransactionScopeCallHandler`. Both `TransactionCallHandler` and `TransactionScopeCallHandler` are defined in the NCS.IConnect.Transactions.PolicyInjection namespace. With the help of the two call handlers, we can adopt AOP mode to program transaction.

11.2. Getting Started

This section will provide you with a demonstration to guide how to program transaction using `Transaction` type.

- *Step I: Create a New Database, Table and Stored Procedures*

  First, we create a new database named iConnectDb in your local SQL Server, and then execute the following SQL script to create a simple Users table with only two columns: UserId and UserName.

```
CREATE DATABASE iConnectDb
GO
USE iConnectDb
GO
CREATE TABLE [Users]
(
  [UserId]   VARCHAR(50)   PRIMARY KEY,
  [UserName] NVARCHAR(128) NOT NULL
)
GO
```

  Execute the following SQL script to create two stored procedures. The `AddUser` is used to insert a new user record, and `UserExists` is used to check if the user record of the specified ID exists.

```
CREATE PROCEDURE [AddUser]
 @userId   VARCHAR (50),
 @userName NVARCHAR (128)
AS
BEGIN
 INSERT Users ( UserId, UserName)
 VALUES(@userId,@userName);
END


CREATE PROCEDURE [UserExists]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
 @userId   VARCHAR (50)
AS
BEGIN
 IF EXISTS (SELECT 1 FROM [Users] WHERE UserId = @userId)
  SELECT 1;
 ELSE
  SELECT 0;
END

GO
```

- *Step II: Create a New Application and Configure `DbHelper`*

  Create a new console project, and add the assembly reference to the `NCS.IConnect.Helpers.Data`, `Microsoft.Practices.EnterpriseLibrary.Logging.Database` and `NCS.IConnect.Transactions`. Then add an App.config, and copy the following configuration to this added configuration file. You need to revise the connection string based on your real security setting. You need to revise the connection string accordingly.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="dataConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
                   Microsoft.Practices.EnterpriseLibrary.Data"/>
  </configSections>
  <dataConfiguration defaultDatabase="iConnectDb"/>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient"/>
  </connectionStrings>
</configuration>
```

  Copy the following code into the Program.cs. As the code snippet describes, two static methods are created, and the `CreateUser` is used to insert a new user record into the Users table and `UserExists` method check if the user record of the specified user ID exists. Both of the two method use iConnect.NET `DbHelper` to execute the stored procedures.

```
class Program
{
    static DbHelper DbHelper = new DbHelper();
    static void Main(string[] args)
    {
        string userId = Guid.NewGuid().ToString();
        string userName = "Foo";
        CreateUser(userId, userName);
       Console.WriteLine("Is the user successfully created? {0}.", UserExists(userId) ? "Yes" :
"No");
    }
    static void CreateUser(string userId, string userName)
    {
        DbHelper.ExecuteNonQuery("AddUser", userId, userName);
    }
    static bool UserExists(string userId)
    {
        return DbHelper.ExecuteScalar<int>("UserExists", userId) == 1;
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Run this program, you will get the following output, which indicates the two static methods work normally.

```
Is the user successfully created? Yes.
```

Now copy the following code snippet to override your `Main` method. In the new `Main` method, the `CreateUser` method with the same parameter is called twice.

```
static void Main(string[] args)
{
    string userId = Guid.NewGuid().ToString();
    string userName = "Foo";
    try
    {
        CreateUser(userId, userName);
        CreateUser(userId, userName);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
   Console.WriteLine("Is the user successfully created? {0}.", UserExists(userId) ? "Yes" : "No");
}
```

Because the UserId is the primary key of the Users table, a `SqlException` will be thrown when executing the `CreateUser` for the second time. However, the successful invocation to the first `CreateUser` method will insert a new user record, because the two times of user creations are not enlisted in a single transaction. Run this program again, the following output is displayed.

```
Violation of PRIMARY KEY constraint 'PK__Users__1788CC4C7F60ED59'. Cannot insert
 duplicate key in object 'dbo.Users'.
The statement has been terminated.
Is the user successfully created? Yes.
```

You can use the above mentioned `Transaction` type to make the two invocations of the `CreateUser` are performed in one single transaction. The programing is like the code snippet below:

```
static void Main(string[] args)
{
    string userId = Guid.NewGuid().ToString();
    string userName = "Foo";
    try
    {
        using (Transaction transaction = new Transaction())
        {
            CreateUser(userId, userName);
            CreateUser(userId, userName);
            transaction.Complete();
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
   Console.WriteLine("Is the user successfully created? {0}.", UserExists(userId) ? "Yes" : "No");
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Now, the two `CreateUser` methods will be executed in a single transaction, and exception will lead to roll back the whole transaction. In this situation, no user record can be inserted into *Users* table, and this can be proved by the program's output.

```
Violation of PRIMARY KEY constraint 'PK__Users__1788CC4C7F60ED59'. Cannot insert
 duplicate key in object 'dbo.Users'.
The statement has been terminated.
Is the user successfully created? No.
```

The above sample demonstrate the basic usage of Transaction, and we will give more detailed information about this type. In the iConnect.NET based N-Layers architecture, it is recommended to program transaction in an AOP manner. Two PIAB call handlers, `TransactionCallHandler` and `TransactionScopeCallHandler`, achieve an easy-to-use programming mode to create transactional method. WCF provides transaction support, and the transaction programming in WCF will be introduced in this section.

11.3. Transaction

As above mentioned, iConnect.NET `Transaction` is a wrapper around `DbTransaction`, so as to support `TransactionScope` like programming. There are a series of overloaded constructors defined, which are used to construct a Transaction instance in different scenario. All of the four constructor's are listed in the following code snippet.

```
[Serializable]
public class Transaction : IDisposable
{
    //Other Members
    public Transaction();❶
    public Transaction(IsolationLevel isolationLevel❷);
    public Transaction(string databaseName❸);
    public Transaction(string databaseName, IsolationLevel isolationLevel);
}
```

❶ When the default constructor is invoked, the default database specific Transaction object is instantiated. The default database is configured in the *<dataConfiguration/>* configuration section, which also exists in the configuration of the above demonstration;

❷ An instance of System.Data.IsolationLevel type, which indicates the isolation level of the transaction. The default value is *IsolationLevel.ReadCommitted*;

❸ The name of configured connection string.

The following short code snippet demonstrates a typical usage of Transaction. If you have used `TransactionScope`, you will notice it is the same as `TransactionScope's` usage. The Transaction should be instantiated in a using block. All of the transactional operations are directly placed in the using block, followed the invocation of `Transaction's` Complete method, which will commit the transaction.

```
using (Transaction trans = new Transaction())
{
    //Transactional Operations...
    trans.Complete();
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Once the Transaction is instantiated, a `DbTransaction` will be created and stored in the current thread's call context as an ambient transaction. You can get such a cached `DbTransaction` object by invoking the followed two static overloaded `GetDbTransaction` methods. The first method return the default database specific `DbTransaction` if exists, and the `DbTransaction` returned by the second is specific to the given connection string name.

```
public class Transaction : IDisposable
{
    //Other Members
    public static DbTransaction GetDbTransaction();
    public static DbTransaction GetDbTransaction(string databaseName);
}
```

11.4. `TransactionCallHandler` & `TransactionScopeCallHandler`

In the iConnect.NET based N-Layer architecture, is is recommended to adopt AOP programming mode to manage transaction. Since the AOP framework we use is EntLib PIAB, we create a Transaction specific call handler: `TransactionCallHandler`. In addition, the built-in `TransactionScope` is also a good alternative to control transaction, particularly distributed transaction. The corresponding `TransactionScopeCallHandler` is also provided.

AOP based transaction adopts declaration programing. We just need to apply the two call handler specific custom attribute to the given transactional method. Once decorated with such custom attributes, the whole method can be guaranteed to be executed in singe transaction.

The following code snippet represents the definition of `TransactionCallHandler` specific attribute: `TransactionCallHandlerAttribute`, which is defined in the `NCS.IConnect.Transactions.PolicyInjection` namespace. You can specify the connection string name, otherwise the default connection string will be used. The default isolation level is *IsolationLevel.ReadCommitted*, which can be overridden by specifying the `IsolcationLevel` property.

```
[AttributeUsage(AttributeTargets.Method)]
public class TransactionCallHandlerAttribute : HandlerAttribute
{
    public string DatabaseName { get; set; }
    public IsolationLevel IsolationLevel { get; set; }
    public TransactionCallHandlerAttribute()
    //Other...
}
```

In the following code snippet, the *OrderProduct* method is converted to a transaction method by applying the `TransactionCallHandlerAttribute` to it. Once executed, the whole method will be executed in a single transaction, whose isolation level is `IsolationLevel.Serializable`.

```
public class OrderService: IOrderService
{
   [TransactionCallHandler(IsolationLevel = IsolationLevel.Serializable)]
   public void OrderProduct(Order order)
   {
       //Other...
   }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`TransactionScopeCallHandler` has also its own attribute: `TransactionScopeCallHandlerAttribute`, whose definition is illustrated in the following code snippet below. Most of the properties correspond to the ones defined in `TransactionScope`. If you do not familiar with them, please refer to MSDN online documentation.

```
[AttributeUsage(AttributeTargets.Method)]
public class TransactionScopeCallHandlerAttribute : HandlerAttribute
{
    public bool Complete { get; set; }
    public TransactionScopeOption TransactionScopeOption { get; set; }
    public IsolationLevel IsolationLevel { get; set; }
    public EnterpriseServicesInteropOption InteropOption { get; set; }
    public string Timeout{get;set;}

    public TimeSpan TimeoutSpan { get; }
    public TransactionOptions TransactionOptions{get;}

    public TransactionScopeCallHandlerAttribute()
    public TransactionScopeCallHandlerAttribute(TransactionScopeOption scopeOption)
    public TransactionScopeCallHandlerAttribute(IsolationLevel isolationLevel)
    public TransactionScopeCallHandlerAttribute(TransactionScopeOption scopeOption, IsolationLevel
isolationLevel)
}
```

11.5. Transaction Programming in WCF

WCF itself provides a complete transaction support, regardless of local transaction or distributed. Not only can you define a transactional service operation, you can also guarantee more than one service invocations are enlisted into one single transaction.

In fact, it is recommended to directly adopt WCF based transaction programming mode if you application is a WCF based distributed application. If you do not familiar with how to programming transactional service in WCF, please refer to the MSDN online documentation.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 12. Exception Handling

12.1. Introduction

When developing any applications, no matter how carefully we write our code, errors are hard to avoid. It's necessary to have an flexible exception handling framework to handle the caught exception. The `EnterLib Exception Handling Application Block (EHAB for short)` enables developers and policy makers to create a consistent exception handling policy, which can be freely changed after deploying. *EHAB* achieves the following goals:

- Encapsulate the logic used to perform the most common exception handling tasks into minimal application code;
- Relieve developers of the requirement to write duplicate code and custom code for common exception handling tasks;
- Allow exception handling policies to be dynamically revised after they have been deployed;.
- Incorporate best practices for exception handling;
- Provide the consistent guideline and practice when working with database.

For *EHAB*, there are two main concepts that we need to notice: `Exception Handler` and `Exception Policy`.

12.1.1. Exception Handler

As its name suggests, exception handler is used to handle the caught exception. Generally speaking, each exception handler owns the sole responsibility, E.G. Logging exception handler is just write the exception related information into the log storage, Error reporting exception handler only need to report the error information in a given manner. A set of exception handlers, which make up an handler pipeline, can be used to handle one single exception.

EHAB provides three kinds of built-in exception handlers:

- `Wrap handler`: This exception handler create a new exception to wrap the original exception, which becomes the inner exception of the new exception;
- `Replace handler`: This exception handler create a new exception to replace the original exception;
- `Logging handler`: This exception handler formats exception related information , such as message, stack trace, etc, and then write them into the given log storage;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Fault Contract exception handler: This exception handler is designed for use at Windows® Communication Foundation (WCF) service boundaries, and generates a new Fault Contract from the exception.

Besides the built-in exception handlers provided by *EHAB*, we can freely create custom handlers. iConnect.NET exception handling can be considered as a group of custom exception handlers.

12.1.2. Exception Policy

In terms of *EHAB*, each exception policy answers two exception handling based questions as below:

- Which exception handler(s) is (are) used to handle the caught exception?
- How to further handle the exception which is produced by the exception handler pipeline?

Exception handling policy is complete defined in configuration. For the first one, one or more exception handlers are configured to handle the given type of exception, so exception policy is complete based on the exception type. For the second one, in a configurable manner, the exception handled by handler pipeline can be ignored or re-thrown.

12.2. Getting Started

This section provides you with a Windows Forms application based sample, which demonstrates you how to leverage the EHAB to show the formated error message in a MessageBox. The custom `NCS.IConnect.ExceptionHandling.MessageBoxExceptionHandler` is used in this demonstration.

Step I: Create a Windows Forms Application and Configure Exception Handling

Create an Windows Forms project, and add the following two assembly references to `Microsoft.Practices.EnterpriseLibrary.ExceptionHandling` and `NCS.IConnect.ExceptionHandling`. Then add the App.config, and copy the following configuration into this configuration file.

```
<configuration>
  <configSections>
    <section name="exceptionHandling"

type="Microsoft.Practices.EnterpriseLibrary.ExceptionHandling.Configuration.ExceptionHandlingSettings,
                   Microsoft.Practices.EnterpriseLibrary.ExceptionHandling" />
  </configSections>
  <exceptionHandling>
    <exceptionPolicies>
      <add name="UI Policy">
        <exceptionTypes>
          <add name="ArgumentException"
               type="System.ArgumentException,
                     mscorlib"
               postHandlingAction="None" >
            <exceptionHandlers>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
              <add name="MessageBox Handler"
                   type="NCS.IConnect.ExceptionHandling.MessageBoxExceptionHandler,
                         NCS.IConnect.ExceptionHandling"
                   formatterType="NCS.IConnect.ExceptionHandling.TextExceptionFormatter,
                                  NCS.IConnect.ExceptionHandling"
                   caption="Error"
                   includeInnerException="true"
                   template="Please kindly Take Note:
Message:{message}
Type:{type}
StackTrace:{stackTrace}
Source:{source}
DateTime:{dateTime}
HelpLink:{helpLink}.
                 " />
            </exceptionHandlers>
          </add>
        </exceptionTypes>
      </add>
    </exceptionPolicies>
  </exceptionHandling>
</configuration>
```

Step II: Programming Exception Handling

In automatically created Form1, drag and drop one button, and write the following click event handler code.

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        throw new ArgumentException("MessageDemo");
    }
    catch (Exception ex)
    {
        if(ExceptionPolicy.HandleException(ex, "UI Policy");
        {
            throw;
        }
    }
}
```

Run the sample and click the button, An `ArgumentException` will be caught and the formatted error message will be shown int a MessageBox as below.

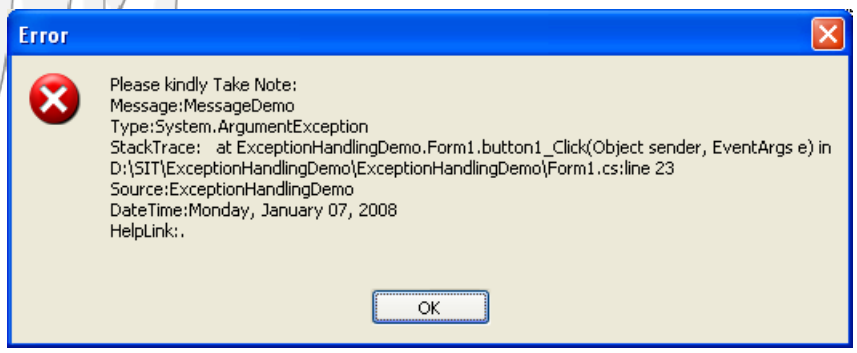


Figure 12.1. The error message box

iConnect.NET provides you with a full set of custom exception handlers, which can be used in most of exception handling scenarios. Error reporting handlers help you to report the formated

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

error message in variety to representation forms; Logging handler write the formated error information into different type of log storage; Wrap/Repace handlers, which extend the built-in `WrapHandler` and `ReplaceHandler`, make it possible to use iConnect.NET Messaging to define exception message; We have also database based wrap/pepace handlers, which is particularly used to handle the `SqlExceoption`/`OracleException`. Filterable exception revolves the limitation of EHAB inherent type-based exception handling mode.

12.3. Error Reporting

Among the iConnect.NET custom exception handlers, there are a series of error reporting based ones. They format the exception information and report to the end user in different representation forms. Some of these can be used in Windows Forms application, and the rest are designed for ASP.NET application. The `MessageBoxExceptionHandler` is a frequently-used error reporting handler, and we will introduce the others.

12.3.1. FormExceptionHandler

`FormExceptionHandler` is used in Windows application. It shows a customized Windows Form with formated exception message. It has a friendly message, and a button to show the exception details. Such an error from is like the figure below. By default, only a simple friendly error message is shown. If you press the *Show Detail* button, the hidden detail message will be displayed.

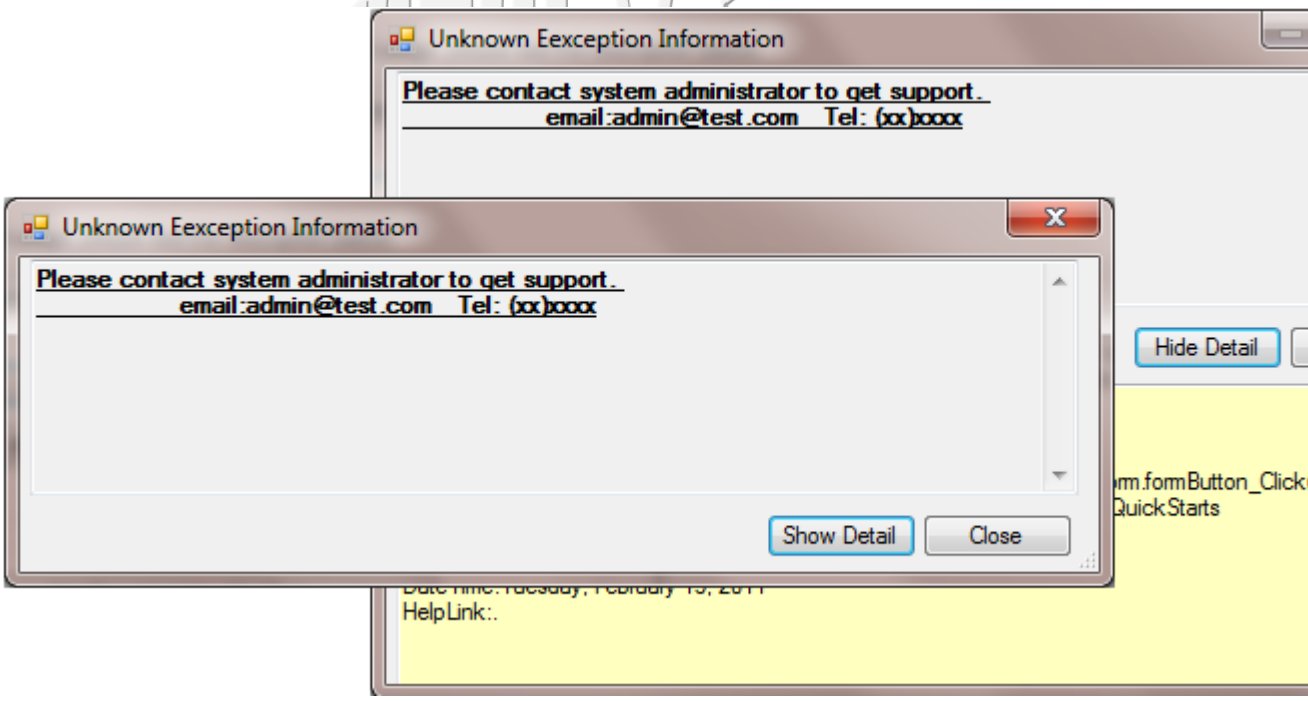


Figure 12.2. Error Windows Form (FormExceptionHandler)

The follow code snippet is `FormExceptionHandler` specific configuration.

```
<add name="Form Handler"
     type="NCS.IConnect.ExceptionHandling.FormExceptionHandler,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
          NCS.IConnect.ExceptionHandling"
     formatterType="NCS.IConnect.ExceptionHandling.TextExceptionFormatter,
                    NCS.IConnect.ExceptionHandling"
     friendlyMessage="Please contact system administrator to get support."❶
     caption="Unknown Eexception Information"
     includeInnerException="true"
     template="Please kindly take note:
Message: {message}
Type: {type}
StackTrace: {stackTrace}
SourceRegular: {source}
DateTime: {dateTime}
HelpLink: {helpLink}.”
     exceptionForm="NCS.IConnect.ExceptionHandling.QuickStart.CustomExceptionForm,
               NCS.IConnect.ExceptionHandling.QuickStart"❷
/>
```

❶ The *friendlyMessage* attribute defines a simple friendly message which is by default shown in the Windows form. The detail error message template is configured as *template* attribute, and above mentioned placeholder can be supported in the template;

❷ The exceptionForm specify the type of Windows form to show the error message. You can directly use the built-in `NCS.IConnect.ExceptionHandling.ExceptionForm`, or create a custom exception form derived from it.

12.3.2. MessageBoxExceptionHandler

`MessageBoxExceptionHandler` is used in Windows application. It shows the formated the exception message with windows's messagebox. The following code snippet demonstrates its configuration. For this handler, you only need to specify the template property. The special characters in final message will be escaped automatically.

```
<add name="MessageBox Handler"
     type="NCS.IConnect.ExceptionHandling.MessageBoxExceptionHandler,
          NCS.IConnect.ExceptionHandling"
     formatterType="NCS.IConnect.ExceptionHandling.TextExceptionFormatter,
                    NCS.IConnect.ExceptionHandling"
     caption="Error"
     includeInnerException="true"
     template="Please kindly take note:
Message:{message}
Type:{type}
StackTrace:{stackTrace}
Source:{source}
DateTime:{dateTime}
HelpLink:{helpLink}.
/>
```

12.3.3. AlertExceptionHandler

`AlertExceptionHandler` is used in ASP.NET application. It shows the formated error message as a javascript alter. The following code snippet demonstrates its configuration. For this handler, you only need to specify the template property. The special characters in final message will be escaped automatically.

```
<add name="Alert Handler"
     type="NCS.IConnect.ExceptionHandling.AlertExceptionHandler,
          NCS.IConnect.ExceptionHandling"
     formatterType="NCS.IConnect.ExceptionHandling.
                    NCS.IConnect.ExceptionHandling"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
     includeInnerException="false"
     template="Please kindly take note: {message}
/>
```

12.3.4. WebTextControlExceptionHandler

`WebTextControlExceptionHandler` is used in ASP.NET application. It' shows the formatted error message in the specified web control. The type of the specified web control to show error message must implement `ITextControl` interface, e.g, `Label`, `TextBox` control. `WebTextControlExceptionHandler` based configuration is as below.

```
<add name="TextControl Handler"
     type="NCS.IConnect.ExceptionHandling.WebTextControlExceptionHandler,
           NCS.IConnect.ExceptionHandling"
     formatterType="NCS.IConnect.ExceptionHandling.TextExceptionFormatter,
                    NCS.IConnect.ExceptionHandling"
     controlName="[Page].[Master].LabelMessage"❶
     includeInnerException="false"
     template="Please kindly take note: {message}
/>
```

❶ The controlName attribute specifies the control to display the formatted error message. By default, this handler will locate the control in the current page. If master page is used, you must specify whether the control is located in the master page or content page. For the control located in the master page, the controlName attribute should be configured as *[Page].[Master].ControlName*. If the control is placed in the content page, the configuration format will be like *[Page].[Master].ContentHolder.ControlName*.

12.4. Wrapping & Replacement

Exception wrapping and replacement are two commonly used exception handling policies, which can convert one category of caught exceptions to an exception of specified type for further handling. iConnect.NET wrap/replace exception handler directly extend the EHAB built-in `WrapHandler` and `ReplaceHandler`, and Messaging integration is enabled, which make it possible to directly leverage iConnect.NET Messaging to define the message of the exception to wrap or replace.

12.4.1. WrapExceptionHandler

WrapExceptionHandler is used to wrap one exception around the original one. Because WrapExceptionHandler is derived from EHAB `WrapHandler`, the configuration properties of `WrapHandler` are all inherited. For the built-in `WrapHandler`, you can choose two ways to define the message of the exception to wrap: literal and resource. *WrapExceptionHandler* gives you the third option: iConnect.NET Message. The following XML snippet gives you a demonstrative configuration of *WrapExceptionHandler.*

```
<add name="Wrap Exception Handler"
     type="NCS.IConnect.ExceptionHandling.WrapExceptionHandler,
           NCS.IConnect.ExceptionHandling"
      wrapExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                         NCS.IConnect.ExceptionHandling"❶
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
      exceptionMessageID="key001"❷
      parameters="account,full name"❸
      delimiter=","❹ />
```

❶ The qualified name (QN) of the type of the exception to wrap the original exception;

❷ The ID of the message entry to define the message of the exception to wrap;

❸ The parameter list to replace the placeholders in the template of the message entry;

❹ The delimiter to split each parameter value configured in the above *parameters* attribute. This is an optional configuration property, and the default value is *comma*.

As aforementioned, WrapExceptionHandler is derived from EHAB `WrapHandler`, so you can directly configure a literal string as the message of the exception to wrap like this:

```
<add name="Wrap Exception Handler"
     type="NCS.IConnect.ExceptionHandling.WrapExceptionHandler,
           NCS.IConnect.ExceptionHandling"
      wrapExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                         NCS.IConnect.ExceptionHandling"
      exceptionMessage="Error..."/>
```

Note

Assuming that under an extreme scenario user has configured above all(Of course, it's not recommended). The result will be certain because different approach has different priority, the order is: ExceptionMessageID, Resource, General Message.

12.4.2. ReplaceExceptionHandler

ReplaceExceptionHandler is used to create a new exception to replace the original one. Because *ReplaceExceptionHandler* is derived from EHAB `ReplaceHandler`, the configuration properties of `ReplaceHandler` are all inherited. For the built-in `ReplaceHandler`, you can choose two ways to define the message of the exception to replace: literal and resource. *ReplaceExceptionHandler* gives you the third option: iConnect.NET Message. The following XML snippet gives you a demonstrative configuration of *ReplaceExceptionHandler*. All of configuration properties are all similar to the ones of the above *WrapExceptionHandler*.

```
<add name="Wrap Exception Handler"
     type="NCS.IConnect.ExceptionHandling.WrapExceptionHandler,
           NCS.IConnect.ExceptionHandling"
     replaceExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                           NCS.IConnect.ExceptionHandling"
     exceptionMessageID="key001"
     parameters="account,full name"
     delimiter="," />
```

You can directly configure a literal string as the message of the exception to replace like this:

```
<add name="Wrap Exception Handler"
     type="NCS.IConnect.ExceptionHandling.WrapExceptionHandler,
           NCS.IConnect.ExceptionHandling"
     replaceExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                           NCS.IConnect.ExceptionHandling"
     exceptionMessage="Error..." />
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

You can also define the message in one resource file, and specify the resource type name and resource name. Resource based configuration is like this:

```
<add name="Wrap Exception Handler"
     type="NCS.IConnect.ExceptionHandling.WrapExceptionHandler,
           NCS.IConnect.ExceptionHandling"
     replaceExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                           NCS.IConnect.ExceptionHandling"
     exceptionMessageResourceType="NCS.IConnect.ExceptionHandling.UnitTests.Properties.Resources,
                                   NCS.IConnect.ExceptionHandling.UnitTests"
     exceptionMessageResourceName="ResourceExceptionMessage" />/>
```

12.5. Database Handlers

Regardless of SQL Server or Oracle, we can directly throw exception from stored procedure. For T-SQL, the system function RAISERROR enable us to throw an error with specified message, severity, etc. As with PL/SQL, there is a corresponding system function RAISE_APPLICATION_ERROR. If the ADO.NET code involving RAISERROR/RAISE_APPLICATION_ERROR, a SqlException/OracleException will be thrown once the execution path hit such a statement. In order to handle the `SqlException`/`OracleException` thrown in this situation, two set of wrap/replace handler are provided. *SqlWrapExceptionHandler / OralceWrapExceptionHandler* and *SqlReplaceExceptionHandler / OralceReplaceExceptionHandler*. As well as the above `WrapExceptionHandler` and `ReplaceExceptionHandler`, the four handlers also support iConnect.NET Messaging, which enable us can specify the ID of the message entry, other than the raw error message.

In the following code snippet, the ID of message entry (key001) and placeholder parameters (parameter1, parameter2) are specified. Once SqlException or OarcleException is thrown, the raw literal string (key001, Parameter1, Parameter2) will be become the message of the exception (for SqlException) or a part of exception message (for OarcleException). When such a SqlException/OracleException is delivered to `SqlWrapExceptionHandler`(`SqlReplaceExceptionHandler`)/`OralceWrapExceptionHandler`(`OralceReplaceExceptionHandler`), the handler can resolve the raw message and extract the message ID and parameter list, and then generate a complete message via iConnect.NET Messaging. The final message will become the message of the exception to wrap/replace.

```
SQL Server:
RAISERROR ('key001,Parameter1,Parameter2', 16, 1)

Oracle:
RAISE_APPLICATION_ERROR(-20000, 'key001,Parameter1,Parameter2');
```

12.5.1. *SqlWrapExceptionHandler & OralceWrapExceptionHandler*

The two handlers are specially designed to wrap the database based exceptions: `SqlException`/`OracleException`. Except for basic exception wrapping functionality, they can revolve the exception message with message ID and parameter(s). Once the message ID and parameters are successfully resolved, the complete message can be generated with the help of

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

iConnect.NET Messaging. Then final message will become the message of the exception to wrap. The following two code snippets demonstrate the two exception handlers' configuration.

- *SqlWrapExceptionHandler*

```
<exceptionTypes>
  <add type="System.Data.SqlClient.SqlException,
            System.Data"
       postHandlingAction="ThrowNewException"
       name="SqlException">
    <exceptionHandlers>
      <add name="SqlWrapExceptionHandler"
           type="NCS.IConnect.ExceptionHandling.Database.SqlWrapExceptionHandler,
                 NCS.IConnect.ExceptionHandling.Database"
           wrapExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                              NCS.IConnect.ExceptionHandling"❶
           delimiter=","❷/>
    </exceptionHandlers>
  </add>
</exceptionTypes>
```

- *OralceWrapExceptionHandler*

```
<exceptionTypes>
  <add type="System.Data.OracleClient.OracleException,
            System.Data.OracleClient"
       postHandlingAction="ThrowNewException"
       name="OracleException">
    <exceptionHandlers>
      <add name="OracleWrapExceptionHandler"
           type="NCS.IConnect.ExceptionHandling.Database.OralceWrapExceptionHandler,
                 NCS.IConnect.ExceptionHandling.Database"
           wrapExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                              NCS.IConnect.ExceptionHandling"❶
           delimiter=","❷/>
    </exceptionHandlers>
  </add>
</exceptionTypes>
```

❶ The qualified name (QN) of the type of the exception to wrap.

❷ The delimiter is to split parameters appearing in the exception message.

As well as above mentioned `WrapExceptionHandler`, the two exception handlers also support the following two ways to specify the message to the exception to wrap: *literal* and *resource*. Once failure of message id/parameter resolving, the two alternatives will take effect. The *literal*/*resource* based message configuration is similar to `WrapExceptionHandler`, and literal message own the lower priority than resource based message.

12.5.2. SqlReplaceExceptionHandler & OralceReplaceExceptionHandler

The two handlers are specially designed to replace the database based exceptions (`SqlException`/`OracleException`) with a new exception of given type. Except for basic exception replacement functionality, they can revolve the exception message with message ID and parameter(s). Once the message ID and parameters are successfully resolved, the complete message can be generated with the help of iConnect.NET Messaging. Then final

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

message will become the message of the exception to replace. The following two code snippets demonstrate the two exception handlers' configuration.

- SqlReplaceExceptionHandler

```
<exceptionTypes>
  <add type="System.Data.SqlClient.SqlException,
            System.Data"
       postHandlingAction="ThrowNewException"
       name="SqlException">
    <exceptionHandlers>
      <add name="SqlReplaceExceptionHandler"
           type="NCS.IConnect.ExceptionHandling.Database.SqlReplaceExceptionHandler,
                 NCS.IConnect.ExceptionHandling.Database"❶
           replaceExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                                 NCS.IConnect.ExceptionHandling"❷
           delimiter=","❸/>
    </exceptionHandlers>
  </add>
</exceptionTypes>
```

- OracleReplaceExceptionHandler

```
<exceptionTypes>
  <add type="System.Data.OracleClient.OracleException,
            System.Data.OracleClient"
       postHandlingAction="ThrowNewException"
       name="OracleException">
    <exceptionHandlers>
      <add name="OracleReplaceExceptionHandler"
           type="NCS.IConnect.ExceptionHandling.Database.OracleReplaceExceptionHandler,
                 NCS.IConnect.ExceptionHandling.Database"
           replaceExceptionType="NCS.IConnect.ExceptionHandling.DbBusinessException,
                                 NCS.IConnect.ExceptionHandling"❶
           delimiter=","❷/>
    </exceptionHandlers>
  </add>
</exceptionTypes>
```

The typical configuration of SqlReplaceExceptionHandler and OracleReplaceExceptionHandler is similar:

❶ The qualified name (QN) of the type of the exception to replace.

❷ The delimiter is to split parameters appearing in the exception message.

12.6. Filterable Handler

iConnect.NET exception handling module is directly built on the basis of Enterprise Library Exception Handling Application Block (EHAB), and we create a series of customer exception handlers to perform the common exception handling tasks. The big limitation for EHAB is that it adopts strict type specific exception handling manner only. That means that the exception handling policy is complete based on the type of the caught exception, and the exception type determines the exception handlers.

In most situations, the same type of exceptions are thrown in different scenarios. For example, the `SqlException` exception will always be thrown when errors happen in the data access

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

against SQL Server, regardless of connection related error or violating the consistence restrict. iConnect.NET provides a special custom exception handler to change the current exception handling manner. Instead of current pure type based exception handler allocation, the new exception will be used to allocate the exception handlers based on a given property value of the exception thrown. Such a custom exception handler is `FilterableExceptionHandler`.

For example, we need to handle the `SqlException` based on the value of `Number` property. We have the following configuration, in which two `Number` property specific filters are configured. The property values of the two filters are 1000 and 2000. If the `Number` of the caught `SqlException` is 1000, the exception handlers of filter1 will be used to handle it. Similarly, the exception handlers of filter2 will be used to handler the `SqlException` with `Number` property is 2000. Otherwise the `SqlException` will be delivered to the handlers configured in the <default> node.

```
<add name="Filterable Handler Policy">
     <exceptionTypes>
          <add type="System.Data.SqlClient.SqlException,
                     System.Data"
                postHandlingAction="ThrowNewException" name="SqlException">
            <exceptionHandlers>
              <add name="Filterable Handler"
                   type="NCS.IConnect.ExceptionHandling.FilterableHandler,
                         NCS.IConnect.ExceptionHandling" >
                <filters>
                  <add name="filter1"
                       type="NCS.IConnect.ExceptionHandling.PropertyValueEquivalencePipeline,
                             NCS.IConnect.ExceptionHandling"
                       property="Number"
                       value="1000">
                    <exceptionHandlers>
                      ...Handlers...
                    </exceptionHandlers>
                  </add>
                  <add name="filter1"
                       type="NCS.IConnect.ExceptionHandling.PropertyValueEquivalencePipeline,
                             NCS.IConnect.ExceptionHandling"
                       property="Number"
                       value="2000">
                    <exceptionHandlers>
                      ...Handlers...
                    </exceptionHandlers>
                  </add>
                </filters>
                <default>
                  ...Handlers...
                </default>
              </add>
            </exceptionHandlers>
          </add>
        </exceptionTypes>
</add>
```

The filter can also be based on a range, E.G. Number <20000; Number > 50000; 2000<Number<=50000. In the following configuration, there are configured three range based filters. The number ranges for the three filters are separately: Number<10000, 10000<=Number<=50000, Number >50000.

```
<add name="Filterable Handler Policy">
        <exceptionTypes>
          <add type="System.Data.SqlClient.SqlException,
                     System.Data"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                postHandlingAction="ThrowNewException" name="SqlException">
            <exceptionHandlers>
              <add name="Filterable Handler"
                   type="NCS.IConnect.ExceptionHandling.FilterableHandler,
                         NCS.IConnect.ExceptionHandling" >
                <filters>
                  <add name="filter1"
                       type="NCS.IConnect.ExceptionHandling.PropertyValueRangePipeline,
                             NCS.IConnect.ExceptionHandling"
                       property            ="Number"
                       lowerBound          = ""
                       lowerRangeBoundType = "Ignore"
                       upperBound          = "10000"
                       upperRangeBoundType = "Exclusive">
                    <exceptionHandlers>
                      ...Handlers...
                    </exceptionHandlers>
                  </add>
                  <add name="filter2"
                       type="NCS.IConnect.ExceptionHandling.PropertyValueRangePipeline,
                             NCS.IConnect.ExceptionHandling"
                       property            ="Number"
                       lowerBound          = "10000"
                       lowerRangeBoundType = "Inclusive"
                       upperBound          = "50000"
                       upperRangeBoundType = "Inclusive">
                    <exceptionHandlers>
                      ...Handlers...
                    </exceptionHandlers>
                  </add>
                    <add name="filter3"
                         type="NCS.IConnect.ExceptionHandling.PropertyValueRangePipeline,
                               NCS.IConnect.ExceptionHandling"
                         property            ="Number"
                         lowerBound          = "50000"
                         lowerRangeBoundType = "Ignore"
                         upperBound          = ""
                         upperRangeBoundType = "Exclusive">
                    <exceptionHandlers>
                      ...Handlers...
                    </exceptionHandlers>
                  </add>
                </filters>
                <default>
                  ...Handlers...
                </default>
              </add>
            </exceptionHandlers>
          </add>
        </exceptionTypes>
</add>
```

Single scalar value based filter need configure the type as `NCS.IConnect.ExceptionHandling.PropertyValueEquivalencePipeline`, and range filter specific type is `NCS.IConnect.ExceptionHandling.PropertyValueRangePipeline`. `PropertyValueRangePipeline` has the following four configuration properties:

- *lowerBound*: The value of lower bound, and it is optional if lowerRangeBoundType = Ignore;
- *upperBound*: The value of upper bound, and it is optional if upperRangeBoundType = Ignore;
- *lowerRangeBoundType*: Indicate whether to exclude, include or ignore the lower bound value. There are three options: Ignore, Exclusive and Inclusive, and default value is Ignore.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- *upperRangeBoundType*: Similar to the lowerRangeBoundType, it indicates whether to exclude, include or ignore the upper bound value.

12.7. WCF Integration

WCF is message based communication infrastructure, only `FaultException`, which will be converted to a message fault, can be propagated from service side to client side. We must adopt special mechanism to integrate EHAB with WCF. As the following figure suggests, the original exception will directly be handled according to the configured exception policy of service side. The final exception will be wrapped as a `FaultException<ServiceExceptionDetail>` instance, which includes full information to reconstruct the same exception.

`FaultException<ServiceExceptionDetail>` is thrown from service side, and it can be caught as is at client side. Then the wrapped exception based information is extracted from the `FaultException<ServiceExceptionDetail>` instance, and reconstruct a new exception. Finally, the reconstructed exception will be handled according to the configured exception policy at client side.

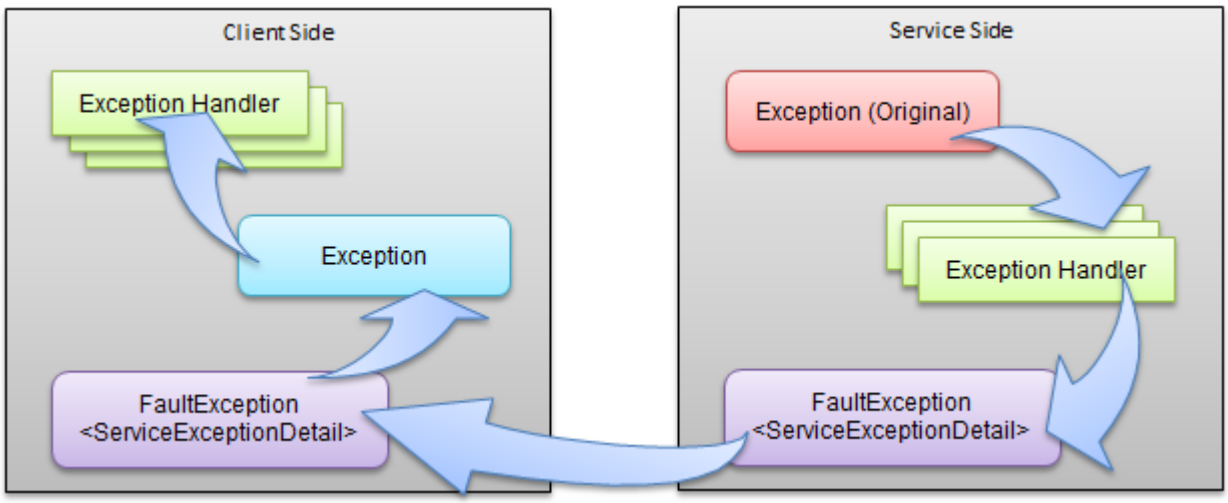


Figure 12.3. Integration between WCF and EHAB

iConnect.NET implements the above mentioned integration between WCF and EHAB by WCF extension: `ExceptionHandlingBehavior`. We just need to apply this behavior to the target service, which needs to use the EHAB to handle the exception thrown from the operation execution. You can directly apply the ExceptionHandlingBehaviorAttribute to the service class below, and the parameter is the name of the exception policy.

```
[ExceptionHandlingBehavior("servicePolicy")]
public class OrderService: IOrderService
{
    //...
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

We can also the service behavior in configurable manner, and the related configuration is like the code snippet below.

```
<configuration>
 <configSections>
    <section name="exceptionHandling"

type="Microsoft.Practices.EnterpriseLibrary.ExceptionHandling.Configuration.ExceptionHandlingSettings,
             Microsoft.Practices.EnterpriseLibrary.ExceptionHandling" />
  </configSections>
 <system.serviceModel>
  <behaviors>
   <serviceBehaviors>
    <behavior name="serviceBehavior">❶
     <exceptionHandling exceptionPolicy="svcPolicy" />❷
    </behavior>
   </serviceBehaviors>
  </behaviors>
  <services>
   <service behaviorConfiguration="serviceBehavior"❸ name="NCS.IConnect.Sample.OrderService">
    <endpoint binding="wsHttpBinding" contract="NCS.IConnect.Sample.IOrderService" />
   </service>
  </services>
    <extensions>
      <behaviorExtensions>
       <add name="exceptionHandling"
            type="NCS.IConnect.ExceptionHandling.WCF.ExceptionHandlingBehaviorElement,
                  NCS.IConnect.ExceptionHandling.WCF"/>❹
      </behaviorExtensions>
    </extensions>
 </system.serviceModel>
 <exceptionHandling>
    <exceptionPolicies>
      <add name="svcPolicy">❺
       ...
      </add>
    </exceptionPolicies>
  </exceptionHandling>
</configuration>
```

❶ Named service behavior including the `ExceptionHandlingBehavior` configuration;

❷ Named ExceptionHandlingBehavior configuration with the name of exception policy;

❸ Refer to the above named service behavior;

❹ `ExceptionHandlingBehavior` configuration element settings.

❺ Named exception policy configuration, which is referenced by 2.

`ExceptionHandlingBehavior` just only handle the caught exception according to the specified exception policy, which is "svcPolicy" in the above demonstration. For the caller to invocate the service, what is caught is always a `FaultException<ServiceExceptionDetail>` once exception is thrown at service side. iConnect.NET provides you with a `NCS.IConnect.ExceptionHandling.WCF.ServiceOperationInvoker` class, which helps to achieve:

- Normally perform service operation invocation and return the result;
- Close or abort the channel after completing service invocation, even encountering error;
- Once `FaultException<ServiceExceptionDetail>` is caught, the wrapped exception related information is extracted and a new exception is reconstructed and thrown.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`ServiceOperationInvoker` defines four static overloaded Invoke methods. The ones with `Func<TChannel, TResult>` type parameter is used to invoke the operation with a concrete return value, and the other two with `Action<TChannel>` type parameter is used to invoke the operation with viod return value. The generic type `TChannel` is the type of service contract, and `TResult` is the type of return value. `CreateProxy` help you to create the service proxy instance, and `ChannelFactory` caching is applied in this method to improve performance. `HandleCommunicationException` method can be invoked to manually handle the communication based exception. Automatic channel closeing/aborting, as well as wrapped exception re-throw, is performed in this method.

```
public class ServiceOperationInvoker
{
    public static TResult Invoke<TChannel, TResult>(Func<TChannel, TResult> function, string
endpointConfigurationName);
    public static void Invoke<TChannel>(Action<TChannel> action, string endpointConfigurationName);
    public static TResult Invoke<TChannel, TResult>(Func<TChannel, TResult> function, TChannel proxy);
    public static void Invoke<TChannel>(Action<TChannel> action, TChannel proxy);

    public static TChannel CreateProxy<TChannel>(string endpointConfigurationName);
    public static void HandleCommunicationException(ICommunicationObject channel, Exception ex);
}
```

Except for `ServiceOperationInvoker`, an additional generic `ServiceOperationInvoker<TChannel>` is defined in the same namespace, and the genetic type is the service contract type. From the its definition in the following code snippet, we can see two similar Invoke instance methods are be defined. The construct accepts a string parameter indicating the configured name of the endpoint to the service.

```
public class ServiceOperationInvoker<TChannel> : ServiceOperationInvoker
{
    public string EndpointConfiguraionName{ get; private set; }
    public ServiceOperationInvoker(string endpointConfigurationName);

    public void Invoke(Action<TChannel> action);
    public TResult Invoke<TResult>(Func<TChannel, TResult> function);
    public TChannel CreateProxy();
}
```

It is recommended to create a single service proxy class to perform service invocation. In general, such a service proxy inherits `NCS.IConnect.Architecture.Client.WcfProxyBase<TChannel>` and implements the service contract interface. In `WcfProxyBase<TChannel>`, an `Invoker` property of `ServiceOperationInvoker<TChannel>` is defined.

```
public class WcfProxyBase<TChannel>: ApplicationContextAwareObject
{
    //Others...
    protected ServiceOperationInvoker<TChannel> Invoker{ get; private set; }
}
```

The following is a demonstrative custom service proxy class: `OrdersProxy`. We just need call the Invoke method to invocate the service operation.

```
public class OrdersProxy : WcfProxyBase<IOrdersService>, IOrdersService
{
    public OrdersProxy():base(Constants.Endpoint)
    {}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public void OrderProducts(Order order)
    {
        this.Invoker.Invoke(proxy=>proxy.OrderProducts(order));
    }
}
```

In the `Invoke` methods both in `ServiceOperationInvoker` and `ServiceOperationInvoker<TChannel>`, the accepted parameter type if a *delegate*, so they do not support *ref/out* parameter. If any *ref/out* parameter is defined in your service operation method, you need follow the programming below.

```
public class OrdersProxy : WcfProxyBase<IOrdersService>, IOrdersService
{
    public OrdersProxy():base(Constants.Endpoint)
    {}
    public void OrderProducts(Order order, out bool approved)
    {
        approved = false;
        IOrdersService proxy = this.Invoker.CreateProxy();
        try
        {
            proxy.OrderProducts(order, out approved);
        }
        catch (Exception ex)
        {
            ServiceOperationInvoker.HandleCommunicationException(proxy as ICommunicationObject, ex);
        }
        finally
        {
            (proxy as ICommunicationObject).Close();
        }
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 13. Logging

13.1. Introduction

EntLib Logging application block is directly used as the Logging Framework of iConnect.NET. The Logging application block allows you to decouple your logging functionality from your application code. The block can route log entries to a Windows® Event Log, a database, or a text (or XML) file. It can also generate an e-mail message containing the logging information, a message you can route through Windows Message Queuing (using a distributor service provided with the block), or a Windows Management Instrumentation (WMI) event. And, if none of these built-in capabilities meets your requirements, you can create a provider that sends the log entry to any other custom location or executes some other action.

In your application, you simply generate a log entry using a suitable logging object, such as the `LogWriter` class, and then call a method to write the information it contains to the logging system. The Logging block routes the log message through any filters you define in your configuration, and on to the listeners that you configure. Each listener defines the target of the log entry, such as Windows Event Log or an e-mail message, and uses a formatter to generate suitably formatted content for that logging target.

You can see from this that there are many objects involved in this multi-step process, and it is important to understand how they interact and how the log message flows through the pipeline of processes. The following figure shows the overall process in more detail, and provides an explanation of each stage.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

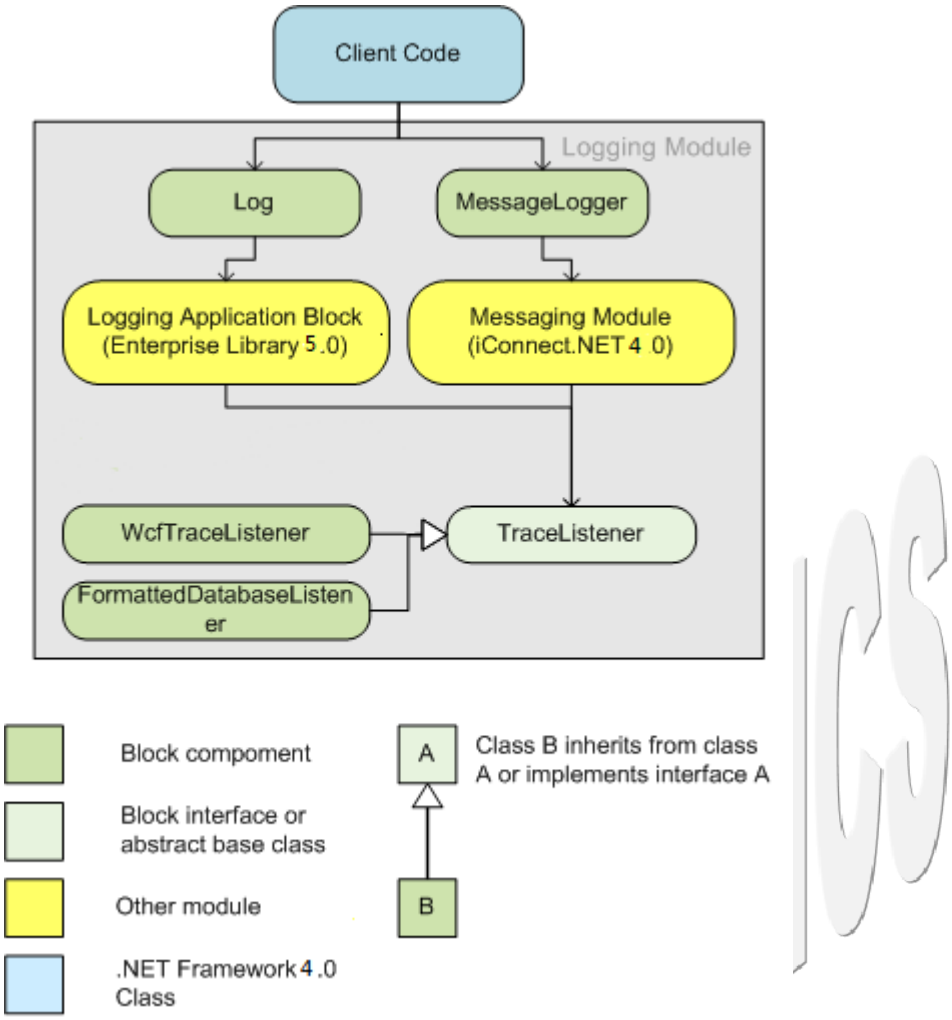


Figure 13.1. An overview of the logging process and the objects in the Logging block

- *Creating the Log Entry*: The user creates a `LogWriter` instance, uses it to create a new `LogEntry`, and passes it to the Logging block for processing. Alternatively, the user can create a new `LogEntry` explicitly, populate it with the required information, and use a `LogWriter` to pass it to the Logging block for processing.

- *Filtering the Log Entry:* The Logging block filters the `LogEntry` (based on your configuration settings) for message priority, or categories you added to the `LogEntry` when you created it. It also checks to see if logging is enabled. These filters can prevent any further processing of the log entries. This is useful, for example, when you want to allow administrators to enable and disable additional debug information logging without requiring them to restart the application.

- *Selecting Trace Sources:* Trace sources act as the link between the log entries and the log targets. There is a trace source for each category you define in the logging block configuration; plus, there are three built-in trace sources that capture all log entries, unprocessed entries that do not match any category, and entries that cannot be processed due to an error while logging (such as an error while writing to the target log).

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- *Selecting Trace Listeners:* Each trace source has one or more trace listeners defined. These listeners are responsible for taking the log entry, passing it through a separate log formatter that translates the content into a suitable format, and passing it to the target log. Several trace listeners are provided with the block, and you can create your own if required.

- *Formatting the Log Entry:* Each trace listener can use a log formatter to format the information contained in the log entry. The block contains log message formatters, and you can create your own formatter if required. The text formatter uses a template containing placeholder that makes it easy to generate the required format for log entries.

iConnect.NET provides two types of `TraceListener`: `FormattedDatabaseTraceListener` and `WcfTraceListener`. `FormattedDatabaseTraceListener` is used to write the formatted `LogEntry` into logging table in database, and `WcfTraceListener` make it possible to remotely perform logging.

13.2. Getting Started

This section leads you through the steps necessary to create a logging demonstration application:

Step I: Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first need to execute the following SQL script against the this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*LoggingDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3B9413F6-3644-4733-8CD6-DA08B203D752'
           ,'LoggingDemo'
           ,'loggingdemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step II: Create a Console Application and Configuring Logging

Create a Console application using your Visual Studio, and add the references to the following assemblies, which are all located in the /bin folder of iConnect.NET installation directory.

- Microsoft.Practices.EnterpriseLibrary.Logging.dll
- NCS.IConnect.Logging.dll
- NCS.IConnect.Logging.Database.dll

Then add an App.config, and copy the following configuration snippet to override the existing content of the configuration file. Do not forget revise the connect string to connect to appropriate database.

```
<configuration>
  <configSections>
    <section name="loggingConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.LoggingSettings,
                   Microsoft.Practices.EnterpriseLibrary.Logging" />
    <section name="iConnect.logging"
             type="NCS.IConnect.Logging.Configuration.LoggingSettings,
                   NCS.IConnect.Logging"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient"/>
  </connectionStrings>
  <iConnect.logging defaultLogViewerProvider="dbLogViewerProvider">
    <logViewerProviders>
      <add name="dbLogViewerProvider"
           type="NCS.IConnect.Logging.DbLogViewerProvider,
                 NCS.IConnect.Logging.Database"
           database="iConnectDb"
           applicationName="loggingDemo"
           pageSize="10"
           returnRecordCount="true"/>
    </logViewerProviders>
  </iConnect.logging>
  <loggingConfiguration tracingEnabled="true"
                        defaultCategory="Troubleshooting">
    <listeners>
      <add name="flatFileTraceListener"
          type="Microsoft.Practices.EnterpriseLibrary.Logging.TraceListeners.FlatFileTraceListener,
                 Microsoft.Practices.EnterpriseLibrary.Logging"

listenerDataType="Microsoft.Practices.EnterpriseLibrary.Logging.Configuration.FlatFileTraceListenerData,
                              Microsoft.Practices.EnterpriseLibrary.Logging"
           fileName ="loggingDemo.log"
           header="----------------------------------------------------------"
           footer="" formatter="textFormatter"
       />
      <add name="dbTraceListener"
           type="NCS.IConnect.Logging.FormattedDatabaseTraceListener,
                 NCS.IConnect.Logging.Database"
           listenerDataType="NCS.IConnect.Logging.Configuration.FormattedDatabaseTraceListenerData,
                              NCS.IConnect.Logging.Database"
           database="iConnectDb"
           formatter="textFormatter"
           applicationName="LoggingDemo"/>
    </listeners>
    <formatters>
      <add
        name="textFormatter"
        type="Microsoft.Practices.EnterpriseLibrary.Logging.Formatters.TextFormatter,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
              Microsoft.Practices.EnterpriseLibrary.Logging"
        template="Message:{message}
Category:            {category}
Severity:            {severity}
Activity Id:         {activity}"/>
    </formatters>
    <categorySources>
      <add name="Troubleshooting" switchValue="All">
        <listeners>
          <add name="flatFileTraceListener"/>
        </listeners>
      </add>
      <add name="Auditing" switchValue="All">
        <listeners>
          <add name="dbTraceListener"/>
        </listeners>
      </add>
    </categorySources>
    <specialSources>
      <errors name="errors" switchValue="All">
        <listeners>
          <add name="flatFileTraceListener"/>
        </listeners>
      </errors>
    </specialSources>
  </loggingConfiguration>
</configuration>
```

There are two category sources, Troubleshooting and Auditing, are configured. Two different `TraceListener`, `FlatFileTraceListener` and `FormattedDatabaseTraceListener` are separately applied to the two category source. As their names suggest, `FormattedDatabaseTraceListener` writes the log entry to database table, whereas the `FlatFileTraceListener` specific logging storage is a given flat file.

Step III: Write and View Logging

In the Main method of the Console application, we call the static class Log's Write method to write two log entries, whose categories are separately Troubleshooting and Auditing.

```
Log.Write("This is an auditing logentry for this purpose of testing.", "Auditing");
Log.Write("This is a troubleshooting logentry for this purpose of testing.", "Troubleshooting");
```

According to the configuration, the first message entry of Auditing category should be written into database table. If you view the T_IC_LOGGING_LOG table, you will find the log entry based record is there. As with the second log entry of Troubleshooting category, it will be written into the configured log file (loggingDemo.log of the current directory). The following is the content of the log file.

```
----------------------------------------------------------
Message:This is a troubleshooting logentry for this purpose of testing.
Category:         Troubleshooting
Severity:         Information
Activity Id:      00000000-0000-0000-0000-000000000000
```

Because iConnect.NET Logging just extends EntLib Logging Application Block, all of the application programing interface of EntLib can be used. The information about how to configure and program EntLib Logging Application will not be included in this developer's guide. If you want to know them, you can refer to the online EntLib developer's guide

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

(`http://msdn.microsoft.com/en-us/library/ff953181(v=PandP.50).aspx`). Our developer's guide only contains the extension related content.

13.3. Writing Log Entries

13.3.1. LogEntry

In general, the basic information to be logged includes event ID, message, priority, and categories you specify in the configuration. However you can collect some additional contextual information, E.G. Application Domain Name, Machine Name, Process ID/Name, Thread ID etc, and write them into log storage. All of the information is encapsulated in a `LogEntry` object. The following code snippet lists all of the public properties of `LogEntry`.

```
[Serializable, InstrumentationClass(InstrumentationType.Event), ManagedName("LogEntryV20"),
XmlRoot("logEntry")]
public class LogEntry : ICloneable
{
    public Guid ActivityId { get; set; }
    public string ActivityIdString { get; }
    public string AppDomainName { get; set; }
    public ICollection<string> Categories { get; set; }
    public string[] CategoriesStrings { get; }
    public string ErrorMessages { get; }
    public int EventId { get; set; }
    public IDictionary<string, object> ExtendedProperties { get; set; }
    public string LoggedSeverity { get; }
    public string MachineName { get; set; }
    public string ManagedThreadName { get; set; }
    public string Message { get; set; }
    public int Priority { get; set; }
    public string ProcessId { get; set; }
    public string ProcessName { get; set; }
    public Guid? RelatedActivityId { get; set; }
    public TraceEventType Severity { get; set; }
    public DateTime TimeStamp { get; set; }
    public string TimeStampString { get; }
    public string Title { get; set; }
    public string Win32ThreadId { get; set; }
}
```

However, collecting additional context information can be expensive in processing terms and, if you are not going to use the information, wastes precious resources and may affect performance. Therefore, the Logging block only collects other less commonly used information from the environment, which you might require only occasionally, if you specify that you want this information when you create the `LogEntry` instance. Four classes within the Logging block can collect specific sets of context information that you can add to your log entry. This includes COM+ diagnostic information, the current stack trace, the security-related information from the managed runtime, and security-related information from the operating system. There is also a dictionary property for the log entry where you can add any additional custom information you require, and which must appear in your logs.

EntLib Logging Application Block provides us with two static facade types, `Microsoft.Practices.EnterpriseLibrary.Logging.Logger` and `Microsoft.Practices.EnterpriseLibrary.Logging.Tracer` to program logging. Except for

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

the two, you can also use the following two facade classes provided in iConnect.NET Logging: `NCS.IConnect.Logging.Log` and `NCS.IConnect.Logging.Messaging.MessageLogger`.

13.3.2. The Logging Facade: Log

As the following code snippet illustrates, there are two Write methods defined in `Log`. You can in advanced prepare a `LogEntry` and call the first overload to achieve logging. If you just need to specify the log message and categories, you can call the second overload. The default values of the rest properties of `LogEntry` are captured.

```
public static class Log
{
    //Others
    public static void Write(LogEntry logEntry);
    public static void Write(string data, params string[] categories);
}
```

Besides the two static Write methods, there are the following three methods defined in Log.

```
public static class Log
{
    //Others
    public static void FlushContextItems();❶
    public static bool IsLoggingEnabled();❷
    public static void SetContextItem(object key, object value);❸
}
```

❶ When this method is called, the `LogWriter`'s `FlushContextItems` is invoked to Empties the context items dictionary;

❷ Return a `Boolean` value indicating whether the logging is enabled or disabled;

❸ When this method is called, the `LogWriter`'s `SetContextItem` is invoked to add a key/value pair to the CallContext dictionary. Context items will be recorded with every log entry.

13.3.3. Messaging Integration

Regardless of `Logger/Tracer` provided by EntLib Logging Application Block or the Log defined in iConnect.NET Logging, you must provide a complete literal message, which is usually hard code in your program. As you know, leveraging iConnect.NET Messaging, we can maintain message entries in separately storage, database or XML file. If we we message provision mechanism provided by Messaging, the hard code programming can be avoided. Logging integrated with iConnect.NET Messaging can be achieved by the special facade `MessageLogger`. According to the following code snippet, we can see the six methods of Log are also defined `MessageLogger`. Besides these, the `WriteMessage` and `WriteMessageN` (N=0,1,2,3, stand for the number of placeholder in the message template) methods with the MessageEntry/MessageEntryN and the arguments to replace the placeholder as the input parameters are defined.

```
public static class MessageLogger
{
    public static void FlushContextItems();
    public static bool IsLoggingEnabled();
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public static void SetContextItem(object key, object value);
    public static bool ShouldLog(LogEntry logEntry);
    public static void Write(LogEntry logEntry);
    public static void Write(string data, params string[] categories);

    public static void WriteMessage(MessageEntry message, object[] args, params string[] categories);
    public static void WriteMessage0(MessageEntry0 message, params string[] categories);
    public static void WriteMessage1(MessageEntry1 message, object arg0, params string[] categories);
    public static void WriteMessage2(MessageEntry2 message, object arg0, object arg1, params string[]
categories);
    public static void WriteMessage3(MessageEntry3 message, object arg0, object arg1, object arg2,
params string[] categories);
}
```

When performing Messaging based logging, you can first get the log message specific `MessageEntry` with the application programming interface of Messaging, can then call the `WriteMessage` method with the arguments to replace the placeholder and logging categories specified. Once this method invoked, iConnect.NET Messaging will format the given message entry and generate a complement literal log message. When you generate the code using Messaging code generation, placeholder number based `MessageEntry` (`MessageEntry0`, `MessageEntry1`, `MessageEntry2` and `MessageEntry3`) classes are generated, and you can pass them into corresponding `WriteMessageN` methods. The following is demonstrative programming.

```
//Write the A00N message with the arguments: a, b, c and d to the default category.
MessageLogger.WriteMessage(Messages.A00N, new object[] {"a", "b", "c", "d"});
//Write the A000 message that has no arguments to the db category.
MessageLogger.WriteMessage0(Messages.A000, "db");
//Write the A001 message that has one argument to the db category.
MessageLogger.WriteMessage1(Messages.A001, "a", "db");
//Write the A003 message that has three arguments to the default category.
MessageLogger.WriteMessage3(Messages.A003, "a", "b", "c");
```

13.4. `FormattedDatabaseTraceListener` and `WcfTraceListener`

In the logging infrastructure, the final logging is implemented by `TraceListener`, which write the formatted log message into the target log storage. Two custom `TraceListener` are provided by iConnect.NET Logging, they `FormattedDatabaseTraceListener` and `WcfTraceListener`.

13.4.1. `FormattedDatabaseTraceListener`

As their names suggest, `FormattedDatabaseTraceListener` is used write the formatted log message into log table in database. There are totally following three logging related tables:

- T_IC_LOGGING_CATEGORY: This table maintain all logging category;
- T_IC_LOGGING_LOG: The table store the concrete log entry without logging category;
- T_IC_LOGGING_CATEGORY_LOG: This is a relationship table between the above table.

The `FormattedDatabaseTraceListener` is used in service side in distributed application or non-distributed application. As well as other database based components, you need to specify the database (connection string name) and application in configuration. The following

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`FormattedDatabaseTraceListener` specific configuration snippet is used in the getting-started application.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <loggingConfiguration tracingEnabled="true" defaultCategory="Troubleshooting">
   ...
    <listeners>
      <add name="dbTraceListener"
           type="NCS.IConnect.Logging.FormattedDatabaseTraceListener,
                 NCS.IConnect.Logging.Database"
           listenerDataType="NCS.IConnect.Logging.Configuration.FormattedDatabaseTraceListenerData,
                             NCS.IConnect.Logging.Database"❶
           database="iConnectDb"❷
           formatter="textFormatter"❸
           applicationName="LoggingDemo"❹/>
    </listeners>
  </loggingConfiguration>
</configuration>
```

❶ The assembly qualified name of the type to define the `FormattedDatabaseTraceListener` specific configuration element;

❷ The configuration name of connection string to connect the logging database, in which the aforementioned tables exist;

❸ The connection name of the log entry formatter;

❹ The name of the current application, which corresponds to a particular record in T_IC_APP table.

13.4.2. `WcfTraceListener`

In the distributed application, we can use `WcfTraceListener` at client side to deliver the log entry to server side by invoking logging service. As with `WcfTraceListener` specific configuration, except for the configuration property of *`listenerDataType`*, we just need to specify the configuration name of the endpoint to invocate the logging service. The following is the demonstrative configuration.

```
<configuration>
  <loggingConfiguration tracingEnabled="true" defaultCategory="Troubleshooting">
   ...
    <listeners>
      <add name="dbTraceListener"
           type="NCS.IConnect.Logging.WcfTraceListener,
                 NCS.IConnect.Logging"
           listenerDataType="NCS.IConnect.Logging.Configuration.WcfTraceListenerData,
                             NCS.IConnect.Logging"
           endpoint="loggingservice"/>
    </listeners>
  </loggingConfiguration>
</configuration>
```

13.5. Viewing Log Entries

Besides the aforementioned custom `TranceListener`, the functionality of logging viewing is also provided in iConnect.NET Logging to make you can search log entries based on different cafeterias.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

13.5.1. Configuration

Like other iConnect.NET components, provider based design is adopted in iConnect.NET Logging. All of the log entry search functionalities are implemented in a particular `LogViewerProvider`. The following two built-in `LogViewerProvider` are provided: `DbLogViewerProvider` and `WcfLogViewerProvider`, which are separately defined the `NCS.IConnect.Logging.Database` and `NCS.IConnect.Logging` assemblies. We use configuration to control which type of `LogViewerProvider` is used.

`DbLogViewerProvider` is used in server side in distributed application or non-distributed application. Except for the two common configuration properties, database and *`applicationName`*, must be specified in configuration, there are extra configuration properties defined in `DbLogViewerProvider` specific configuration: *`pageSize`* and *`returnRecordCount`*. Paging mechanism is used when log entry search, and *`pageSize`* is used to control the maximum count of log entries for each page. The *`returnRecordCount`* indicates whether the logging entry count should be returned. They both configuration properties are optional, and their default values are separately 10 and false. You can configure the `DbLogViewerProvider` following the demonstrative configuration below.

```
<configuration>
  <configSections>
    <section name="iConnect.logging"
             type="NCS.IConnect.Logging.Configuration.LoggingSettings,
                   NCS.IConnect.Logging"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient"/>
  </connectionStrings>
  <iConnect.logging defaultLogViewerProvider="dbLogViewerProvider">
    <logViewerProviders>
      <add name="dbLogViewerProvider"
           type="NCS.IConnect.Logging.DbLogViewerProvider,
                 NCS.IConnect.Logging.Database"
           database="iConnectDb"
           applicationName="LoggingDemo"
           pageSize="10"
           returnRecordCount="true"/>
    </logViewerProviders>
  </iConnect.logging>
 </configuration>
```

`WcfLogViewerProvider` is used in distributed application for remotely log entry search. As with its specific configuration, just only the configuration name of WCF endpoint to invocate the log viewing service needs to be set. The following is a demonstrative configuration with `WcfLogViewerProvider`. The service invocated by this provider is `NCS.IConnect.Logging.LogViewerWcfService`.

```
<configuration>
  <configSections>
    <section name="iConnect.logging"
             type="NCS.IConnect.Logging.Configuration.LoggingSettings,
                   NCS.IConnect.Logging"/>
  </configSections>
  <system.serviceModel>
      ...
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
      <client>
          <endpoint name="logViewService"
                    address="http://127.0.0.1/demoServices/LogViewer.svc"
                    binding="wsHttpBinding"
                    contract="NCS.IConnect.Logging.IWcfLogViewer" .../>
      </client>
  </system.serviceModel>
  <iConnect.logging defaultLogViewerProvider="dbLogViewerProvider">
    <logViewerProviders>
      <add name="dbLogViewerProvider"
           type="NCS.IConnect.Logging.WcfLogViewerProvider,
                 NCS.IConnect.Logging"
           endpoint="logViewService"/>
    </logViewerProviders>
  </iConnect.logging>
 </configuration>
```

13.5.2. `LogViewer`

The static facade type `LogViewer` is used for viewing log entries programming. You can invoke the `FindAllLogs` method to get all log entries, and a series of overloaded `FindLogs` methods make you can get the log entries based on different search criteria combination. What are returned in these method are a strongly typed `DataTable`, `LoggingDataSet.T_IC_LOGGING_LOGDataTable`.

```
public static class LogViewer
{
    //Others
    public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindAllLogs();
   public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindLogs(LogSearchCriteria searchCriteria,
int pageIndex, out int count);
   public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindLogs(string categoryName, int pageIndex,
out int count);
   public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindLogs(DateTime? fromDateTime, DateTime?
toDateTime, int pageIndex, out int count);
   public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindLogs(int? priority, string severity,
int pageIndex, out int count);
    public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindLogs(string categoryName, DateTime?
fromDateTime, DateTime? toDateTime, int pageIndex, out int count);
    public static LoggingDataSet.T_IC_LOGGING_LOGDataTable FindLogs(string appDomainName, string
machineName, string severity, int pageIndex, out int count);
}
```

There is a special type `LogSearchCriteria` to make you can freely combine the search conditions. According to the code snippet, which represents the definition of `LogSearchCriteria`, we can see it has the same properties as the aforementioned `LogEntry`.

```
public class LogSearchCriteria
{
    //Others
    public string AppDomainName { get; set; }
    public string Category { get; set; }
    public int? EventId { get; set; }
    public DateTime? From { get; set; }
    public string MachineName { get; set; }
    public int? Priority { get; set; }
    public string ProcessId { get; set; }
    public string ProcessName { get; set; }
    public string Severity { get; set; }
    public string ThreadName { get; set; }
    public string Title { get; set; }
    public DateTime? To { get; set; }
    public string Win32ThreadId { get; set; }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Log viewing is the main functionality of `LogViewer`. In addition, you can leverage the same facade type to perform log entry cleanup. You can invoke the ClearAll method to clear all recorded log entries. You can also call the three overloaded Clear methods to control which log entries should be cleared.

```
public static class LogViewer
{
    //Others
    public static void Clear(LogSearchCriteria searchCriteria);
    public static void Clear(string categoryName);
    public static void Clear(DateTime? fromDateTime, DateTime? toDateTime);
    public static void ClearAll();
}
```

13.6. Monitoring Instrumentation

Monitoring instrumentation, which leverages the *EntLib* Logging infrastructure, provides us with an easy-to-use solution to track the layer specific call stack for an activity. Generally speaking, an activity to track is started by end user's keyboard/mouse operation in an user interactive based application. For a standalone service, an new activity will be automatically started if no incoming activity ID is detected when the service operation is invoked. By analyzing the logged monitoring information, we cannot only track the which components and methods are invoked in the call stack, but can get the following detail information as well:

- Environment Information: log time, machine name, application domain name, process ID, process name, thread name, Win32 thread ID and message etc.;
- Activity Information: activity name, user name, module id and start time etc.;
- Execution Time: the ticks to start the tracked method and method completes, the method's execution time;
- Inputs/Outputs: input parameters, reference parameters, output parameters and return value;
- Exception: the information of the exception thrown.

13.6.1. Start New Activity

Monitoring instrumentation is based on activity, so we must create an ambient activity to track. The ambient activity creation can be implemented by instantiating a `NCS.IConnect.Logging.Instrumentation.ActivityContextScope`.

```
public class ActivityContextScope : IDisposable
{
    public ActivityContextScope(string moduleId❶,
                                string activityName❷,
                                string description = ""❸,
                                ActivityContextScopeOption option =
ActivityContextScopeOption.Required❹,
                                IDictionary<string, object> exetenededProperties = null❺);
    public void Dispose();
}
public enum ActivityContextScopeOption
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
{
    Required,❻
    RequiresNew,❼
    Suppress❽
}
```

❶ The module ID;

❷ The name of activity;

❸ The description of activity;

❹ The options to control the ambient activity creation behavior;

❺ The extended properties;

❻ If an ambient activity exists, it will be directly used as the current activity and new activity will never be created;

❼ A new activity will be created and used as the current activity, regardless of whether the ambient activity exists or not;

❽ No ambient activity exists in the scope.

As illustrated in the `ActivityContextScope` definition, it implements the `IDisposable` interface. In general, we create a new `ActivityContextScope` in a using block as below. Once an `ActivityContextScope` is created, a new activity or existing activity will become the ambient activity if we do not specify the option parameter as `Supress`. The ambient activity is stored in the current thread specific *TLS* (Thread Local Storage), and can be detected by the monitoring instrumentation logger in the same thread. If we need to track an asynchronous operation, it must be guaranteed that the `ActivityContextScope` is created in the same thread as the one in which the asynchronous operation is invoked. At the end of the using block where the Dispose method is invoked, the ambient activity will be recovered to the original state.

```
using(ActivityContextScope contextScope = new ActivityContextScope("Order", "Create Order")
{
    //...
}
```

The activity information is encapsulated in the following class of NCS.IConnect.Logging.Instrumentation.ActivityContext, which is defined the NCS.IConnect.Core assembly instead of NCS.IConnect.Logging assembly. The current `ActivityContext` will be created by instantiating the `ActivityContextScope`, and we can get it via the static Current property.

```
[Serializable]
public class ActivityContext
{
    public Guid ActivityId { get; private set; }
    public string ModuleId { get; private set; }
    public string ActivityName { get; private set; }
    public string Description { get; set; }
    public DateTime StartTime { get; private set; }
    public int CallStackLevel { get; set; }❶
    public IDictionary<string, object> ExtenedProperties { get; private set; }

    public ActivityContext(string moduleId, string activityName, string description = "",
IDictionary<string, object> exetenededProperties = null);
    public static ActivityContext Current{get;set;}
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ The depth of the call stack. E.G If a particular activity is started by calling F1 method of A, the `CallStackLevel` of the monitoring log entry for this method is 0. If the method F2 of B is called in this method, the `CallStackLevel` for F2 method invocation specific log entry will be 1.

13.6.2. `MonitoringCallHandler`

As a non-business based crosscutting concern, monitoring instrumentation is separated with business logical, and the monitoring log writing is conducted by the `NCS.IConnect.Logging.Instrumentation.MonitoringCallHandler`. We need to explicitly apply this `CallHandler` to the tracked methods in either attribute decoration manner or configurable manner. For usage of `CallHandler`, please refer to the chapter "`IoC` and `AOP`".

```
[ConfigurationElementType(typeof(MonitoringCallHandlerData))]
public class MonitoringCallHandler : ICallHandler
{
    public string ModuleId { get; set; }
    public string FunctionId { get; set; }
    public string ComponentType { get; set; }
    public int Order { get; set; }

    [InjectionConstructor]
    public MonitoringCallHandler()
        : this(string.Empty, string.Empty, string.Empty)
    { }
    public MonitoringCallHandler(string moduleId, string functionId, string componentType);
}
```

The above code snippet represents the definition of `MonitoringCallHandler`, except for the properties of ModuleId and FuncitonId which represents the ID the the module and function for the tracked method. The ComponentType property of string type represents the layer specific component type. Generally speaking, we use the constants defined in the `NCS.IConnect.Logging.Instrumentation.ComponentTypes` to evaluate this property. As the following code snippet illustrates, all layers corresponds to one constant in the `ComponentTypes` class.

```
public static class ComponentTypes
{
    public const string Business       = "Business";
    public const string DataAccess     = "DataAccess";
    public const string Database       = "Database";
    public const string Service        = "Service";
    public const string ServiceProxy   = "Service Proxy";
    public const string UI             = "User Intrface";
    public const string UnKnwon        = "UnKnwon";
}
```

The following code snippet represents the definition of the `MonitoringCallHandler` specific attribute. The three properties, ComponentType, FunctionId and ModuleId correspond to the ones defined in the `MonitoringCallHandler` type. If the `FunctionId` and `ModuleId` are not explicitly initialized, their default values are separately the name of the tracked method of assembly. As with the property of `ComponentType`, It the tacked types are the sub classes of `iConnect.NET` built-in base class, such as `DataAccessBase`, `BusinessComponentBase`,

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`ServiceBase`, `ServiceProxyBase`, etc, the corresponding layer specific component type will be specified, otherwise it will be Unknown.

```
public class MonitoringCallHandlerAttribute : HandlerAttribute
{
    public MonitoringCallHandlerAttribute();
    public override ICallHandler CreateHandler(IUnityContainer container);

    public string ComponentType { get; set; }
    public string FunctionId { get; set; }
    public string ModuleId { get; set; }
}
```

13.6.3. Monitoring `TraceListener`

The monitoring instrumentation log entries are ultimately written into the persistent storage by monitoring `TraceListener`. iConnect.NET provides us with one `TraceListener`, NCS.IConnect.Logging.Instrumentation.MonitoringDatabaseTranceListener, as the type name suggests, this `TraceListener` directly writes the monitoring log entries into the target database. If we need to perform remote monitoring instrumentation, we can use the afore-mentioned `WcfTraceListener`, which transmits the log entries to the `TraceListenerService` that uses the local `TraceListener` for log entry writing.

The following XML snippet demonstrates how to configure the `MonitoringDatabaseTranceListener`, and we can see it has the same configuration definition as the `FormattedDatabaseTraceListener`. By default, the source of monitoring log entry is "Monitoring", however which can be overridden in monitoring instrumentation configuration.

```
<configuration>
  <loggingConfiguration tracingEnabled="true" defaultCategory="General">
    <listeners>  *
      <add name="MonitoringDatabaseTranceListener"
           type="NCS.IConnect.Logging.Instrumentation.MonitoringDatabaseTranceListener,
                 NCS.IConnect.Logging.Database"

listenerDataType="NCS.IConnect.Logging.Instrumentation.Configuration.MonitoringDatabaseTranceListenerData,
                              NCS.IConnect.Logging.Database"
           database="..."
           formatter="Text Formatter"
           applicationName="..."/>
    </listeners>
    <categorySources>
      <add switchValue="All" name="Monitoring">
        <listeners>
          <add name="MonitoringDatabaseTranceListener"/>
        </listeners>
      </add>
    </categorySources>
    ...
    <formatters>
      <add
        name="Text Formatter" .../>
    </formatters>
  </loggingConfiguration>
</configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

13.6.4. Monitoring Log Entry Retrieval

We can use the `NCS.IConnect.Logging.Instrumentation.MonitoringLogger` to retrieve monitoring log entries. Generally speaking, there are large scale of log entries, so it is only enabled to retrieve a particular activity based log entries. However, we can get all of activity entries based on the search criteria.

```
public static class MonitoringLogger
{
    static MonitoringLogger();
    public static ActivityInfo[] GetActivities(ActivityCriteria criteria);
    public static MonitoringLogEntryInfo[] GetMonitoringLogEntries(string activityId);
}
```

The method `GetActivities` get a array of `NCS.IConnect.Logging.Instrumentation.ActivityInfo`, from which we can get the activity related information. The following code snippet represents the definition of all properties of `ActivityInfo`.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class ActivityInfo
{
    public string ActivityId { get; set; }
    [DataMember]
    public string ActivityName { get; set; }
    [DataMember]
    public string Description { get; set; }
    [DataMember]
    public string ExtenededProperties { get; set; }
    [DataMember]
    public string ModuleId { get; set; }
    [DataMember]
    public DateTime StartTime { get; set; }
    [DataMember]
    public string UserName { get; set; }
}
```

The `NCS.IConnect.Logging.Instrumentation.ActivityCriteria` object which is used as the input parameter of the `GetActivities` method, represents the search criteria. As the following code snippet illustrates, we can make up the search criteria by combining the ID/name of the activity, module ID, start time range and user name. If the property is not explicitly specified, the corresponding filter condition will be ignore. All of the explicitly specified properties will be reset when we call the `Reset` method. The static property `Default` returns an `ActivityCriteria` in which all of the properties have the default value.

```
public class ActivityCriteria
{
    public ActivityCriteria();
    public void Reset();

    public string ActivityId { get; set; }
    public string ActivityName { get; set; }
    public string ModuleId { get; set; }
    public DateTime StartAfter { get; set; }
    public DateTime StartBefore { get; set; }
    public string UserName { get; set; }

    public static ActivityCriteria Default { get; }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

What is returned by calling the `GetMonitoringLogEntries` method is an array of `NCS.IConnect.Logging.Instrumentation.MonitoringLogEntryInfo`, in which all of the monitoring log entry related information is included. The following code snippet represents the definition of all of the properties of `MonitoringLogEntryInfo`.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class MonitoringLogEntryInfo : LogEntryInfo
{
    [DataMember]
    public string ActivityId { get; set; }
    [DataMember]
    public string CallStack { get; set; }
    [DataMember]
    public int? CallStackLevel { get; set; }
    [DataMember]
    public string ComponentType { get; set; }
    [DataMember]
    public long EndTicks { get; set; }
    [DataMember]
    public string Exception { get; set; }
    [DataMember]
    public string ExtendedProperties { get; set; }
    [DataMember]
    public string FunctionId { get; set; }
    [DataMember]
    public string Inputs { get; set; }
    [DataMember]
    public string MethodName { get; set; }
    [DataMember]
    public string ModuleId { get; set; }
    [DataMember]
    public string Outputs { get; set; }
    [DataMember]
    public string ReturnValue { get; set; }
    [DataMember]
    public decimal? SlapsedInSeconds { get; set; }
    [DataMember]
    public long StartTicks { get; set; }
    [DataMember]
    public string TypeName { get; set; }
}
```

13.6.5. Configuration

If we need to apply the `MonitoringCallHandler` to the tracked methods with configuration, we have two choices. We can select the traditional Policy Injection and Unity Interception based Policy Injection configuration. For the two kinds of configuration to apply call handlers, we can refer to *EntLib* and Unity developer's guide. iConnect.NET monitoring instrumentation has its own configuration which is outlined in the following XML snippet.

```
<configuration>
  <configSections>
    <section name="iConnect.instrumentation"
             type="NCS.IConnect.Logging.Instrumentation.Configuration.InstrumentationSettings,
                   NCS.IConnect.Logging"/>
  </configSections>
<iConnect.instrumentation>
  <monitoring enabled="true" ❶
              defaultProvider="dbProvider" ❷
              parameterFormatter="Default"❸
              nullParameterValueString ="NULL"❹
              nilParameterString ="No Parameters"❺
              cannotSerializeString="Not Serializable"❻
              voidReturnTypeString ="N/A"❼
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
              serializationErrorTemplate ="{Message}"❽
              logEntryTile="Instrumentation Monitoring"❾
              logEntryCategory ="Monitoring"❿
              maxItemsInObjectGraph ="50"⓫
              logEntryPriority ="-1"⓬
              monitoringLevel="Full">⓭
    <monitoringProviders>
      <add name="dbProvider"
            type="NCS.IConnect.Logging.Instrumentation.DbMonitoringProvider,
                  NCS.IConnect.Logging.Database"
            applicationName ="..."
            database="..."/>⓮
      <add name="dbProvider"
            type="NCS.IConnect.Logging.Instrumentation.WcfMonitoringProvider,
                  NCS.IConnect.Logging"
            endpoint ="..."/>⓯
    </monitoringProviders>
  </monitoring>
</iConnect.instrumentation>
</configuration>
```

❶ A global switch to enable/disable the monitoring instrumentation. The default value is false;

❷ The name of default `MonitoringProvider` to mainly perform monitoring log entries retrieval;

❸ A `NCS.IConnect.Logging.Instrumentation.Configuration.ParameterFormatterMode` enumeration indicating if the default or custom `ParameterFormatter` is used. The default value is Default, and it should be specified Custom for custom `ParameterFormatter`;

❹ The literal text used as the logged parameter value when the parameter is null. The default value is Null;

❺ The literal text logged for inputs of parameterless method. The default value is "No Parameters";

❻ The literal text used as the logged parameter value when the parameter cannot be serialized. The default value is "Not Serializable";

❼ The literal text logger for output of the method of void return type. The default value is "N/A";

❽ The template used as the logged parameter value when serialization fails. The default value is "{Message}" representing the exception message will be logged;

❾ The title of log entry. The default value is "Monitoring";

❿ The category of log entry. The default value is "Monitoring";

⓫ The maximum number of items in an object graph to serialize or deserialize. The default value is 60;

⓬ The priority of log entry. The default value is -1;

⓭ A NCS.IConnect.Logging.Instrumentation.MonitoringLevel enumeration indicating the monitoring level. There are two monitoring level, Full and Basic. Just only method execution elapsed time is logged for Basic monitoring level;

⓮ The configuration for `NCS.IConnect.Logging.Instrumentation.DbMonitoringProvider`,which get the monitoring log entries directly from database;

⓯ The configuration for `NCS.IConnect.Logging.Instrumentation.WcfMonitoringProvider` which get the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

monitoring log entries by invocating service. The service type is `NCS.IConnect.Logging.Instrumentation.MonitoringService`.

For Full monitoring level, the input and output for any tracked methods will be logged. The parameter and return value formatting is conducted by `ParameterFormatter`. All of the `ParameterFormatter` implements the following `NCS.IConnect.Logging.Instrumentation.IParameterFormatter` interface.

```
public interface IParameterFormatter
{
    LinedString FormateInputs(IParameterCollection inputs);
    LinedString FormateOutputs(IParameterCollection outputs);
    LinedString FormatReturnValue(Type returnType, object returnValue);
}
public class LinedString : List<string>
{
    public LinedString();
    internal LinedString(string str);
    public string AppendLine();
    public string AppendLine(string str, params object[] args);
    public string AppendLine(string str, int indentLevel, params object[] args);
    public override string ToString();
    public string ToString(int indentLevel);

    public IList<string> Lines { get; }
```

By default, the `NCS.IConnect.Logging.Instrumentation.DefaultParameterFormatter` is used for parameter and return value formatting. For usage of custom ParameterFormatter, you can apply it following the following configuration.

```
<iConnect.instrumentation>
  <monitoring  parameterFormatter="Custom"
               customParameterFormatter = "provider1"
    <monitoringProviders>
       <add name="provider1" type="..." .../>
       <add name="provider2" type="..." .../>
    </monitoringProviders>
  </monitoring>
</iConnect.instrumentation>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 14. Audit Trail

14.1. Introduction

Audit trail or audit log is a chronological sequence of audit records, each of which contains evidence directly pertaining to and resulting from the execution of a business process or system function. Audit records typically result from activities such as transactions or communications by individual people, systems, accounts or other entities. The general information related to the activity to audit includes:

- The unique identity of the activity to audit. Because we regard a transaction as an audit unit, the transaction ID is used to uniquely identify a given activity to audit;
- The user account to perform the activity to audit;
- The date time when the activity happened.

Except for the above general audit trail information, the data change caused by the activity will also be logged in some scenarios. In detailed, not only is source table list involved in the activity logged, but also these source table specific record changes are written into the log storage. The audit trail with change data is necessary to the activity involving sensitive data.

iConnect.NET Audit Trail provides you with two kinds of change data capture: trigger and *SQLCDC*. For the trigger based one, the change data capture logic is defined in source table specific trigger. Once data change happens to the source table, its specific triggers will be woken to write the change data into log table. iConnect.NET provide you a T4 based SQL generator, which helps to automatically generate the SQL script of the triggers. If your database is SQL Server 2008 enterprise/develoepr edition, You can adopt *SQLCDC* based audit trail, from which you can gain better performance.

14.2. Getting Started

The sample in this section is used to demonstrate all of the necessary programming steps to write the audit information, including the change data, into the related log table. This sample assumes a SQL Server database with necessary iConnect.NET objects is here. You can use the script file *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql* to create them, if such a database is not available.

Step I: Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first need to execute the following SQL script against the this database to insert a new record, which

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*AuditTrailDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3B9413F6-3644-4733-8CD6-DA08B203D751'
           ,'AuditTrailDemo'
           ,'audittraildemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Step II Create source table and stored procedure

Execute the following SQL script to create a table T_USERS, which will be used as the source table.

```
CREATE TABLE [T_USERS]
(
  [USER_ID]          [VARCHAR](50)   PRIMARY KEY,
  [USER_NAME]        [NVARCHAR](128) NOT NULL,

  [VERSION_NO]       [INT]           NOT NULL,
  [TRANSACTION_ID]   [VARCHAR](50)   NOT NULL,
  [CREATED_BY]       [VARCHAR](256)  NOT NULL,
  [CREATED_TIME]     [DATETIME]      NOT NULL,
  [LAST_UPDATED_BY]  [NVARCHAR](256) NOT NULL,
  [LAST_UPDATED_TIME][DATETIME]      NOT NULL,
)
GO
```

Then execute the following script to create a stored procedure `P_USERS_I`, which will be invoked to insert a new user record.

```
CREATE PROCEDURE P_USERS_I
 @p_user_id         [VARCHAR] (50),
 @p_user_name       [NVARCHAR] (128),
 @p_created_by      [NVARCHAR](256),
 @p_transaction_id  [VARCHAR](50)
AS
BEGIN
 INSERT INTO[T_USERS]
           ([USER_ID]
           ,[USER_NAME]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            ,[LAST_UPDATED_TIME])
 VALUES
           (@p_user_id,
            @p_user_name,
            1,
            @p_transaction_id,
            @p_created_by,
            GETDATE(),
            @p_created_by,
            GETDATE());
END
```

Step III: Generate the script of the audit table and trigger

In this step, we will leverage T4 to generate the SQL script of the source table specific triggers and audit table. You first create a Console application using your Visual Studio. Add a configuration file, and name it "*AuditTrail.CodeGeneration.config*". As the file name suggests, this configuration file will be used as the configuration file of T4 based SQL generator. Copy this following configuration, which include the connection string setting, into the added configuration file. Revise the connection string to suit for your data base distribution and security setting.

```
<configuration>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

Add a text file, and name it *AuditTrail.CodeGeneration.tt*. This file with .tt as its extension is a T4 based template file. Copy the follow code into the template file. If iConnect.NET is not installed into the default directory (C:\Program Files\), you should revise the assembly references accordingly.

```
<#@ template language="C#" hostSpecific="true" debug="true"   #>
<#@ output extension="remove" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Common.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Helpers.Data.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.SqlGeneration.dll"
#>
<#@ assembly
  name="C:\Program Files\NCS iConnect.NET\{version
no}\bin\Microsoft.Practices.EnterpriseLibrary.Common.dll" #>
<#@ import namespace="NCS.IConnect.SqlGeneration.SqlServer" #>
<#@ import namespace="NCS.IConnect.Common" #>
<#@ include file="T4Toolbox.tt" #>
<#
   new AuditTableGenerator(this.Host.ResolvePath("Audittrail.Codegeneration.config"),
       "iConnectDb","dbo","T_USERS").Run();
   new TriggerGenerator(this.Host.ResolvePath("Audittrail.Codegeneration.config"),"iConnectDb",
       "dbo", "dbo", "T_USERS").Run();
#>
```

Once saving the above template, 4 script files, as well as a useless Audittrail.CodeGeneration.remove file, will be generated. As shown in the following figure, the five generated files are included on the project as the dependent files of the template file. The file T_USERS_A includes the DDL to create the source table (T_USERS) specific audit

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

table (T_USERS_A). The rest three are script files which separately contains the DDL to create insert, update and delete triggers. After the script files are generated, you can execute them in the target database.

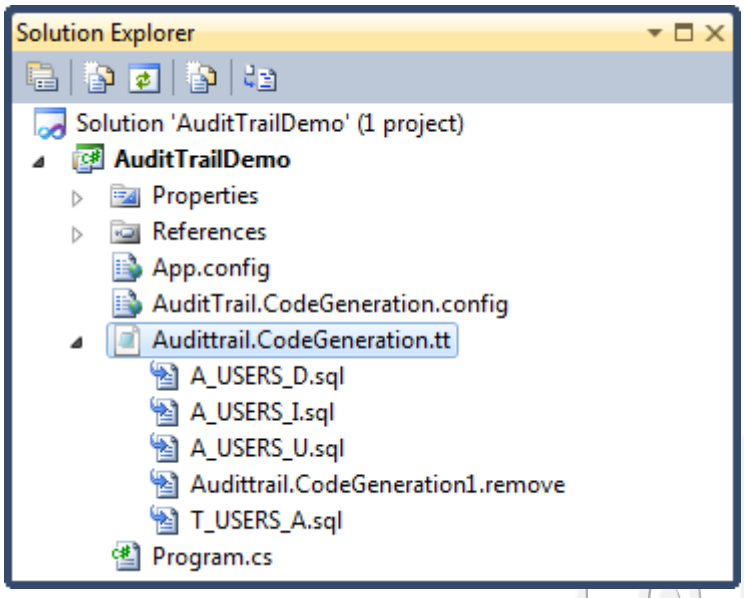


Figure 14.1. File structure of template and generated script files

Step IV Audit the activity of user creation

Now we will demonstrate the programming parts. You first need to add all of the necessary assembly reference as below:

- `System.Transactions`
- `NCS.IConnect.Core`
- `NCS.IConnect.AuditTrail`
- `NCS.IConnect.Helpers.Data`

Then add the App.config for this demonstration application, and copy the following configuration into the configuration file.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="dataConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
                   Microsoft.Practices.EnterpriseLibrary.Data"/>
    <section name="iConnect.auditTrail"
             type="NCS.IConnect.AuditTrail.Configuration.AuditTrailSettings,
                   NCS.IConnect.AuditTrail"/>
 </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="Server=.; Database=iConnectDb; Uid=sa; Pwd=password"
         providerName="System.Data.SqlClient"/>
  </connectionStrings>
  <dataConfiguration defaultDatabase="iConnectDb"/>
  <iConnect.auditTrail autoFlush="true">
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    <storageProviders>
      <add name="multipleTable"
           type="NCS.IConnect.AuditTrail.MultipleTableStorageProvider,
                 NCS.IConnect.AuditTrail"
           applicationName="AuditTrailDemo"
           database="iConnectDb"
           auditTableSuffix="_A"
           encoder="defaultEncoder"/>
    </storageProviders>
    <encoders>
      <add name="defaultEncoder"
           type="NCS.IConnect.AuditTrail.Encoder,
                 NCS.IConnect.AuditTrail" />
    </encoders>
  </iConnect.auditTrail>
</configuration>
```

Finlay, copy the following code into the Main method of the Program.cs. If some type cannot be resolved, please import related namespaces accordingly.

```
static void Main(string[] args)
{
    ApplicationContext.Current.UserName = "Foo";
    using (TransactionScope transactionScope = new TransactionScope())
    {
        using (AuditLog auditLog = new AuditLog("User Creation"))
        {
            var dbHelper = new DbHelper();
            dbHelper.ExecuteNonQuery("P_USERS_I", "001", "Bar",
                ApplicationContext.Current.UserName,
                ApplicationContext.Current.TransactionId);
        }
        transactionScope.Complete();
    }
}
```

The above program leverages the `DbHelper` to execute the P_USERS_I procedure. The user creation based code is placed in the using block of instantiating `AuditLog`. This means audit logging will be performed against the nested operation sequence. In addition, both the data access and audit logging operation are executed in a single transaction.

Run this program, the activity of user creation based audit trail related information will be written into audit log tables: T_IC_AUDIT_LOG, T_IC_AUDIT_LOG_DATA and T_USERS_A. You can query the logged audit trail data against these tables.

14.3. Writing Audit Data into Log Storage

The audit log storage for iConnect.NET is essentially some database tables. The *T_IC_AUDIT_LOG* table is used to store the general audit trail information, including the ID and function name of the transaction to audit, the account of the user to perform the activity, and the date time when the audited activity happened. The detailed information, including the change data, is stored in the *T_IC_AUDIT_LOG_DATA* table and source table specific audit table, E.G. The source table *T_USERS* in the above demonstration specific audit table is *T_USERS_A*, which has similar table schema to the source table.

The general audit trail information is always logged in programmatic (managed code) manner. As with detailed information, it is by default written by triggers in most scenarios. Besides the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

trigger based change data capture, you can also perform audit trial against a `DataSet` and `DbCommand`. Before delving into the variety of audit trail types, we need first know the following two important terms: *Singe Table Audit* and *Multiple Table Audit*.

14.3.1. Single Table Audit V.S. Multiple Table Audit

In the above demonstration, all of the audit trail detail information is distributed to two kinds of audit tables. T_IC_AUDIT_LOG_DATA and source table specific audit tables. In detailed, the source table (T_USERS) specific change data is ultimately stored into its specific audit table (T_USERS). We regard this kind of audit as *Multiple Table Audit*.

The following table illustrates the schema of the T_IC_AUDIT_LOG_DATA database table. The LOG_ID and LOG_SUB_ID makes up the composite primary key of the table. OPERATION indicates the operation types: Insert, Update and Delete.

Table 14.1. T_IC_AUDIT_LOG_DATA Schema

Column Name	Data Type
LOG_ID	`VARCHAR(50)`
LOG_SUB_ID	`INT`
TRANSACTION_ID	`VARCHAR(50)`
OPERATION	`NVARCHAR(20)`
OPERATION_TIME	`DATETIME`
TABLE_NAME	`NVARCHAR(50)`
AUDIT_DATA	`NTEXT`

In the multiple table audit scenario, the AUDIT_DATA column is always empty, since all of the audit data is stored in the source table specific audit table. This column serves another kind of audit: *Single Table Audit*, in which the change data is formated as a single literal string and stored in this column.

The data access operations, including writing audit data into storage, audit data query and achieve are all implemented in the component: `StorageProvider`. There are two built-in storage providers, which specific to the above two kinds of audit. They are `NCS.IConnect.AuditTrail.SingleTableStorageProvider` and `NCS.IConnect.AuditTrail.MultipleTableStorageProvider`.

Which type of storage provider will be used is determined by configuration. The following gives you a complete audit trail configuration, in which the `MultipleTableStorageProvider` is used.

```
<iconnect.auditTrail autoFlush="true"❶>
  <storageProviders>
    <add name="multipleTable"
         type="NCS.IConnect.AuditTrail.MultipleTableStorageProvider,
               NCS.IConnect.AuditTrail" ❷
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        database="iConnectDb"❸
        encoder="defaultEncoder"❹
        auditTableSuffix="_A"❺
        applicationName="Sample" />
  <encoders>
    <add name="defaultEncoder"
        type="NCS.IConnect.AuditTrail.Encoder,
              NCS.IConnect.AuditTrail"/>❻
  </encoders>
 </storageProviders>
</iconnect.auditTrail>
```

❶ autoFlush indicate if the audit data is written to storage on demand when `AuditLog`'s Write method is invoked (*autoFlush* = false), or when the `AuditLog` is disposed (*autoFlush* = true);

❷ The qualified assembly name of the `MultipleTableStorageProvider` type;

❸ The name of the connection string to the database containing the audit log tables;

❹ The name of the Encoder, which is configured in the <encoders> node;

❺ In general, the name of the source table with a suffix is used as its specific audit table. The suffix is specified using the *auditTableSuffix* attribute;

❻ The configuration of the Encoder, which is used to format the audit trail information.

The configuration property *auditTableSuffix* of `MultipleTableStorageProvider` defines the default mapping between the source table and its specific audit table. If a particular source table specific audit table is not named according to such a naming convention, you can manually configure the mapping. As illustrated in the following code snippet, the source table name T_IC_AUDIT_TEST use the T_IC_AUDIT_TEST_LOG table as its audit table.

```
<iconnect.auditTrail autoFlush="true">
  <tableMappings>
    <add name="T_IC_AUDIT_TEST_LOG"❶  transactionTableName="T_IC_AUDIT_TEST"❷/>
  </tableMappings>
  ...
</iconnect.auditTrail>
```

❶ The name of audit table specific to the given source table;

❷ The name of source table.

If you need to transfer from multiple table audit to single table audit, you just need to configure the `SingleTableStorageProvider` instead. The following is a demonstrative `SingleTableStorageProvider` based configuration.

```
<iconnect.auditTrail autoFlush="true">
  <storageProviders>
    <add name="singleTable"
        type="NCS.IConnect.AuditTrail.SingleTableStorageProvider,
              NCS.IConnect.AuditTrail"/>
        applicationName="Sample."
        database="iConnectDb"
        encoder="defaultEncoder" />
    ...
 </storageProviders>
</iconnect.auditTrail>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

14.3.2. Create an AuditLog

All of the audit trail related functionalities can be performed by invoking the facade class: `NCS.IConnect.AuditTrail.AuditLog`. You can instantiate such an `AuditLog` instance by invoking appropriate constructors, which are all defined in the following code snippet.

```
public class AuditLog : IDisposable
{
    public AuditLog();
    public AuditLog(AuditTrailHeader header❶);
    public AuditLog(string functionName❷);
    public AuditLog(AuditTrailHeader header, bool autoFlush❸);
    public AuditLog(string functionName, bool autoFlush);
    public AuditLog(string functionName, string transactionId❹, string userName❺);
   public AuditLog(string functionName, string transactionId, string userName, string appVersion❻);

    public AuditLog(string functionName, string transactionId, string userName, string appVersion,
bool autoFlush);
}
```

❶ An `AuditTrailHeader` instance, which contains the general audit trail information;

❷ A readable function name of the activity to audit;

❸ If this attribute is specified true, the attached audit data will be first stored in buffer memory prior to being flushed into the log storage when the `AuditLog` instance is disposed. Otherwise the `Flush` method need to explicitly invoked to write the audit data into storage; The configuration property *autoFlush* determines the default value;

❹ The ID which uniquely identifies the activity to audit. The value of *TransactionId* application context item is by default used;

❺ The account of the user to perform the activity to audit. The value of the *UsernName* application context item is by default used;

❻ The version of the running application. The version no of the executable assembly is by default used.

The properties related to the general audit data, which are passed into variety of constructors, are all defined into the class of `NCS.IConnect.AuditTrail.AuditTrailHeader`. The following code snippet represents its definition.

```
public class AuditTrailHeader
{
    public AuditTrailHeader();
    public AuditTrailHeader(string functionName);
    public AuditTrailHeader(string functionName, string transactionId, string userName);
    public AuditTrailHeader(string functionName, string transactionId, string userName, string
appVersion);
    public AuditTrailHeader(string functionName, string transactionId, string userName, string
appVersion,
        Dictionary<string, object> extendedProperties);

    public string AppVersion { get; set; }
    public Dictionary<string, object> ExtendedProperties { get; set; }
    public string FunctionName { get; set; }
    public string TransactionId { get; set; }
    public string UserName { get; set; }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

As aforementioned, the transaction, which is uniquely identified by *TransactionId*, is the basic audit unit. iConnect.NET adopts such a transaction identifying policy as below:

- If the ambient transaction (`System.Transactions.Transaction.Current`) does not exist, the value which is manually specified as the `TransactionID` of the current application context is used as the transaction ID. Otherwise:
- If distributed identifier (`System.Transactions.Transaction.Current.TransactionInformation.DistributedIdentifier`) is not empty, it is used as the transaction ID. Otherwise,
- The local identifier (`System.Transactions.Transaction.Current.TransactionInformation.LocalIdentifier`) is used as the transaction id.

14.3.3. Writing Audit Data with Triggers

If you adopt trigger based change data capture, you need to create source table specific triggers. Regardless of single table or multiple table based trigger, you can leverage the SQL generator hosted in iConnect.NET Studio to generate them.

If you adopt multiple table audit, you can use T4 based code generation to generate the given source tables specific triggers. As we did in the above demonstration, we just need to add a .tt file and is associated configuration file.

Two `TriggerGenerator` classes are defined in the following two namespaces: `NCS.IConnect.SqlGeneration.SqlServer` and `NCS.IConnect.SqlGeneration.Oracle`, which are separately used for SQL Server/Oracle specific trigger generation. You just need to instantiate a `TriggerGenerator` instance and invoke the `Run` method in the template file. The following is a demonstrative T4 template file for trigger generation.

```
<#@ template language="C#" hostSpecific="true" debug="true"   #>
<#@ output extension=".sql" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.SqlGeneration.dll"
#>
<#@ import namespace="NCS.IConnect.SqlGeneration.SqlServer" #>❶
<#@ include file="T4Toolbox.tt" #>
<#
   new AuditTableGenerator(this.Host.ResolvePath("Audittrail.Codegeneration.config"),
"iConnectDb","dbo",
                            "SourceTable1","SourceTable2","SourceTable3").Run();
#>
```

❶ The namespace of SQL Server specific `TriggerGenerator` class. For Oracle, you need to use `NCS.IConnect.SqlGeneration.Oracle` instead.

The following code snippet represents the definition of the constructors of the two `TriggerGenerator`.

```
//SQL Server:
public class TriggerGenerator : TriggerGeneratorBase
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
{
    public TriggerGenerator(string configurationFile, ❶
                            string connectionStringName, ❷
                            string schema, ❸
                            string schemaOfAuditTable❹,
                            params string[] tables)❺;
}
//Oracle
public class TriggerGenerator : TriggerGeneratorBase
{
    public TriggerGenerator(string configurationFile,
                            string connectionStringName,
                            string sourceOwner❻,
                            string OwnerOfAuditTable❼,
                            params string[] tables);
}
```

❶ The path of the configuration file;

❷ The name of the connection string configured in the above configuration file;

❸ The schema of the source table;

❹ The schema of audit table;

❺ The names of the source tables;

❻ The owner (user name) of the source database;

❼ The owner (user name) of the log database

When `TriggerGenerator` executes, it will read the specified configuration. In such a configuration, not only is the connection string needed to be configured, but also the system fields' column name should be specified. The column name of the six system fields: *CreatedBy*, *CreatedTime*, *VersionNo*, *TransactionId*, *LastUpdatedBy* and *LastUpdatedTime*. The following is a demonstrative configuration.

```
<configuration>
  <configSections>
    <section name="iConnect.dataAccess"
             type="NCS.IConnect.Helpers.Data.Configuration.DataAccessSettings,
                   NCS.IConnect.Helpers.Data"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  <iConnect.dataAccess>
    <systemFields>
      <add name="VersionNo"       fieldName="VERSION_NO"/>
      <add name="TransactionId"   fieldName="TRANSACTION_ID"/>
      <add name="CreatedBy"       fieldName="CREATED_BY"/>
      <add name="CreatedTime"     fieldName="CREATED_TIME"/>
      <add name="LastUpdatedBy"   fieldName="LAST_UPDATED_BY"/>
      <add name="LastUpdatedTime" fieldName="LAST_UPDATED_TIME"/>
    </systemFields>
  </iConnect.dataAccess>
</configuration>
```

Prior to executing the generated triggers, it must be guaranteed that the source tables specific audit tables exist. The DDL to create the audit tables can also the generated in the same manner. The following is a demonstrative T4 template to generate the given source tables

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

("SourceTable1", "SourceTable2", "SourceTable3") specific audit tables. It uses the configuration as same as the one used for trigger generation.

```
<#@ template language="C#" hostSpecific="true" debug="true"   #>
<#@ output extension=".sql" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.SqlGeneration.dll"
#>
<#@ import namespace="NCS.IConnect.SqlGeneration.SqlServer" #>
<#@ include file="T4Toolbox.tt" #>
<#
   new AuditTableGenerator(this.Host.ResolvePath("Audittrail.Codegeneration.config"), "iConnectDb",
                           "SourceTable1","SourceTable2","SourceTable3").Run();
#>
```

As the above code snippet represents, you just need to instantiate an `AuditTableGenerator`, and invoke the `Run` method in the body of T4 template. The following represents the definition of the `AuditTableGenerator`'s constructor.

```
SQL Server:
public class AuditTableGenerator : AuditTableGeneratorBase
{
    public AuditTableGenerator(string configurationFile,
                               string connectionStringName,
                               string schema,
                               params string[] tables);
}
Oracle:
public class AuditTableGenerator : AuditTableGeneratorBase
{
    public AuditTableGenerator(string configurationFile,
                               string connectionStringName,
                               params string[] tables);
}
```

Since all of the detail audit information is written into the storage by triggers, it is very simple to program. You just need to place the business logic related to the activity to audit in the using block of `AuditLog`. The following code snippet represents the typical programming.

```
using (AuditLog log1 = new AuditLog("function"))
{
    using (AuditLog log2 = new AuditLog("function"))
    {
        <<Business Logic>>
    }
}//Only here, audit data will be stored into storage.
```

14.3.4. Writing Audit Data with SQLCDC

In the previous versions, iConnect.NET Audit Trail gives us only one solution to capture data change: trigger. If the project uses SQL Server 2008 (Enterprise or Developer edition), we can directly leverage a new built-in feature to perform data change capture: CDC (Change Data Capture). SQLCDC, working as a SQL agent job, can be waked up when any data modification (Insert, Update and Delete) is performed against the table whose CDC feature is enabled.

Unlike trigger, all of the SQLCDC based change data capture is performed out of the scope of the current database transaction. This can greatly reduce the transaction duration, and bring lower latency and avoid dead lock. In order to facilitate SQLCDC based audit trail, we create

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

a series of SQL Generators to help you: enable CDC for database and source table; migrate the captured data into audit table.

If you want to transfer from triggers based audit to SQLCDC based, you need to use T4 based SQL generator to generate the follow two kinds of SQL script:

- Enable the database and given source tables CDC feature, and specify the columns of the source table, which need to be tracked by SQLCDC;
- Add new Job step into the given SQL Job to migrate the captured data into audit table.

The following is a demonstrative T4 template to generate the above two kinds of script into a single script file. After executing the scripts generated, the CDC feature of the given three source tables ("SourceTable1","SourceTable2", "SourceTable3") will be enabled. And three job steps, which are used to migrate the source table specific captured data into the corresponding audit table, will be inserted into the SQLCDC clean up job.

```
<#@ template language="C#" hostSpecific="true" debug="true" #>
<#@ output extension=".sql" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.SqlGeneration.dll"
#>
<#@ import namespace="NCS.IConnect.SqlGeneration" #>
<#@ include file="T4Toolbox.tt" #>
<#
   new EnableCdcGenerator(this.Host.ResolvePath("AuditTrail.CodeGeneration.config"),
                       "iConnectDb","iConnectDb","dbo","",true,
                       "SourceTable1","SourceTable2","SourceTable3").Run();
#>
```

As illustrated in the above code snippet, you just need to instantiate an `EnableCdcGenerator` instance and invoke the Run method in the T4 template file. The following code snippet represents the definition of `EnableCdcGenerator's` constructor.

```
public class EnableCdcGenerator : SqlGenerator
{
    public EnableCdcGenerator(string configPath❶,
                              string connectionStringName❷,
                              string destinationDatabaseName❸,
                              string destinationSchema❹,
                              string roleName❺,
                              bool supportsNetChanges❻,
                              params string[] tableNames❼);
}
```

❶ The path of the configuration file, which is similar to the trigger generator based one;

❷ The name of connection string configured in the above configuration file;

❸ The name of the database to store audit data;

❹ The schema of the audit table;

❺ The name of the database role used to gate access to change data. role_name is sysname and must be specified. If explicitly set to NULL, no gating role is used to limit access to the change data;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❻ Indicates whether support for querying for net changes is to be enabled for this capture instance. supports_net_changes is bit with a default of 1 if the table has a primary key or the table has a unique index that has been identified by using the @index_name parameter. Otherwise, the parameter defaults to 0;

❼ The source table name list.

When you transfer from trigger based audit to SQLCDC based, no revision need to be made to the programming.

14.3.5. Writing Audit Data with DataSet

We can consider the above two kinds of audit as automatic change data capture, which is performed at database level, and no managed code is needed. In the following sections, we will introduce three types of manual data change capture. The first one is `DataSet` based audit, which can be achieved by invoking the following Write method of AuditLog.

```
public class AuditLog
{
   //Others...
   public DataSetAuditTrailEntity(DataSet data, bool logChangeOnly);
}
```

The above `Write` method is invoked, the detail audit trail data will be extracted and formated, and then written into the log storage. The parameter *`logChangeOnly`* indicates whether just only the changed data row (the `DataRowState` is not *UnChanged)* need to be logged. The following code snippet illustrates programing audit trail with `DataSet`. Please note that only single table audit mode can be used in `DataSet` base audit.

```
using(AuditLog log = new AuditLog("function"))
{
    // Log the data changes in DataSet,
    // it'll return false if the provider doesn't support.
    log.Write(dataset, true);
}
```

14.3.6. Writing Audit Data with DbCommand

Besides the `DataSet` object, you can also perform audit trail against a `DbCommand` instance. There is another `Write` overload method accepting a `DbCommand` instance as its first parameter. The following code snippet represents this method's definition. The second parameter indicates the result to execute this `DbCommand`.

```
public class AuditLog
{
   //Others...
   public bool Write(DbCommand command, object result);
}
```

Once the above method is invoked, the `DbCommand` related information, including command text and parameter list. Combining the result data manually passed in the method, this information will be formated string, whose schema is like this. Such a formated literal string will be finally written into the log storage once the `Flush` method is invoked.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<Commnad Text>,<Parameter1 Name>=<Value> <Parameter2 Name>=<Value> ... ,<Command Result>
```

As the following code snippet illustrated, the `DbCommand` audit based programming is similar to the `DataSet` based one. Just single table audit is supported.

```
using(AuditLog log = new AuditLog("function"))
{
    // it'll return false if the provider doesn't support.
    log.Write(command, result);
}
```

14.3.7. Writing Audit Action

The most straightforward audit trail is to directly write a raw audit trail data. We call this type of audit as action based audit, which can be achieved by invoking the following `Write` method. The three parameters separately represents: the name of the activity to audit and related descriptive and remark information.

```
public class AuditLog
{
   //Others...
   public bool Write(string action, string description, string remark);
}
```

The action audit is implemented in a special `StorageProvider`: `NCS.IConnect.AuditTrail.ActionStorageProvider`. You must guarantee one `ActionStorageProvider` is configured accordingly when you program action base audit. The following code snippet represents the audit trail configuration with an `ActionStorageProvider`.

```
<iConnect.auditTrail autoFlush="true">
  <storageProviders>
     <add name="actionProvider"
          type="NCS.IConnect.AuditTrail.ActionStorageProvider,
                NCS.IConnect.AuditTrail"/>
          applicationName="Sample"
          database="iConnectDb" />
    ...
 </storageProviders>
</iconnect.auditTrail>
```

The action based audit use a single table *T_IC_AUDIT_ACTIONS* to store the action related information.

14.4. Querying Audit Data

AuditLog class provides static `GetLogSummary` and `GetLogDetail` method to retrieve audit trail data using storage provider.

```
DataSet ds = AuditLog.GetLogSummary(startDate, endDate);
DataSet ds = AuditLog.GetLogDetail(tableName,logId);
```

For the invocation to the `GetLogSummary` and `GetLogDetail` methods, please notes:

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- For `SingleTableStorageProvider` and `MultipleTableStorageProvider`, `GetLogSummary` is to retrieve the main record in *T_IC_AUDIT_LOG* and *T_IC_AUDIT_LOG_DATA*.
- For `ActionStorageProvider`, `GetLogSummary` is to retrieve the record in *T_IC_AUDIT_LOG* and *T_IC_AUDIT_ACTIONS*
- It only makes sense to call `GetLogDetail` for `MultipleTableStorageProvider`. `GetLogDetail` will return the detail data changes stored in the backup tables.

Please note both of `GetLogSummary` and `GetLogDetail` always return a weakly typed `DataSet`. Similar to the `Write` methods, `GetLogSummary` also will call all providers and all data returned by different providers will be merged into one `DataSet`. So the data retrieved by `SingleTableStorageProvider` will put into "T_IC_AUDIT_LOG_QUERY",(`MultipleTableStorageProvider` also uses it), but the data retrieved by `ActionStorageProvider` will put into "T_IC_AUDIT_ACTIONS_QUERY".

If you just want to retrieve data with one specified storage provider, for example, you want to retrieve action audit data written by `ActionStorageProvider`:

```
AuditTrailSearchCriteria entity = new AuditTrailSearchCriteria();
entity.StartDate = DateTime.Now.AddMonths(-1);
entity.EndDate = DateTime.Now.AddDays(1);

AuditLog.RetrievingData onRetrievingData = delegate(HandleProviderEventArgs e)
{
    e.Cancel = !(e.Provider is ActionStorageProvider);
};

DataSet actual = AuditLog.GetLogSummary(entity, onRetrievingData);
```

14.5. Archiving Audit Data

`AuditLog` class provides static `Archive` method to delete the useless audit data.

```
// Archives log data logged before archiveDate
AuditLog.Archive(archiveDate);

// Archives log data logged before between fromDate and toDate
AuditLog.Archive(fromDate, toDate);
```

Current solution only deletes the data, but you can customize the stored procedure to move data into another table or database. Like `Write` method, you can write the code to archive audit data with one specified storage provider as follows:

```
AuditLog.Archive(fromDate, toDate,
    delegate(HandleProviderEventArgs e)
    {
        e.Cancel = !(e.Provider is ActionStorageProvider);
    }
);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 15. Hierarchy

15.1. Introduction

A hierarchical data model is a data model in which the data is organized into a tree-like structure. The structure allows representing information using parent/child relationships: each parent can have many children but each child only has one parent (also known as a 1: many ratio). All attributes of a specific record are listed under an entity type. The iConnect.NET hierarchy component is used to maintain these hierarchy data and let developer easily extend special businesses logic.

15.2. Getting Started

This section gives you a demonstrative sample application to guide you program Hierarchy, including how to build the hierarchical data, and how to bind the data to a `TreeView`.

Step I: Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first need to execute the following SQL script in this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*HierarchyDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3B9413F6-3644-4733-8CD6-DA08B203D751'
           ,'HierarchyDemo'
           ,'hierarchydemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Step II: Create an ASP.NET Application and Configure Hierarchy

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Create an empty ASP.NET web application using your Visual Studio, and add the references to the following two assemblies, which are located in the /bin folder of the iConnect.NET installation directory.

- NCS.IConnect.Core.dll;
- NCS.IConnect.Hierarchy.dll;

Then copy the following configuration snippet to override the existing content of the we.config, and do not forget to revise the connection string to target at the appropriate database. This configuration includes not only the Hierarchy related configuration but also Application Context related configuration, which is mandatory.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="iConnect.applicationContexts"
             type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                   NCS.IConnect.Core"/>
    <section name="iConnect.hierarchy"
             type="NCS.IConnect.Hierarchy.Configuration.HierarchySettings,
                   NCS.IConnect.Hierarchy"/>
  </configSections>

  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..." />
  </connectionStrings>
  <iConnect.applicationContexts defaultContextLocator="HttpSessionStateContextLocator">
    <contextLocators>
      <add name="HttpSessionStateContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.HttpSessionStateContextLocator,
                 NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>
  <iConnect.hierarchy>
    <storageProviders>
      <add name="dbProvider"
           type="NCS.IConnect.Hierarchy.DbStorageProvider,
                 NCS.IConnect.Hierarchy"
           applicationName="HierarchyDemo" database="iConnectDb" />
    </storageProviders>
  </iConnect.hierarchy>
</configuration>
```

Step III: Building and Binding Hierarchical Data

iConnect.NET Hierarchy is leveraged by this demonstrative application to build the hierarchical data of an organization, whose own the following tree-like structure:

```
NCSI
    NCSI-SZ
        3041
        3042
        3043
    NCSI-CD
    NCSI-SH
```

Finally we add a Web Form and name it Default.aspx. In this page, only one `TreeView` web control named *treeOrg* is added. The following code snippet represents the HTML of this page's body.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:TreeView runat="server" ID="treeOrg" />
    </div>
    </form>
</body>
```

The organization is built when the page is loaded for the first time, and the built hierarchical organization node is also bound to the `TreeView` when page is loaded. The following code snippet represents the whole code behind of the *Default.aspx*.

```
public partial class Default : System.Web.UI.Page
{
    private HierarchyManager HierarchyManagerManager
    {
        get{return this.ViewState["HierarchyManager"] as HierarchyManager;}
        set{this.ViewState["HierarchyManager"] = value;}
    }
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!this.IsPostBack)
        {
            this.HierarchyManagerManager = new HierarchyManager("Org", "dbProvider");
            if (!this.HierarchyManagerManager.HasNode())
            {
                this.HierarchyManagerManager.AddRootNode("NCSI");
                this.HierarchyManagerManager.AddNode("NCSI-SZ", "NCSI");
                this.HierarchyManagerManager.AddNode("NCSI-CD", "NCSI");
                this.HierarchyManagerManager.AddNode("NCSI-SH", "NCSI");
                this.HierarchyManagerManager.AddNode("3041", "NCSI-SZ");
                this.HierarchyManagerManager.AddNode("3042", "NCSI-SZ");
                this.HierarchyManagerManager.AddNode("3043", "NCSI-SZ");
                this.HierarchyManagerManager.Save();
            }
            var binding = new TreeNodeBinding();
            binding.TextField       = "Code";
            binding.ValueField      = "Code";
            binding.ToolTipField    = "Description";
            this.treeOrg.DataBindings.Add(binding);
            this.treeOrg.DataSource = this.HierarchyManagerManager.GetNodes();
            this.treeOrg.DataBind();
        }
    }
}
```

Run this application, you will see the organization structure is displayed in this page.

15.3. `HierarchyNode`

iConnect.NET Hierarchy is a lightweight framework used to build and maintain tree-like hierarchal data structure. In the application programming interface, the `HierarchyNode<TExtension>` class is used to represent a node in the tree. Considering extensibility that Hierarchy can be extended to cater for domain specific hierarchical data structure, E.G. Organization, this type is defined as a generic one. As illustrated in the following code snippet, the generic type `TExtension` must inherits from `NodeExtension`. Its property Extension of `TExtension` type is used to hold the domain specific properties. The `NodeExtension` just only a "blank" type.

```
[Serializable]
public class HierarchyNode<TExtension> : IHierarchyData where TExtension: NodeExtension, new()
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
{
    //Others
    TExtension Extension{get;}
}
[Serializable]
public abstract class NodeExtension
{
    public NodeExtension();
}
```

In iConnect.NET Hierarchy, a hierarchy node always exists as a part of a hierarchy tree. In other word, once initialized, a hierarchy node is added into a hierarchy tree. There are some properties or methods defined in `HierarchyNode<TExtension>` to the given hierarchy node's root node, parent node and child nodes, or to check if it is a leaf node. We can call them navigation properties/methods. The definition of such navigation members is represented in the following code snippet.

```
[Serializable]
public class HierarchyNode<TExtension> : IHierarchyData where TExtension: NodeExtension, new()
{
    //Others
    public HierarchyNode<TExtension> RootNode { get; }
    public HierarchyNode<TExtension> ParentNode { get; }
    public HierarchyNodeCollection<TExtension> ChildNodes { get; }
    public bool HasChildren { get; }
}
```

Regardless which concrete domain a hierarchy node is specific to, they share a set of common properties, which are directly defined in the `HierarchyNode<TExtension>` type. The following code snippet represents the definition of these common properties.

```
[Serializable]
public class HierarchyNode<TExtension> : IHierarchyData where TExtension: NodeExtension, new()
{
    //Others
    public string Alias { get; set; }❶
    public string Category { get; }❷
    public string Code { get; set; }❸
    public int Depth { get; }❹
    public string Description { get; set; }❺
    public DateTime EffectiveStartDate { get; set; }
    public DateTime EffectiveEndDate { get; set; }❻
    public string Flag { get; set; }❼
    public string Path { get; }❽
    public string Remarks { get; set; }❾
    public int Sequence { get; set; }❿
}
```

❶ The alias of the hierarchy node;

❷ The category of the hierarchy node. Each category corresponds to a given hierarchy tree. In other words, a hierarchy tree is composed of a group of hierarchy nodes of the same category;

❸ The hierarchy node's code which can be considered as a unique identifier;

❹ The depth of the path from the current node up to the root node;

❺ The descriptive information of the hierarchy node;

❻ The start/end date of the effective period if the hierarchy node is not effective all the time. By default, there is no effectiveness policy applied to the hierarchy node;

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❼ The hierarchy node's flag, which can be used to hold any customized information;

❽ The path from the root node to the current node;

❾ The remark information of the hierarchy node;

❿ The hierarchy node's sequence, which is usually used to control the display order of the level of hierarchy nodes;

Except for `HierarchyNode<TExtension>`, its specific collection type, `HierarchyNodeCollection<TExtension>`, is also defined in iConnect.NET Hierarchy. The following code snippet represents the definition of `HierarchyNodeCollection<TExtension>`. Because it implements the `IHierarchicalEnumerable` interface, we can directly bind a `HierarchyNodeCollection<TExtension>` object to a `TreeView`.

```
[Serializable]
public class HierarchyNodeCollection<TExtension> : IEnumerable<HierarchyNode<TExtension>>,
    IHierarchicalEnumerable, IEnumerable where TExtension: NodeExtension, new()
{
    public HierarchyNodeCollection();
    public IEnumerator<HierarchyNode<TExtension>> GetEnumerator();
    public IHierarchyData GetHierarchyData(object enumeratedItem);
    IEnumerator IEnumerable.GetEnumerator();
    public int Count { get; }
}
```

15.4. StorageProvider and HierarchyManager

As well most of iConnect.NET components, provider based designed is applied to Hierarchy. The hierarchy data retrieval and save are implemented in a given StorageProvider. There are two kinds of StorageProvider: `DbStorageProvider` and `WcfStorageProvider`. `DbStorageProvider` is used at the service side in distributed application or non-distributed application. The following code snippet represents the `DbStorageProvider` specific configuration.

```
<configuration>
  <connectionStrings>
    <add name="iConnectDb" connectionString="..." />
  </connectionStrings>
  <iConnect.hierarchy>
    <storageProviders>
      <add name="dbProvider"
           type="NCS.IConnect.Hierarchy.DbStorageProvider,
                 NCS.IConnect.Hierarchy"
           applicationName="HierarchyDemo"
           database="iConnectDb" />
    </storageProviders>
  </iConnect.hierarchy>
</configuration>
```

As with `WcfStorageProvider` based configuration, we just only need to specify the configuration name of the endpoint to invocate the hierarchy storage service and name of provider at service side. The following XML snippet represents the configuration of `WcfStorageProvider`.

```
<configuration>
  <system.serviceModel>
    <client>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
      <endpoint name="hierarchySvc"
                address="http://localhost/ICQuickStartService/Hierarchy.svc"
                binding="wsHttpBinding"
                contract="NCS.IConnect.Hierarchy.IHierarchyStorageService"
                .../>
       </client>
  </system.serviceModel>
  <iConnect.hierarchy>
    <storageProviders>
      <add name="wcfStorageProvider"
           type="NCS.IConnect.Hierarchy.WcfStorageProvider,
                 NCS.IConnect.Hierarchy"
           endpoint="hierarchySvc" serverProviderName="dbProvider"/>
    </storageProviders>
  </iConnect.hierarchy>
</configuration>
```

Unlike other iConnect.NET components with provider based design, the default provider cannot be configured. The reason for this is that more than one domain specific `StorageProvider` are all used in most applications. So when we instantiate the facade `HierarchyManager`, the configuration name of the `StorageProvider` must be explicitly specified. In addition, more than one hierarchy trees are maintained in database. In order to determine which tree a given hierarchy node is belong to, we classify the hierarchy node to a particular category. In order words, each category corresponds to a hierarchy tree, and a hierarchy tree is built of the hierarchy nodes of the same category.

A `HierarchyManager` is used to maintain a given hierarchy tree. When we instantiate a `HierarchyManager` by calling its constructor, both `StorageProvider` name and category should be specified. The definition of `HierarchyManager`'s constructor is represented in the following code snippet.

```
[Serializable]
public class HierarchyManager : HierarchyManagerBase<NilNodeExtension>
{
    public HierarchyManager(string category, string providerName);
}
[Serializable]
public abstract class HierarchyManagerBase<TNodeExtesion> where TNodeExtesion: NodeExtension, new()
{
    //Others
}
[Serializable]
public class NilNodeExtension : NodeExtension
{
    // Methods
    public NilNodeExtension();
}
```

As the matter of fact, the above code snippet represents the definition of the whole `HierarchyManager`. Because all of the functionaries are implemented in its base class, `HierarchyManagerBase<TNodeExtesion>`, no member is defined in `HierarchyManager`.

15.5. Hierarchy Maintenance

When `HierarchyManager` is initialized, all of the hierarchy nodes of the given category will be retrieved by the given `StorageProvider` and built as a hierarchy tree, which is attached to the `HierarchyManager`. You can also call the `Load` method to reload the hierarchy data and

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

rebuild the hierarchy tree. A series of methods exposed from HierarchyManager (actually defined in `HierarchyManagerBase<TNodeExtesion>`) help you maintain the built tree. E.G. You can add a child node under an existing node, or remove an existing node (including all nodes beneath it) from the tree or to another node.

All of these operations are all performed against the hierarchy tree in memory, so we must call the following Save method to submit the change to database. The `HasChanges` method returns a boolean value indicating if the hierarchy tree has been changed since it is loaded.

```
public abstract class HierarchyManagerBase<TNodeExtesion> where TNodeExtesion: NodeExtension, new()
{
    //Others
    public void Save();
    public virtual bool HasChanges();
    public void Load();
}
```

15.5.1. Add a New Hierarchy Node

If we need to add an new child node for an existing node, we can call the `AddNode` method of HierarchyManager. We just only specify the code of the hierarchy node to add and the parent node. If the loaded hierarchy tree is empty, which can be determined by calling the `HasNode` method, the `AddRootNode` can be called to add a root hierarchy node of the specified code. The definition of the mentioned three methods is repented in the following code snippet.

```
public abstract class HierarchyManagerBase<TNodeExtesion> where TNodeExtesion: NodeExtension, new()
{
    //Others
    public HierarchyNode<TNodeExtesion> AddNode(string code, HierarchyNode<TNodeExtesion> parentNode);
    public HierarchyNode<TNodeExtesion> AddNode(string code, string parentCode);
    public HierarchyNode<TNodeExtesion> AddRootNode(string code);
    public bool HasNode();
}
```

The `AddNode` overload with a *`parentNode`* parameter is used in the scenario in which you get the parent node in advanced. As with how to get hierarchy node(s), the following methods can be used. `GetNodeByCode` method returns the hierarchy node of the specified code. `GetNodes` enable you of specifying an evaluation expression filter the retrieved hierarchy nodes. If the optional parameter is not explicitly specified, the returned node collection only contains the root node. The root node can be directly returned by calling the `GetRootNode` method. For the `GetNodeByCode` and `GetRootNode` methods, `HierarchyException` will be thrown if no matched hierarchy node can be found.

```
public abstract class HierarchyManagerBase<TNodeExtesion> where TNodeExtesion: NodeExtension, new()
{
    //Others
    public HierarchyNode<TNodeExtesion> GetNodeByCode(string code);
    public HierarchyNodeCollection<TNodeExtesion> GetNodes(Predicate<HierarchyNode<TNodeExtesion>>
predicate = null);
    public HierarchyNode<TNodeExtesion> GetRootNode();
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

15.5.2. Remove an Existing Node from Hierarchy Tree

The following two `RemoveNode` methods enable you to remove an existing hierarchy node, including all nodes beneath it, from the hierarchy tree. You can specify the code of the hierarchy node to remove, or directly the hierarchy node itself. A `HierarchyException` will be thrown if the hierarchy node does not exist the hierarchy tree attached to the `HierarchyManager`.

```
public abstract class HierarchyManagerBase<TNodeExtesion> where TNodeExtesion: NodeExtension, new()
{
    //Others
    public void RemoveNode(HierarchyNode<TNodeExtesion> hierarchyNode);
    public void RemoveNode(string code);
}
```

15.5.3. Remove an Exiting Node to another Node

If you need to remove a particular node, including all nodes beneath it, to another node, you can call the following `MoveTo` methods. Both hierarchy node to remove and target node (or node's code) must be specified.

```
public abstract class HierarchyManagerBase<TNodeExtesion> where TNodeExtesion: NodeExtension, new()
{
    //Others
    public void MoveTo(HierarchyNode<TNodeExtesion> hierarchyNode, HierarchyNode<TNodeExtesion>
targetNode);
    public void MoveTo(HierarchyNode<TNodeExtesion> hierarchyNode, string targetCode);
}
```

15.6. System Parameter

In general, application level setting can be defined in configuration file. However, configuration file based system setting is static and need to be manually maintained. In some situation, we need a dynamical setting mechanism. For example, in order to make the administrator easily manage the system settings, a special visual console is built to maintain some predefined system parameters. In addition, the setting need to be take effect immediately once a particular system parameter is changed. *System Parameter*, which extends the core Hierarchy, is just such a lightweight framework for dynamical system settings. The System Parameter specific assembly is NCS.IConnect.Hierarchy.Parameter.dll.

In the application programming interface, a system parameter is represented by a `SystemParameter`. As illustrated in the follow code snippet, a `SystemParameter` has the similar property definition to aforementioned hierarchy node. As the matter of fact, `SystemParameter` directly encapsulates a hierarchy node. You can get the inner hierarchy node via the Node property. The Name&Value property actually corresponds to the code and description of hierarchy node.

```
[Serializable]
public class SystemParameter
{
    public string Name { get; }
    public string Alias { get; set; }
    public string Value { get; set; }
    public string Flag { get; set; }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public string Remarks { get; set; }
    public HierarchyNode<NilNodeExtension> Node { get; }
}
```

System Parameter specific hierarchical data structure is a tree of only two levels including root node. Each`SystemParameter` correspond to a particular leaf node. System Parameter has its specific facade type: `ParameterManager`. Because all of the system parameter nodes are belong to a predefined category (*IconnectSystemParameter*), only the `HierarchyProvider`'s name is specified when instantiating a `ParameterManager`. The following code snippet represents the definition of `ParameterManager`.

```
[Serializable]
public class ParameterManager
{
    public ParameterManager(string providerName);

    public SystemParameter CreateParameter(string name, string value);
    public bool ParameterExists(string name);
    public SystemParameter GetParameter(string name);
    public string GetParameterValue(string name);
    public void DeleteParameter(string name);

    public bool HasChanges();
    public void Load();
    public void Save();

    public IHierarchicalEnumerable DataSource { get; }
    public HierarchyManager HierarchyManager { get; }
}
```

You can create a new `SystemParameter` by calling the `CreateParameter` method and delete an existing `CreateParameter` by calling the `DeleteParameter` method. An `ArgumentException` will be thrown if the `SystemParameter` to add exists or the `SystemParameter` to delete does not exist. The `GetParameter` and `GetParameterValue` can be called to get a given `SystemParameter` or its value. An `ArgumentException` will be also thrown if no matched An `SystemParameter` can be found. In order to avoid throwing exception, you can call the `ParameterExists` method to check the existence of a `SystemParameter` prior to calling these methods.

The three methods `HasChanges`, `Load` and `Save` are directly borrowed from `HierachyManager`. In fact, the aforementioned methods are all finally implemented by calling a `HierachyManager`, which can be got via the HierarchyManager property. As with DataSource, it returns a `HierarchyNodeCollection<TNodeExtesion>` including all system parameter specific hierarchy nodes.

15.7. Organization Management

As well as System Parameter, Organization Manage is also a domain oriented extended Hierarchy. As it name suggestion, Organization Manage is used to manage hierarchical organization structure, including the users belong to a given organization. Organization Management specific assembly is NCS.IConnect.Hierarchy.Organization.dll.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

15.7.1. `OrganizationUser` and `OrganizationExtension`

In iConnect.NET Hierarchy application programming interface, the `OrganizationUser` is used to represent a user belong to a particular organization. The definition of `OrganizationUser` is represented in the following code snippet.

```
[Serializable]
public class OrganizationUser
{
    public string UserName { get; }❶
    public string DelegateTo { get; set; }❷
    public string ReportTo { get; set; }❸
    public HierarchyNode<OrganizationExtension> Node { get; }❹
}
```

❶ The user name;

❷ The name of the user to whom the user delegates to.

❸ The name of the user to whom the user directly reports to;

❹ The hierarchy node of the organization to which the user belongs to.

The organization based hierarchy node is represented by a `HierarchyNode<OrganizationExtension>` object. The organization specific properties, which are not defined in `HierarchyNode<TNodeExtension>`, are all defined in `OrganizationExtension`. As the following code snippet illustrates, there is only one Users property, which represents all of users.

```
[Serializable]
public class OrganizationExtension : NodeExtension, IEnumerable<OrganizationUser>, IEnumerable
{
    public IList<OrganizationUser> Users { get; private set; }
}
```

15.7.2. `OrganizationManager`

`OrganizationManager` is facade type of Organization Management. As illustrated in the following code snippet, `OrganizationManager` directly inherits from `HierarchyManagerBase<OrganizationExtension>`. You can leverage the application programing interface inherited from base class to manage the organization hierarchy tree. Extra methods defined in `OrganizationManager` are used to add user into a given organization, or remove a user from a particular organization.

```
[Serializable]
public class OrganizationManager : HierarchyManagerBase<OrganizationExtension>
{
    public OrganizationManager(string category, string providerName);
    public OrganizationUser AddUser(HierarchyNode<OrganizationExtension> node,
        string userName, string reportTo, string delegateTo);
    public OrganizationUser GetUser(HierarchyNode<OrganizationExtension> node, string userName);
    public void RemoveUser(OrganizationUser user);
}
```

`OrganizationManager` has its own database based `StorageProvider`, `NCS.IConnect.Hierarchy.Organization.DbStorageProvider` (not

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`NCS.IConnect.Hierarchy.DbStorageProvider`). Before consuming functionalities of `OrganizationManager`, you must make user appropriate `StorageProvider` is configured. Its configuration is as below.

```
<?xml version="1.0"?>
<configuration>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..." />
  </connectionStrings>
  <iConnect.hierarchy>
    <storageProviders>
      <add name="orgDbProvider"
        type="NCS.IConnect.Hierarchy.Organization.DbStorageProvider,
        NCS.IConnect.Hierarchy.Organization"
        applicationName="HierarchyDemo"
        database="iConnectDb" />
    </storageProviders>
  </iConnect.hierarchy>
</configuration>
```

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 16. States

16.1. Introduction

The iConnect.NET State component enhances the ASP.NET Session object. It provides strongly typed access to the objects saved in the Session. It also enables the Session data to be split into groups. Each group has its own expiry policy.

As we known, Session variables are stored in a `SessionStateItemCollection` object through the `HttpContext.Session` property. The Session variable can be accessed through its name or an integer index. The value returned from Session variable must be cast to the appropriate type. Therefore the iConnect.NET States provides a configuration to define user-friendly name and data type for necessary Session variables and then generate strongly typed states through iConnect.NET code generation. It's similar to ASP.NET profile.

The default session mode is *InProc*, which stores he session state as “living” objects in the same process as the web requests. Once the SessionState item is added, it will always be kept in the memory until it is manually removed or session timeout. If you use State Server mode or SQL Server mode the serialization/de-serialization, data transfer between web server and state/SQL server can affect performance. Therefore the iConnect.NET State component provides a configurable backing store to move the idle large data from web server to storage.

16.2. Getting Started

This section gives you a demonstrative sample application to guide you program iConnect.NET States using code generation.

Step I: Create a ASP.NET Web Application

You first create a new empty ASP.NET Web application using your Visual Studio, and add the necessary references to the following assemblies, which are located in the /bin directory of iConnect.NET installation directory.

- NCS.IConnect.Core.dll;
- NCS.IConnect.Common.dll;
- NCS.IConnect.State.dll;

Step II Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

need to execute the following SQL script against the this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*StatesDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3B9413F6-3644-4733-8CD6-DA08B203D751'
           ,'StatesDemo'
           ,'statesdemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Step III: Define State Structure

This demonstrative application is used to maintain employee information, which is classified as two groups: Profile information and HR information. We use an XML file to define two groups of state items. So in the newly created web application, we need to add an XML file name it "States", and copy the following XML snippet into this file.

```
<?xml version="1.0" encoding="utf-8" ?>
<state timeout="00:10:00" minimumByteSize="99999">
  <group name="Profile" scope="">
    <add name="FirstName" type="System.String" />
    <add name="LastName"  type="System.String" />
    <add name="Age"  type="int?" />
    <add name="Address"  type="System.String" />
  </group>
  <group name="HrInfo"  timeout="00:00:10" minimumByteSize="1024" scope="">
    <add name="EnrollmentDate" type="System.DateTime?" />
    <add name="Grade"  type="System.String" />
    <add name="ReportTo"  type="System.String" />
    <add name="AnualLeave"  type="int?" />
  </group>
</state>
```

Step IV: Generate Code

Code generation of iConnect.NET States is based on T4, so we need to add a new T4 template file in the project. We right click the project in the solution explorer, and choose "Add/New item" context menu item. From the installed project templates in the "Add new item" dialog box, select the "General" template. Finally choose the "Text Template" from the general item list. Name the newly added text template file "States.tt", the copy the following code snippet

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

to override the existing content in this file. Please not forget to revise the assembly references if iConnect.NET is not installed in the default directory (C:\Program Files).

```
<#@ template  language="C#" hostSpecific="true" debug="true" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Common.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.State.dll" #>
<#@ import namespace="NCS.IConnect.State.CodeGeneration" #>
<#@ import namespace="NCS.IConnect.Common.CodeGeneration" #>
<#@ import namespace="NCS.IConnect.Core.CodeGeneration" #>
<#@ import namespace="System.CodeDom.Compiler" #>
<#@ include file="T4Toolbox.tt" #>
<#@ output extension="cs" #>
<#
   var codeGenerator = new StateGenerator(
        this.Host.ResolvePath("States.xml"),
        Languages.CSharp,
        new CodeGeneratorOptions(),
        "NCS.IConnect.State");
   codeGenerator.Run();
#>
```

Now, any attempt to save the template file will trigger the T4 engine to generate the code in the dependent file *States.cs*. Code generation can also be triggered by "Run Custom Tool". The template file specific "Custom Tool" will run when you right click the file in the solution explorer and choose the "Run Custom Tool" item in the context menu. The following code snippet represents the structure of the generated C# code.

```
namespace NCS.IConnect.State
{
    public class PageBase : NCS.IConnect.Architecture.Client.PageBase
    {
        public virtual ExtendedRootStateNode State {get;}
    }
    public sealed partial class ExtendedRootStateNode : RootStateNode {

        public ProfileGroupStateNode Profile {get;}
        public HrInfoGroupStateNode HrInfo {get;}
        ...
    }
    public sealed class ProfileGroupStateNode : GroupStateNode
    {
        ...
        public string FirstName {get; set;}
        public string LastName{get; set;}
        public int? Age{get; set;}
        public string Address{get; set;}
    }
    public sealed class HrInfoGroupStateNode : GroupStateNode
    {
        ...
        public System.DateTime? EnrollmentDate {get; set;}
        public string Grade{get; set;}
        public string ReportTo{get; set;}
        public int? AnualLeave{get; set;}
    }
}
```

Step V: Programming iConnect.NET States

We add a *Global.asax* for this application, and copy the following code snippet to override the existing content.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
using NCS.IConnect.State;
namespace StatesDemo
{
    public class Global : StateAwareApplication
    {
        public override RootStateNode CreateStateNode()
        {
            return new ExtendedRootStateNode();
        }
    }
}
```

Add then copy the following XML snippet as the configuration of web.config. Do not forget specify the connection string to connect to appropriate database.

```
<configuration>
 <configSections>
  <section name="iConnect.state"
           type="NCS.IConnect.State.Configuration.StateSettings,
                 NCS.IConnect.State"/>
 </configSections>
 <connectionStrings>
  <add name="iConnectDb"
       connectionString="..."
       providerName="System.Data.SqlClient"/>
 </connectionStrings>
 <iConnect.state defaultBackingStore="dbBackingStore" scavengingInterval="00:00:10">
  <backingStores>
   <add name="dbBackingStore"
        type="NCS.IConnect.State.DbBackingStore,
              NCS.IConnect.State"
       applicationName="StatesDemo"
       database="iConnectDb"/>
  </backingStores>
 </iConnect.state>
</configuration>
```

Finally, we create a web form, and name it Default.aspx. In order to make programming more easy, we need to revise the code behind and make the page inherit the generated `PageBase` type.

```
namespace StatesDemo
{
    public partial class Default : PageBase
    {
       //...
    }
}
```

The web page is used to main the employee information, which is defined in the above XML file and related C# code has also been generated. Now we design the web page according to the following figure.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

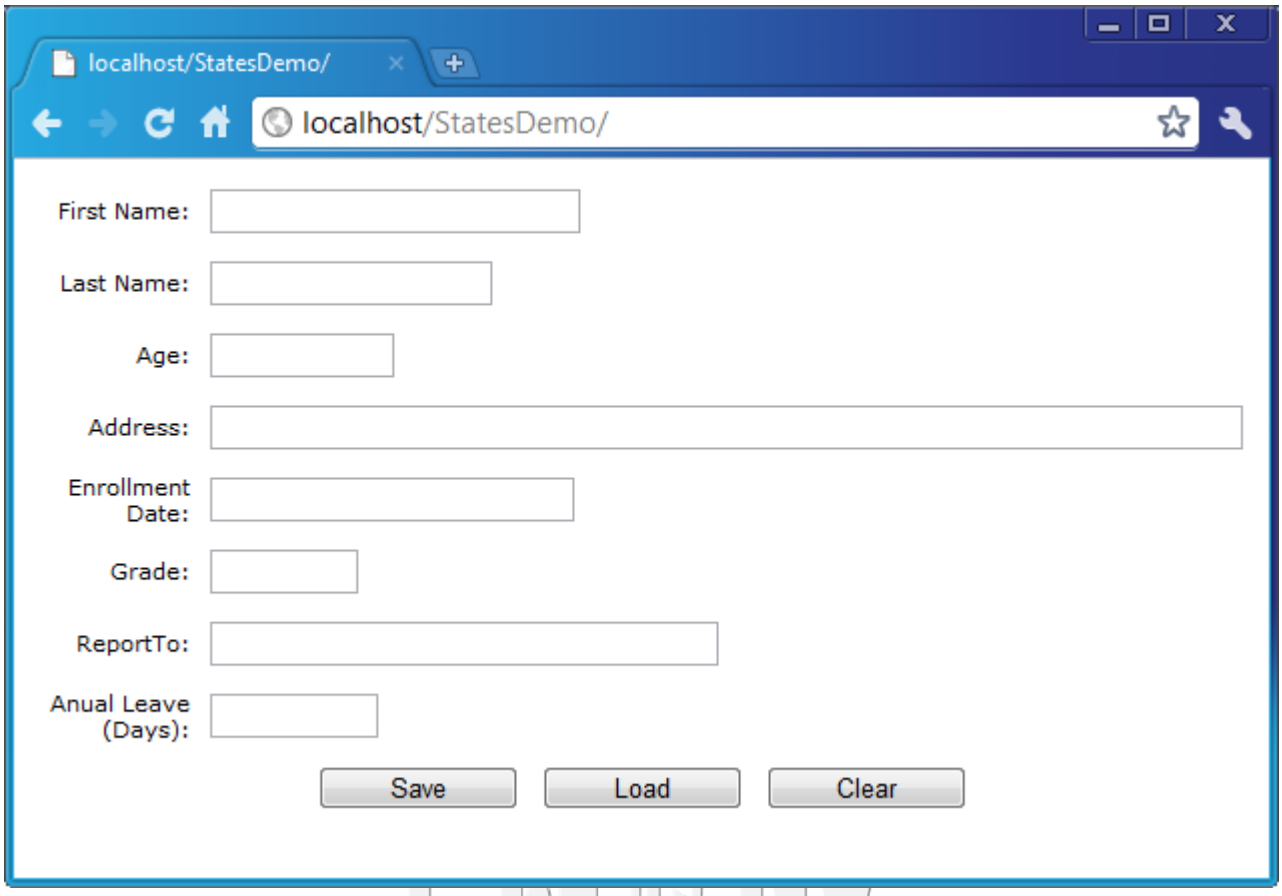


Figure 16.1. States Demonstrative Web Page

When the Save button is clicked, all of the information entered into the `TextBox` are captured and saved into *States*. The following code snippet represents the Save button's click event handler.

```
protected void ButtonSave_Click(object sender, EventArgs e)
{
    this.State.Profile.FirstName = this.TextBoxFirstName.Text;
    this.State.Profile.LastName = this.TextBoxLastName.Text;
    if (!string.IsNullOrEmpty(this.TextBoxAge.Text))
    { this.State.Profile.Age = int.Parse(this.TextBoxAge.Text); }
    else
    {
        this.State.Profile.Age = null;
    }
    this.State.Profile.Address = this.TextBoxAddress.Text;

    if (!string.IsNullOrEmpty(this.TextBoxEnrollmentDate.Text))
    {
        this.State.HrInfo.EnrollmentDate = DateTime.Parse(this.TextBoxEnrollmentDate.Text);
    }
    else
    {
        this.State.HrInfo.EnrollmentDate = null;
    }
    this.State.HrInfo.ReportTo = this.TextBoxReportTo.Text;
    this.State.HrInfo.Grade = this.TextBoxGrade.Text;
    if (!string.IsNullOrEmpty(this.TextBoxAnualLeave.Text))
    {
        this.State.HrInfo.AnualLeave = int.Parse(this.TextBoxAnualLeave.Text);
    }
    else
    {
        this.State.HrInfo.AnualLeave = null;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    }
}
```

User can click the Load button to retrieve the employee information from *States* and display them in the page. The following code snippet represents the Load button's click event handler.

```
protected void ButtonLoad_Click(object sender, EventArgs e)
{
    this.TextBoxFirstName.Text = this.State.Profile.FirstName;
    this.TextBoxLastName.Text = this.State.Profile.LastName;
    this.TextBoxAge.Text = this.State.Profile.Age == null ? string.Empty :
this.State.Profile.Age.ToString();
    this.TextBoxAddress.Text = this.State.Profile.Address;

    this.TextBoxEnrollmentDate.Text = this.State.HrInfo.EnrollmentDate == null ? string.Empty :
this.State.HrInfo.EnrollmentDate.ToString();
    this.TextBoxReportTo.Text = this.State.HrInfo.ReportTo;
    this.TextBoxGrade.Text = this.State.HrInfo.Grade;
    this.TextBoxAnualLeave.Text = this.State.HrInfo.AnualLeave == null ? string.Empty :
this.State.HrInfo.AnualLeave.ToString();
}
```

The `Clear` button is used to clear all of the `TextBox` in the page. Its click event handler is represented in the following code snippet.

```
protected void ButtonClear_Click(object sender, EventArgs e)
{
    this.TextBoxFirstName.Text = string.Empty;
    this.TextBoxLastName.Text = string.Empty;
    this.TextBoxAge.Text =string.Empty;
    this.TextBoxAddress.Text = string.Empty;

    this.TextBoxEnrollmentDate.Text = string.Empty;
    this.TextBoxReportTo.Text = string.Empty;
    this.TextBoxGrade.Text = string.Empty;
    this.TextBoxAnualLeave.Text = string.Empty;
}
```

So far you can run this application to maintain the employee information which is stored in iConnect.NET States.

16.3. How to Define State Items

All of the predefined state items are defined in an XML file, from which the States code generator get all hierarchical state items to generate C# or VB.NET code. The following code snippet represents a typical XML of state item definition. From the XML schema we can see that a state item, which has a unique name, can defined in the root node (<state/>) or a group node (<group/>). The names of the state items in the root node or same group node must be unique. However, the ones under different group can share the same name. The data type of the state item is defined via *`type`* attribute.

```
<?xml version="1.0" encoding="utf-8" ?>
<state timeout="00:10:00" minimumByteSize="99999" >
  <group name="Group1" timeout="00:10:00" minimumByteSize="1024" scope="~/Group1/" >
    <add name="Item1" type="System.String" />
    <add name="Item2"  type="int?" />
  </group>
  <group name="Group2" timeout="00:20:00" minimumByteSize="2048" scope="~/Group2/">
    <add name="Item1" type="System.String" />
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    <add name="Item2"  type="int?" />
  </group>
  <add name="Item1" type="System.String" />
  <add name="Item2"  type="int?" />
</state>
```

16.4. Expiration Policy

Backing storing is the core functionality of iConnect.NET States, which remove the expired SessionState item into the backing storage. The expiration policy is directly defined in the aforementioned XML to define the predefined state items. iConnect.NET State determines if a given state item is active or expires based on the following policy:

- Expiration policy is specific to state item group, other than to the state item itself;
- Each state item group has a property of MinimumByteSize of integer type. If the size of the serialized state item in the group smaller than MinimumByteSize, the state items in this group are alway active;
- Each state item group has a property of Scope, which is a path relative to the root directory. This property stands for the scope of a given state item group. For the Scope of state item group, we can specify a directory (E.G. ~/HR/) or a path of an .aspx page (E.G. ~/HR/Default.aspx). You can also specify more than one path like "~/HR/, ~/PM/". For current request path, if it is not in the scope of the state item group's scope, the state items in these groups are all expired. This scope is an optional property, and it means that the scope is the root directory of the web application if it is not explicitly specified;
- Each state item group has a property of *Timeout*, which stands for a duration between the latest activity time and the time when all of state items will expire. The current system time will be recorded as the latest activity time of the state item group once any state item in the group is accessed.
- Both MinimumByteSize and Timeout can be explicitly specified to a given state item group. If not explicitly specified, it will inherit the ones specified in the root node (<state/>). The two properties for the root node are both mandatory.

16.5. State Item Backing Storing

For each state item expiration check, the expired state items will be stored into the backing storage. iConnect.NET provides a built-in database based backing storing mechanism, which is implemented in `DbBackingStore`. You can also create your own custom `BackingStore` which inherits `NCS.IConnect.State.IBackingStore`. Which type of `BackingStore` is used is determined by configuration. As the following XML snippets illustrates, you can configure more than one `BackingStore`, and the default one must be explicitly specified.

```
<configuration>
 <configSections>
  <section name="iConnect.state"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
          type="NCS.IConnect.State.Configuration.StateSettings,
                NCS.IConnect.State"/>
 </configSections>
 <connectionStrings>
  <add name="iConnectDb"
       connectionString="..."
       providerName="System.Data.SqlClient"/>
 </connectionStrings>
 <iConnect.state defaultBackingStore="dbBackingStore"❶
                 scavengingInterval="00:00:10">
  <backingStores>
   <add name="dbBackingStore"
        type="NCS.IConnect.State.DbBackingStore,
              NCS.IConnect.State"
       applicationName="StatesDemo"
       database="iConnectDb"/>❷
  </backingStores>
 </iConnect.state>
</configuration>
```

❶ The configuration name of default `BackingStore`;

❷ The `DbBackingStore` specific configuration element, including the connection string name of backing storing database, and the application name;

The state item expiration checking and followed backing storing is triggered by each HTTP request arrival. However, considering the performance, we specify an interval between two check events in succession. It means the latest check time will be recorded, and the current request specific expiration checking will be ignored if the duration between this time and the current time is not greater than the interval. Such a interval is specified via *`scavengingInterval`* property of <iConnect.state/> configuration section. If *`scavengingInterval`* is an optional property and the default value is *00:00:30* (30 seconds).

16.6. Code Generation

We always program States using the code which generated by iConnect.NET States code generation, which is based on T4. States based code is generated by NCS.IConnect.State.CodeGeneration.StateGenerator, which can read the XML file defining state items and generate matched C# or VB.NET code. You just only instantiate the StateGenerator, and invoke Run method in T4 template. The following code snippet represents such a typical T4 template.

```
<#@ template  language="C#" hostSpecific="true" debug="true" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Common.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.Core.dll" #>
<#@ assembly  name="C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.State.dll" #>
<#@ import namespace="NCS.IConnect.State.CodeGeneration" #>
<#@ import namespace="NCS.IConnect.Common.CodeGeneration" #>
<#@ import namespace="NCS.IConnect.Core.CodeGeneration" #>
<#@ import namespace="System.CodeDom.Compiler" #>
<#@ include file="T4Toolbox.tt" #>
<#@ output extension="cs" #>
<#
   var codeGenerator = new StateGenerator(
                           this.Host.ResolvePath("States.xml")❶,
                           Languages.CSharp❷,
                           new CodeGeneratorOptions()❸,
                           "NCS.IConnect.State"❹);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
   codeGenerator.Run();
#>
```

❶ The path of the XML file to define the hierarchical state items;

❷ The programing language for the generated code, both C# and VB.NET are supported;

❸ A `CodeGeneratorOptions` object to control the style of the generated code;

❹ The namespace in which the generated classes are defined in.

If we perform code generation against the XML file in the section 3.1, the generated code will be like the following code snippet. Each state item group has its corresponding class (Group1GroupStateNode, Group2GroupStateNode), which inherits `GroupStateNode`. As with the state item group specific class, each of its properties corresponds to a particular state item. The root node corresponds to the generated `ExtendedRootStateNode` class, which inherits `RootStateNode`. There is a matched property in `ExtendedRootStateNode` for all state items and groups. Except for these, a `PageBase` class is also generated, and it inherits NCS.IConnect.Architecture.Client.PageBase. `PageBase` has a read-only State property of `ExtendedRootStateNode` type. So when you make your web page inherit the generated `PageBase`, you can program iConnect.NET States in a strongly typed fashion.

```
namespace NCS.IConnect.State {

    public class PageBase : NCS.IConnect.Architecture.Client.PageBase
    {
        public virtual ExtendedRootStateNode State {get;}
    }
    public sealed partial class ExtendedRootStateNode : RootStateNode {

        public Group1GroupStateNode Group1 {get;}
        public Group2GroupStateNode Group2 {get;}
        public string Item1 {get; set;}
        public int?   Item2 {get; set;}
        ...
    }
    public sealed class Group1GroupStateNode : GroupStateNode
    {
        ...
        public string Item1 {get; set;}
        public int?   Item2 {get; set;}
    }
    public sealed class Group2GroupStateNode : GroupStateNode
    {
        ...
        public string Item1 {get; set;}
        public int?   Item2 {get; set;}
    }
}
```

iConnect.NET States achieves state item expiration checking and followed backing storing by registering `HttpApplication`'s `PostRequestHandlerExecute` event. And such an event registration is implemented in `NCS.IConnect.State.StateAwareApplication`, which is a custom `HttpApplication`. So you must create a Global.asax, and make Global inherit `StateAwareApplication`, and override the `CreateStateNode` to return an `ExtendedRootStateNode` instance.

```
public class Global : StateAwareApplication
{
   public override RootStateNode CreateStateNode()
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
   {
      return new ExtendedRootStateNode();
   }
}
```

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 17. Session Management

17.1. Introduction

iConnect.NET Session Management is irrelevant to ASP.NET Session, and is not used to persist session specific state. The main functionality provided by iConnect.NET Session Management is session tracking. Leveraging this component, the administrator can track all of the sessions, including their login user name, IP address, Host Name, Last Activity Time, etc. Expiration mechanism is provided to prevent the user from consuming your application once session timeout (it is not ASP.NET Session Timeout). For administrative reason, a particular active session can also be killed by administrator. In addition, you can use Session Management to control the count of concurrent sessions specific to a single user account. Session Management can be both used in Windows application and ASP.NET web application.

17.2. Getting Started

This section gives you a demonstrative sample application to guide you program Session Management, including:

- How to start a new session once successfully login;
- How to automatically renew the current session and keep it active;
- How to terminate the active session when logout;
- How to list all active sessions;

Step I: Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first need to execute the following SQL script against the this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*SessionMgtDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            ('3B9413F6-3644-4733-8CD6-DA08B203D751'
            ,'SessionMgtDemo'
            ,'sessionmgtdemo'
            ,''
            ,''
            ,1
            ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
            ,'Sys'
            ,GETDATE()
            ,'Sys'
            ,GETDATE())
GO
```

Step II: Create an ASP.NET application and Configure Session Manage

Create an empty ASP.NET application and name it "SessionMgtDemo". Add the reference the following assemblies, which are located in the /bin folder of iConnect.NET installation directory.

- NCS.IConnect.Security.dll;
- NCS.IConnect.Security.Server.dll;
- NCS.IConnect.SessionManagement.dll;

And then the copy the following XML snippet to override the existing content of web.config. This XML snippet includes the configuration of Application Context, Membership and Session Management. Do not forget to revise the connection string to connect to your appropriate database.

```
<configuration>
 <configSections>
    <section name="iConnect.applicationContexts"
             type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                   NCS.IConnect.Core,"/>
    <section name="iConnect.sessionManagement"
             type="NCS.IConnect.SessionManagement.Configuration.SessionManagementSettings,
                   NCS.IConnect.SessionManagement"/>
 </configSections>
  <iConnect.applicationContexts defaultContextLocator="HttpSessionStateContextLocator">
    <contextLocators>
       <add name="HttpSessionStateContextLocator"
            type="NCS.IConnect.Core.ApplicationContexts.HttpSessionStateContextLocator,
                  NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>
 <connectionStrings>
  <add name="iConnectDb"
       connectionString="..."
       providerName="System.Data.SqlClient"/>
 </connectionStrings>
 <iConnect.sessionManagement defaultProvider="DbProvider"
                             defaultScavenger="backgroundScavenger">
  <providers>
   <add name="DbProvider"
        type="NCS.IConnect.SessionManagement.DbSessionProvider,
              NCS.IConnect.SessionManagement"
       database="iConnectDb"
       applicationName="SessionMgtDemo"
       maxConcurrentSessions="1"
       thresholdBreakingBehavior="BlockNewAttempt"
       timeout="00:03:00"
       renewalRefreshInterval="00:00:05"/>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  </providers>
  <scavengers>
   <add name="backgroundScavenger"
        type="NCS.IConnect.SessionManagement.BackgroundScavenger,
              NCS.IConnect.SessionManagement"
        database="iConnectDb"
        applicationName="SessionMgtDemo"
        scavengingWorkInterval="00:05:00"
        scavengingWindow="00:10:00"/>
  </scavengers>
 </iConnect.sessionManagement>
 <system.web>
  <membership defaultProvider="DbMembershipProvider">
   <providers>
    <add name="DbMembershipProvider"
         applicationName="SessionMgtDemo"
         database="iConnectDb"
         type="NCS.IConnect.Security.DbMembershipProvider,
               NCS.IConnect.Security.Server"/>
   </providers>
  </membership>
  <authentication mode="Forms">
   <forms defaultUrl="~/Default.aspx" loginUrl="~/Login.aspx"/>
  </authentication>
 </system.web>
</configuration>
```

Step III Register `PostAcquireRequestState` event for Session Renewal

We create a Global.asax into the project, and register `HttpApplication`'s `PostAcquireRequestState` event to automatically renew the current session and keep it active (The session's last activity time will reflect the latest request). The following code snippet represents the definition of Global.asax.

```
public class Global : System.Web.HttpApplication
{
    protected void Application_PostAcquireRequestState(object sender, EventArgs e)
    {
        if (!this.User.Identity.IsAuthenticated)
        {
            return;
        }

        if (null == HttpContext.Current.Session)
        {
            return;
        }
        string sessionID = HttpContext.Current.Session["SessionID"] as string;
        if (null != sessionID)
        {
            SessionManager.Renew(sessionID);
        }
    }
}
```

Step IV: Create Three Pages: Default.aspx, Login.aspx, and Register.aspx

In the web application, we will create three web pages. All of the active sessions will be listed in the Default.aspx, and the Login.aspx, and Register.aspx are separately used for login and user account registration.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

In the Default.aspx, only two web controls are added. One of the two is a `GridView` to list all retrieved sessions, and the other is a `LoginStatus` indicating the current login status and triggering login/logout. The following XML snippet represents the HTML of the page's body.

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:GridView ID="GridViewCurrentSessions" runat="server" AutoGenerateColumns="false">
            <Columns>
                <asp:BoundField DataField="SessionId" HeaderText="Session ID" />
                <asp:BoundField DataField="UserName"  HeaderText="User Name" />
                <asp:BoundField DataField="IPAddress" HeaderText="IP Address" />
                <asp:BoundField DataField="HostName"  HeaderText="Host Name" />
                <asp:BoundField DataField="StartTime" HeaderText="StartTime" />
                <asp:BoundField DataField="LastActivityTime" HeaderText="Last Activity Time" />
            </Columns>
        </asp:GridView>
        <asp:LoginStatus ID="LoginStatus1" runat="server" onloggedout="LoginStatus1_LoggedOut" />
    </div>
    </form>
</body>
```

When page is loaded, all of the active session list are bound to the `GridView`. The session retrieval is completed by calling the `GetSessionInfoList` method of the static `SessionManager`.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        this.GridViewCurrentSessions.DataSource = SessionManager.GetSessionInfoList(new
SessionCriteria{ Status= SessionStatus.Active});
        this.GridViewCurrentSessions.DataBind();
    }
}
```

Once logout, the current session need to be normally terminated. So we register the `LoginStatus`'s `LoggedOut` event and the Terminate method of `SessionManager` to terminate the current session.

```
protected void LoginStatus1_LoggedOut(object sender, EventArgs e)
{
    string sessionID = HttpContext.Current.Session["SessionID"] as string;
    if (null != sessionID)
    {
        SessionManager.Terminate(sessionID);
    }
}
```

There is only one `CreateUserWizard` control in the Register.aspx. The following XML snippet represents the HTML of the page's body. We set the ContinueDestinationPageUrl property to navigate to Login.aspx once registration completes, and set the *`LoginCreatedUser`* as false to prevent the registered user automatically signing in.

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:CreateUserWizard ID="CreateUserWizard1" runat="server"
ContinueDestinationPageUrl="~/Login.aspx"
            LoginCreatedUser="False" />
    </div>
    </form>
</body>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

As with *Login.aspx*, we place only one `Login` control. The following XML snippet represents the HTML of the page's body. We set the CreateUserUrl and CreateUserText to add a register link to the `Register.aspx`.

```
<body>
    <form id="form1" runat="server">
    <div>
        <asp:Login ID="Login1" runat="server" OnAuthenticate="Login1_Authenticate"
CreateUserUrl="~/Register.aspx"
            CreateUserText="Register">
        </asp:Login>
    </div>
    </form>
</body>
```

We register the Authenticate event to perform manual authentication and session initialization. As the following code snippet illustrates, the Start method of `SessionManager` is used to initialize the new session, and session information and session creation status is returned. The returned Session ID is stored in the session state.

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
{
    if (!ExtendedMembership.ValidateUser(Login1.UserName, Login1.Password))
    {
        Login1.FailureText = ExtendedMembership.GetValidateUserErrorInfo(Login1.UserName,
Login1.Password);
        return;
    }
    SessionInfo sessionInfo;
    SessionCreateStatus status;
    if (SessionManager.Start(Login1.UserName, out sessionInfo, out status))
    {
        this.Session["SessionInfo"] = sessionInfo;
        FormsAuthentication.RedirectFromLoginPage(Login1.UserName, false);
        this.Session["SessionID"] = sessionInfo.SessionId;
    }
    else
    {
        if (status == SessionCreateStatus.ExceedConcurrentThrehold)
        {
            Login1.FailureText = "Exceed the configured maximum concurrent threshold per user.";
        }
        else
        {
            Login1.FailureText = "Login failed. Please contact with administrator.";
        }
    }
}
```

Step V: Run the Demonstrative Application

You run this application and navigate to Register.aspx to register two user accounts, E.G. Alice & Bob. Then sign in the application using the user account Alice and navigate to Default.aspx. You will see your current session is listed as shown in the following screen shot.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

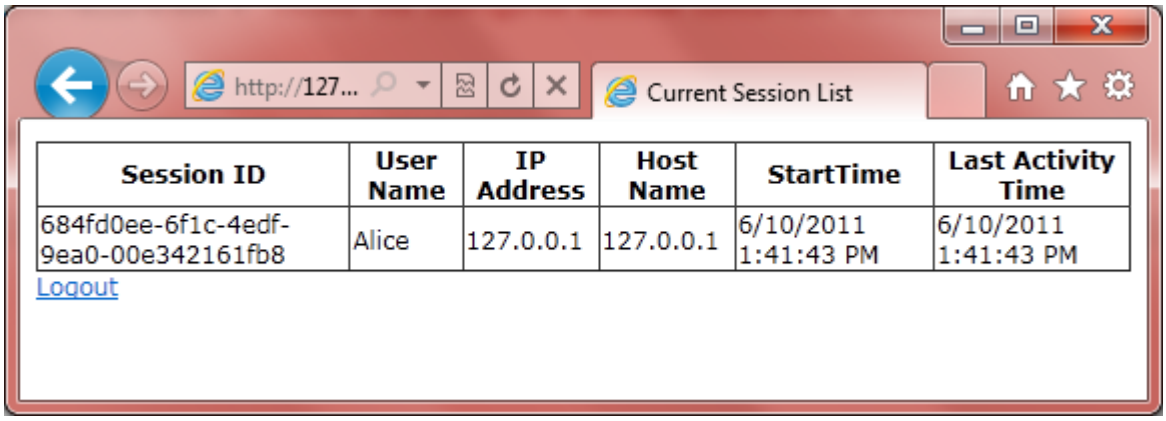


Figure 17.1. Active Session List

In Session Management configuration, we set the *`maxConcurrentSessions`* of the default `SessionProvider` as 1, which means there is only one session for the same user account. In order to test this, we keep the current user login, we sign in the application using the same user account (Alice). You will get the following error indicating "Exceed the configured maximum concurrent threshold per user." However, you can sign in our application using another user account, E.G. Bob.

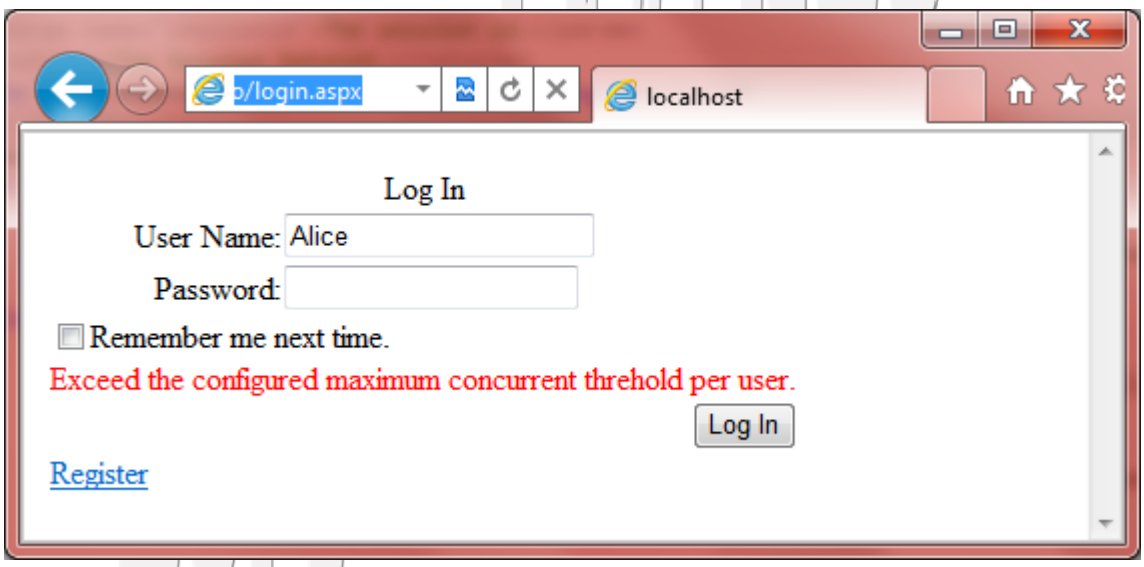


Figure 17.2. Exceeding Maximum Concurrent Session Error

17.3. `SessionInfo` and `SessionInfoCollection`

In iConnect.NET Session Management application programming interface, the session is represented by a `NCS.IConnect.SessionManagement.SessionInfo` object. The following code snippet represents the definition of `SessionInfo`.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class SessionInfo
{
    public SessionInfo(string sessionId, string userName, DateTime startTime, DateTime
lastActivityTime, string hostName, string ipAddress, SessionStatus status);
    [DataMember]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public string HostName { get;}❶
    [DataMember]
    public string IPAddress { get;}❷
    [DataMember]
    public DateTime LastActivityTime { get;}❸
    [DataMember]
    public string SessionId { get;}❹
    [DataMember]
    public DateTime StartTime { get;}❺
    [DataMember]
    public SessionStatus Status { get;}❻
    [DataMember]
    public string UserName { get;}❼
}
```

❶ The host name of client machine;

❷ The IP address of client machine;

❸ The last activity time. In general, it reflects the last HTTP request time for web application, whereas it is last mouse or keyboard operation time for GUI application;

❹ The ID which uniquely identifies a given session;

❺ The time the session is started;

❻ The status of the session;

❼ The login user name.

Each runtime session has a given status, including Active, Inactive, Killed and Terminated. The status of session is represented by the `NCS.IConnect.SessionManagement.SessionStatus` enumeration, and its definition is represented in the following code snippet.

```
public enum SessionStatus
{
    Unknown,
    Active,
    Inactive,
    Killed,
    Terminated
}
```

Except for SessionInfo, its specific collection type, `SessionInfoCollection` is also defined in Session Management application programming interface.

17.4. SessionProvider

As well as other iConnect.NET modules or components, provider based designed is still adopted in Session Management, and all of session management related functionalities are implemented in `SessionProvider`. There are two built-in SessionProvider defined, `DbSessionProvider` and `RemoteSessionProvider`.

As its name suggests, `DbSessionProvider` manages sessions by maintaining the session data stored in database. It is used in service side in distributed application or non-distributed application, and the `DbSessionProvider` specific configuration is illustrated in the following configuration snippet.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<configuration>
 <connectionStrings>
  <add name="iConnectDb"
       connectionString="..."
       providerName="System.Data.SqlClient"/>
 </connectionStrings>
 <iConnect.sessionManagement defaultProvider="DbProvider" ...>
  <providers>
   <add name="DbProvider"
        type="NCS.IConnect.SessionManagement.DbSessionProvider,
              NCS.IConnect.SessionManagement"
       database="iConnectDb"
       applicationName="SessionMgtDemo" .../>
  </providers>
  ...
 </iConnect.sessionManagement>
</configuration>
```

The `RemoteSessionProvider` provides session management functionalities by invoking WCF service. You just need to specify the configuration name of the endpoint to invocate the session management service. The `RemoteSessionProvider` can be configured as below.

```
<configuration>
   <iConnect.sessionManagement defaultProvider="remoteSessionProvider">
    <providers>
      <add name="remoteSessionProvider"
           type="NCS.IConnect.SessionManagement.RemoteSessionProvider,
                 NCS.IConnect.SessionManagement"
           endpoint="SessionManagement" />
    </providers>
  </iConnect.sessionManagement>
  <system.serviceModel>
  <client>
   <endpoint name="SessionManagement"
             address="http://localhost/SessionMgtDemo/SessionManagement.svc"
             binding="wsHttpBinding"
             contract="NCS.IConnect.SessionManagement.ISessionService"
             .../>
  </client>
 </system.serviceModel>
</configuration>
```

17.5. Starting Sessions

As the above Getting-Started application demonstrates, we can invoke the `Start` of the static facade `SessionManager` to start and initialize a new session. The following code snippet represents the definition of Start method.

```
public static class SessionManager
{
    //Others
    public static bool Start(string userName, out SessionInfo sessionInfo, out SessionCreateStatus
status);
}
```

Generally speaking, we start a new session after each successful login. Once login succeeds, we call the Start method with the authenticated user name is passed. If session is successfully started, the method returns true, otherwise returns false. Besides the sucess/failure indicator is returned, there are two extra objects are returned in the form of output parameter. The first is a SessionInfo representing the current session, and the other is a `SessionCreateStatus` enumeration representing the status of current session creation operation. As the following

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

code snippet illustrates, there are totally three status: Success, ExceedConcurrentThreshold and ProviderError.

```
public enum SessionCreateStatus
{
    Success,❶
    ExceedConcurrentThreshold,❷
    ProviderError❸
}
```

❶ The session is successfully started;

❷ The count of concurrent sessions specific to the given user account exceeds the allowed maximum value;

❸ Exception is thrown from `SessionProvider`.

Once session is successfully started by invoking the Start method, the `SessionStarted` event of `SessionManager` will be raised. You can register this event to perform some post-session-start works. The following code snippet represents the definition of `SessionStarted` event and corresponding `EventArgs`: `SessionEventArgs`, from whose SessionInfo you can get the session information.

```
public static class SessionManager
{
    //Others
    public static EventHandler<SessionEventArgs> SessionStarted;
}
public class SessionEventArgs : EventArgs
{
    public SessionInfo SessionInfo { get; }
}
```

17.6. Concurrent Session Throttling

For some applications, there is such a requirement that just only one concurrent session for a given user account is allowed. iConnect.NET Session Management make you can freely set the maximum concurrent sessions. By default, for a particular session starting attempt by calling the aforementioned Start method, the output parameter status will return ExceedConcurrentThreshold once exceeds the configured threshold.

For the default current session throttling behavior, Session Management gives you another one: The new session is normally started and the oldest active session will be killed. The two kinds of threshold breaking behaviors are defined in the following `ThresholdBreakingBehavior` enumeration.

```
public enum ThresholdBreakingBehavior
{
    KillExistingSession,
    BlockNewAttempt
}
```

The two concurrent session throttling properties, maximum concurrent sessions and threshold breaking behavior, are configured to concrete `SessionProvider`. In the following configuration

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

snippet, the maximum concurrent sessions and threshold breaking behavior of `DbSessionProvider` are separately specified as 5 and KillExistingSession.

```
<?xml version="1.0"?>
<configuration>
 <iConnect.sessionManagement defaultProvider="DbProvider" ...>
  <providers>
   <add name="DbProvider"
        type="NCS.IConnect.SessionManagement.DbSessionProvider,
              NCS.IConnect.SessionManagement
       maxConcurrentSessions="5"
       thresholdBreakingBehavior="KillExistingSession" .../>
  </providers>
  ...
 </iConnect.sessionManagement>
</configuration>
```

Note

Considering security, the both two properties, MaxConcurrentSessions and ThresholdBreakingBehavior, are defined in `DbSessionProvider` other than `RemoteSessionProvider`, because they should be configured at server side.

17.7. Session Expiration and Renewal

Session has a timeout period, and it is active only when the duration between last activity time and current time is lower than the timeout period. In ideal situation, session's last activity time should precisely reflect the last HTTP request time for web application or last mouse/keyboard operation for GUI application. In other words, session renewal will be triggered by each HTTP request or mouse/keyboard operation. Session renewal is performed by calling the `Renew` method of `SessionManager`. As the following code snippet illustrates, the Renew method has only one parameter of *`sessionId`* represents the ID of the session to renew. Session renewal just only specific to active sessions. If the session to renew is inactive, terminated or killed, a `SessionException` will be thrown.

```
public static class SessionManager
{
    //Others
    public static void Renew(string sessionId);
}
```

In order to make the session is automatically renewed, we usually register `HttpApplication`'s event to call the above Renew method for ASP.NET application. Because this method accepts the session ID as parameter, we can stored the session ID into session state after session starts. In this situation, it must be guaranteed that the session state is available when the registered event is fired. According to this principal, the following events of `HttpApplication` can be registered: `PostAcquireRequestState`, `PreRequestHandlerExecute`, `PostRequestHandlerExecute`. In the getting-started demonstrative application, what we registered is `PostRequestHandlerExecute`.

As with GUI application, in order to keep the session's last activity time consistent with the latest mouse/keyboard operation, we have to register all controls's MouseMove/KeyDown

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

event. There is a static `RegisterControlEvent` method defined in `SessionManager`, and you can it to perform batch `MouseMove` event registration. When this method is invoked, the given event handler will be registered all of the specified controls and their nested controls.

```
public static class SessionManager
{
    //Otheres
   public static void RegisterControlEvent(Control.ControlCollection collection, MouseEventHandler
mouseEventHandler);
}
```

Considering HTTP request or mouse move may be every frequent, and database access even WCF service invocation will be involved in each session renewal, it is not impossible to really renew the current session for each HTTP request or mouse move. As the matter of fact, not all real session renewal (get access or invocate service) happens for each invocation to the `Renew` method of `SessionManager`. There is an minimum interval between two real session renewals in succession, and we name it *RenewalRefreshInterval*. As well as session timeout, such an interval is also configured to `DbSessionProvider`. In the following configuration snippet, the session timeout and RenewalRefreshInterval are separately specified as 3 minutes and 5 seconds.

```
<?xml version="1.0"?>
<configuration>
 <iConnect.sessionManagement defaultProvider="DbProvider" ...>
  <providers>
   <add name="DbProvider"
        type="NCS.IConnect.SessionManagement.DbSessionProvider,
              NCS.IConnect.SessionManagement"
       timeout="00:03:00"
       renewalRefreshInterval="00:00:05" .../>
  </providers>
  ...
 </iConnect.sessionManagement>
</configuration>
```

Generally speaking, we will catch the `SessionException`, which has a Status property as below. If detecting the renewed session is Inactive, Terminated or Killed, re-login is necessary.

```
[Serializable]
public class SessionException : Exception
{
   //Others
    public SessionStatus Status { get;}
}
```

Each successful session renewal always raise the `SessionRenewed` event, which is represented in the following code snippet. You can register this event to conduct some post-session-renewal tasks.

```
public static class SessionManager
{
    //Others
    public static EventHandler<SessionEventArgs> SessionRenewed;
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

17.8. Terminating Sessions

In general, session termination is performed when the end user signs out of application. Once terminated, the status of the session will be transferred to Terminated. If the `Renew` method of `SessionManager` is called to renew a terminated session, a `SessionException` will be thrown. Session termination can be done by invoking the `Terminate` method of `SessionManager`, whose definition is represented in the following code snippet.

```
public static class SessionManager
{
    //Otheres
    public static void Terminate(string sessionId);
}
```

There is also a post-session-termination event, `SessionTerminated`, which is raised after a session is successfully be terminated. The following code snippet represents the definition of this event.

```
public static class SessionManager
{
    //Others
    public static EventHandler<SessionEventArgs> SessionTerminated;
}
```

17.9. Killing Sessions

For some administrative reasons, E.G. Relieving heavy load, the administrator of appropriate privileges can kill some sessions. One killed, the status of the session will be changed to Killed, and a `SessionException` will be thrown when renewing a killed session. Killing sessions can be done by invoking the `Kill` method of the `SessionManager`, whose definition is represented in the following code snippet.

```
public static class SessionManager
{
    //Otheres
    public static void Kill(List<string> sessionIds);
    public static void Kill(string sessionId);
}
```

As well as the aforementioned three session actions, a post event named `SessionKilled` is also defined in `SessionManager`. Its definition is represented in the following code snippet.

```
public static class SessionManager
{
    //Others
    public static EventHandler<SessionEventArgs> SessionKIlled;
}
```

17.10. Scavenging Sessions

There is always a session record in database for any started sessions. Large scale of records will greatly influent the database access performance, so a scavenging mechanism,which cleans

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

away the useless session records, is very necessary. In iConnect.NET Session Management, session scavenging is completed in a component of `SessionScavenger`.

A built-in `SessionScavenger`, `BackgroundScavenger`, is provided. As its name suggests, `BackgroundScavenger` which works in a background thread, scavenges the useless session records at a given intervals. It is worth mentioning that the useless session to scavenge is not the expired ones. If the session record is deleted at the moment when it expires, we cannot determine it status since there is no corresponding record. We give `BackgroundScavengera` special property, ScavengingWindow, which is a duration between the last activity time and the time when it can be scavenged. ScavengingWindow must be greater than session's timeout period.

The ScavengingWindow and interval to carry out session scavenging are both configurable. The following XML snippet represents the configuration of `BackgroundScavenger`.

```
<configuration>
 <iConnect.sessionManagement defaultProvider="DbProvider"
                             defaultScavenger="backgroundScavenger">
  <scavengers>
   ...
   <add name="backgroundScavenger"
        type="NCS.IConnect.SessionManagement.BackgroundScavenger,
              NCS.IConnect.SessionManagement"
        database="iConnectDb"
        applicationName="SessionMgtDemo"
        scavengingWorkInterval="00:05:00"
        scavengingWindow="00:10:00"/>
  </scavengers>
 </iConnect.sessionManagement>
</configuration>
```

17.11. Retrieve Session(s)

You can get a give session or session list based on different search criteria by calling the following `GetSessionInfo`/`GetSessionInfoList`.

```
public static class SessionManager
{
    //Others
    public static SessionInfo GetSessionInfo(string sessionId);❶
    public static SessionInfoCollection GetSessionInfoList(SessionCriteria criteria);❷
    public static SessionInfoCollection GetSessionInfoList(string userName);❸
}
```

❶ Get a given session based on the specified session ID;

❷ Get sessions based on a `SessionCriteria`, whose definition is represented in the following code snippet:

```
public class SessionCriteria
{
    public List<string> HostNames { get; set; }
    public List<string> IPAddresses { get; set; }
    public DateTime LastActivityTimeFrom { get; set; }
    public DateTime LastActivityTimeTill { get; set; }
    public List<string> SessionIds { get; set; }
    public DateTime StartTimeFrom { get; set; }
    public DateTime StartTimeTill { get; set; }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public SessionStatus Status { get; set; }
    public List<string> UserNames { get; set; }
}
```

❸ Get the given user specific sessions;

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 18. Security

18.1. Introduction

Security is a broad topic. Research has shown that early design of authentication and authorization eliminates a high percentage of application vulnerabilities. If you are a sophisticated programmer, I believe you have written much code to implement security based functionalities for variety of applications, these functionalities include:

- **User Management**: User Creation, Deletion, Maintenance and Authentication.
- **Password Management:** Password Encryption, Retrieval and Reset and Password based policy management.
- **Role Management and Role-based Authorization:** Role Creation and Deletion, Adding user(s) to role(s).

If you have made any comparisons between the security implements across different applications, you will find they are all similar. If you adopt iConnect.NET as your development framework, you will never need to rewrite such duplicate code any more. iConnect.NET Security, which provides you with a series of security related functionalities and best practices, relies you from tedious repeat programming work. Besides the general security related functionalities mentioned above, iConnect.NET Security component gives you some additional functionalities as well:

- **Personality and Profiling:** You can customize the profiles for any users, even anonymous users.
- **Site Map and Menu:** Site map provides you with a consistent way to perform navigation in the whole site. You can also authorize user against each menu item.

In this version of iConnect.NET all of the functions have been aligned and improved against Microsoft .NET Framework built-in security functions (especially ASP.NET Membership & Role) and *Enterprise Library Security Application Block*. Comparing to Microsoft provisions, you can benefit more from iConnect.NET Security as below:

- **Bigger Spectrum of Data Storage Support:** The two mainstream relational databases, SQL Server and Oracle, are both supported.
- **Distributed Architecture Support:** You can leverage the WCF, to make iConnect.NET Security shipped in a distributed scenario.
- **Business Action based Authorization:** iConnect.NET Authorization does not only provide **Rule** based authorization, which is directly borrowed from Enterprise Library Security Block, but provides an additional **Business Action**-based Authorization as well.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- **Database Storage based Site Map:** Comparing with ASP.NET Site Map implementation, which stores all of the site map notes in configuration files, iConnect.NET Site Map approaches database as the persistent storage.

The Security component contains the following major modules:

- Membership
- Role
- Site map
- Profile
- Authorization

18.2. Membership

18.2.1. Introduction

Generally speaking, authentication is the first barrier of security system. Armed with iConnect.NET, you can use iConnect..NET membership to authenticate the login user in a terse and elegant fashion. Except for the basic authentication related functionalities, iConnect.NET membership provides two additional categories of functionalities as well: **User Management** and **Password Management**.

iConnect.NET membership provides a set of easy-to-use APIs to manage user accounts in your application. The common user management related functionalities include: adding a new user account; deleting an existing user account and other related user information specific to the user; Retrieving users based on different combination of search criteria. As with password management, not only the basic password based functionalities are supported, such as user registration; password retrieval and reset, you can also leverage the iConnect.NET membership configuration to apply different password policies to your application. These password policies include: specifying maximum number of invalid password or password answer attempts are allowed before the membership user is locked out; setting the max times which the new password cannot be same as the historic password for the same user, etc.

This section is dedicated to instruct you how to consume the functionalities provided by iConnect.NET Security Membership. Starting by a simple walk through to step you though a simple console application which utilizes the basic membership functionalities, we will get a completed introduction to the programming with iConnect.NET Membership. The introduction to usage in distributed scenario, as well as the introduction to the design overview and membership extensibility, will be placed in the last part.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.2.2. Getting Started

In this section,you will walk though a simple console application step by step, which will simulate the basic membership related scenarios: create a new user account; perform authentication against the newly created user; delete the created user.

Step I: Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first need to execute the following SQL script in this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*SecurityDemo*".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3c9413F6-3644-4733-8CD6-DA08B203D751'
           ,'SecurityDemo'
           ,'securitydemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Step II: Create a Console Application and Configure Membership

Create a Console application using your Visual Studio, and make sure the references to the following two assemblies are added. You can find the iConnect.NET related assemblies in the /bin folder of iConnect.NET installation directory.

- System.Web
- NCS.IConnect.Security
- NCS.IConnect.Security.Server

If you select .NET Framework 4.0 for you application, you need to add the reference to `System.Web.ApplicationServices` assembly. Then add an application configuration file and

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

copy the following XML snippet to override the existing configuration. Do not forget to revise the connection string accordingly.

```
<configuration>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    <membership defaultProvider="dbMembershipProvider">
      <providers>
        <clear/>
        <add name="dbMembershipProvider"
             applicationName="SecurityDemo"
             database="iConnectDb"
             type="NCS.IConnect.Security.DbMembershipProvider,
                   NCS.IConnect.Security.Server"
             requiresQuestionAndAnswer="false"
             requiresUniqueEmail="false"/>
      </providers>
    </membership>
  </system.web>
</configuration>
```

Step III: Program Membership

Write the following code to fulfill the three membership related tasks: `User creation`, `User validation (Authentication)` and `User Deletion`.

```
//Create a new user: Foo
MembershipUser user = Membership.CreateUser("Foo", "Password+");
Console.WriteLine("The user account \"{0}\" has been successfully created!", user.UserName);

//Athenticate the user Foo using correct password.
bool passed = Membership.ValidateUser("Foo", "Password+");
Console.WriteLine("The user account \"{0}\" has {1}passed the validation with the password you
supplied!",
    "Foo", passed ? string.Empty : "Not ");

//Delete the user Foo from data source.
bool deleted = Membership.DeleteUser("Foo");
Console.WriteLine("The user account \"{0}\" has {1}been scuessfully deleted!", "Foo", passed ?
string.Empty : "Not");
```

Run the console application, and you can get the following output.

```
The user account "Foo" has been successfully created!
The user account "Foo" has passed the validation with the password you supplied!
The user account "Foo" has been scuessfully deleted!
```

18.2.3. Configuration

iConnect.NET membership is completely based on ASP.NET membership, which adopts strategy design pattern. The membership related functionalities are ultimately implemented in variety of iConnect.NET membership providers. iConnect.NET self-defined membership providers are all derived from ASP.NET membership base provider: `MembershipProvider`. `MembershipProvider` defines a series of read-only properties, which determine the membership behavior at runtime, and you can specify these properties in an administratively configuring

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

fashion. The following XML represents the configuration the ASP.NET Membership with iConnect.NET `DbMembershipProvider` is used.

```
<system.web>
    <membership defaultProvider="DbProvider" userIsOnlineTimeWindow="20"❶>
      <providers>
        <add
          name="DbProvider" ❷
          type="NCS.IConnect.Security.DbMembershipProvider,
                NCS.IConnect.Security.Server" ❸
          database="db" ❹
          applicationName="iconnect"❺
          enablePasswordRetrieval="false" ❻
          enablePasswordReset="true" ❼
          requiresUniqueEmail="false" ❽
          requiresQuestionAndAnswer="false" ❾
          minRequiredPasswordLength="6" ❿
          maxPasswordLength = "36" 11
          maxInvalidPasswordAttempts = "3" 12
          minRequiredNonalphanumericCharacters="1" 13
          hashAlgorithmType="SHA1" 14
          passwordFormat="Hashed"15
          passwordRepeatWindow="3" 16
          requiresOldPassword = "true"17
          passwordStrengthRegularExpression =" @\"(?=.{6,})(?=(.*\d){1,})(?=(.*\W){1,})"18
          passwordAttemptWindow = "2" 19/>
      </providers>
    </membership>
</system.web>
```

❶ Specifies the number of minutes after the last activity date/time stamp for a user during which the user is considered online.

❷ The name of the membership provider which uniquely identifies the provider in the configured providers collection.

❸ The qualified name of the membership provider type.

❹ The connection string name, which refers to the connection string defined in the <connectionStrings/> configuration section. It indication the connection to the database storing the membership data.

❺ The name of the application.

❻ Indicates whether the password is allowed to retrieve.

❼ Indicates whether the password is allowed to reset.

❽ A value indicating whether a unique email address is required.

❾ A value indicating whether answer is required when resetting password.

❿ The minimum length of password.

11 The maximum length of password. This is only provided by iConnect.NET.

12 The count of invalid password or password answer attempts allowed before the membership user is locked out..

13 The minimum number of special characters that must be present in a valid password.

14 The valid hash algorithm supported by .NET framework, by default is SHA1.

16 A value indicating the format for storing passwords in the membership data store. It can be "Clear", "Hashed" or "Encrypted".

15 The max times which the new password cannot be same as the historic password for the same user. This is provided by iConnect.NET.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

17 A value indicating whether the membership provider is configured to require the user to provide the old password for password changing. This is provided by iConnect.NET.

18 The regular expression used to evaluate a password.

19 The number of minutes in which a maximum number of invalid password or password answer attempts are allowed before the membership user is locked out.

All of the above `MessageProvider` based properties can only be applied to `DbMembershipProvider`, which is used at server side in distributed application or non-distributed application. In distributed scenarios, the `WcfMembershipProvider` is used at the client side. For `WcfMembershipProvider`, just only two properties, *`applicationName`* and *`endpoint`*, can be configured.

```
<system.serviceModel>
  <client>
     <endpoint name="membershipService"
       behaviorConfiguration="sampleEndpointBehavior"
       binding="wsHttpBinding"
       contract="NCS.IConnect.Security.IMembership"
       address="http://localhost/ICSampleService/Iconnect/Membership.svc"/>
  </client>
</system.serviceModel>
<system.web>
   <membership defaultProvider="WcfMembershipProvider">
    <providers>
        <add name="WcfMembershipProvider"
             applicationName="application name"
             type="NCS.IConnect.Security.WcfMembershipProvider,
                   NCS.IConnect.Security.Client"
             endpoint="membershipService"/>
    </providers>
  </membership>
</system.web>
```

18.2.4. User Maintenance

When programing iConnect.NET membership, you just need to program with the two facade type `System.Web.Security.Membership` and `NCS.IConnect.Security.ExtendedMembership`. `Membership` is ASP.NET built-in facade class On the basis of the Membership facade type, `ExtendedMembership` provides some additional APIs.

Tip

It is recommended to use System.Web.Security.Membership as possible as you can. NCS.IConnect.Security.ExtendedMembership is just used only if your expected APIs cannot be found in System.Web.Security.Membership and you can make sure the iConnect.NET specific membership provider must be shipped in your application. This kind of abstract orientated programming can decouple the dependence to the iConnect.NET membership provider.

- Creates a new user account

```
MembershipUser userFoo =Membership.CreateUser("Foo"❶, "Password+"❷);
MembershipUser userBar = Membership.CreateUser("Bar", "Password+", "bar@ncsi.com.cn"❸);

MembershipCreateStatus createStatus;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
MembershipUser userBaz = Membership.CreateUser("Baz", "Password+", "baz@ncsi.com.cn",
    "What is your pet's name?"❹, "Little Q"❺, true❻, out createStatus❼);

Guid userKey = Guid.NewGuid();
MembershipUser userJohnDoe = Membership.CreateUser("John Doe", "Password+", "johndoe@ncsi.com.cn",
    "What is your pet's name?", "Little Q", true, userKey❽, out createStatus);
```

Or

```
ExtendedMembershipUser userFoo = ExtendedMembership.CreateUser("Foo", "Password+");
ExtendedMembershipUser userBar = ExtendedMembership.CreateUser("Bar", "Password+",
"bar@ncsi.com.cn");

MembershipCreateStatus createStatus;
ExtendedMembershipUser userBaz = ExtendedMembership.CreateUser("Baz", "Password+",
"baz@ncsi.com.cn",
    "What is your pet's name?", "Little Q", true, out createStatus);

Guid userKey = Guid.NewGuid();
ExtendedMembershipUser userJohnDoe = ExtendedMembership.CreateUser("John Doe", "Password+",
   "johndoe@ncsi.com.cn", "What is your pet's name?", "Little Q", true, userKey, out createStatus);
```

❶ The name of the user account to create.

❷ The password of the user account to create.

❸ Email address

❹ Password question for password retrieval or reset.

❺ The answer to the password question.

❻ A value of `Boolean` type, which indicates whether the user is approved to log on.

❼ A **System.Web.Security.MembershipCreateStatus** enumeration which indicates the success, failure or other status for the user account creation attempt.

❽ The user identifier from the membership data source for the user

Generally speaking, there are two main ways to create a user account. The first way is user registration that a new user registers himself or herself as a membership user. In this manner, the user specifies the user name, password and other related information. For some systems, instead of user registration, system will automatically generate a batch of user accounts, and randomly generate their passwords. In the applications of high security requirement, it is usually not allowed for the users to use these automatically generated user account to sign in the application. For these user accounts, the password change is mandatory prior to login. To address such a user account creation, another `CreateUser` overload is added in `ExtentedMembership` as below. The latest parameter "***`autoGenerate`***" indicates whether the user account is generated by system.

```
public static ExtendedMembershipUser CreateUser(string username, string password, string email,
    string passwordQuestion, string passwordAnswer,
bool isApproved, object providerUserKey, out MembershipCreateStatus status, string flag,
    string comment, bool autoGenerate));
```

- Deletes an existing user

```
bool deleted = Membership.DeleteUser("Foo"❶);
deleted = Membership.DeleteUser("Bar", true❷);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ The name of the user to delete.

❷ A value indicating whether to delete other information regarding the user to delete.

```
ExtendedMembershipUser user = ExtendedMembership.GetUser("Foo");
bool deleted = ExtendedMembership.DeleteUser(user❶);

user = ExtendedMembership.GetUser("Bar");
deleted = ExtendedMembership.DeleteUser(user,true❷);
deleted = ExtendedMembership.DeleteUser("Baz", true,1❸);
```

❶ An `NCS.IConnect.Security.ExtendedMembershipUser` instance which represents the user to delete.

❷ A value indicating whether to delete other information regarding the user.

❸ The version no used for concurrent operation detection.

Tip

System.Web.Security.MembershipUser is used to represent a single membership user. NCS.IConnect.Security.ExtendedMembershipUser extends MembershipUser and defines some extra iConnect.NET user specific properties, such as AuditInfo (Auditing information). The delete method operating on NCS.IConnect.Security.ExtendedMembershipUser is belong to iConnect.NET Membership only, and in System.Web.Security.Membership,there is not an equivalent one.

- Gets all users

```
MembershipUserCollection users = Membership.GetAllUsers();

int userCount;
users = Membership.GetAllUsers(0❶, 10❷, out userCount❸);
```

Or

```
ExtendedMembershipUserCollection users = ExtendedMembership.GetAllUsers();

int userCount;
users = ExtendedMembership.GetAllUsers(0❶, 10❷, out userCount❸);
```

❶ The index of the page of results to return. Zero(0) indicates the first page.

❷ The size of the page of results to return.

❸ The total number of users retrieved.

Note

As well as System.Web.Security.MembershipUserCollectioninstance represents the collection of System.Web.Security.MembershipUser, the collection of NCS.IConnect.Security.ExtendedMembershipUser is represented by an NCS.IConnect.Security.ExtendedMembershipUserCollection instance.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Gets a particular user

  You can use a series of `GetUserXxx` methods of Membership or `ExtendedMembership` facade type to get a particular user, based on the user name, user provider key and email specified.

```
MembershipUser user = Membership.GetUser();❶
user = Membership.GetUser(true❷);

user = Membership.GetUser("Foo"❸);
user = Membership.GetUser("Foo", true);

Guid userKey = new Guid("0B489CD2-4840-4227-9A3E-D149FDF149E6");
user = Membership.GetUser(userKey❹);
user = Membership.GetUser(userKey,true);

string userName = Membership.GetUserNameByEmail("foo@ncsi.com.cn"❺);
```

  Or

```
ExtendedMembershipUser user = ExtendedMembership.GetUser()❶;
user = ExtendedMembership.GetUser(true❷);

user = ExtendedMembership.GetUser("Foo"❸);
user = ExtendedMembership.GetUser("Foo", true);

Guid userKey = new Guid("0B489CD2-4840-4227-9A3E-D149FDF149E6");
user = ExtendedMembership.GetUserByKey(userKey❹);
user = ExtendedMembership.GetUserByKey(userKey, true);

string userName = ExtendedMembership.GetUserNameByEmail("foo@ncsi.com.cn"❺);
```

❶ Gets the current logged-on membership user.

❷ A `Boolean` value indicating whether to update the last-activity time stamp.

❸ The name of the user to retrieve.

❹ The unique user identifier from the membership data source for the user.

❺ The e-mail address to search for.

Note

If the user you specified does not exist in the user account storage, an exception will be thrown with a friendly error message indicating the given user does not exist.

Note

The last activity date/time is stored in the membership table. The action performed by a particular membership user will change his or her last activity date/tim to the current system date time. So you should consider whether the last activity date/time should be changed in your current scenario and choose an appropriate overloaded method. The rest overloaded ones without the activity date/time indicator will automatically alter the last activity date/time.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

iConnect.NET static facade type "`ExtendedMembership`" provides two additional overloaded `TryGetUser` methods. No exception is thrown if no user found, instead a `Boolean` value is returned to indicate whether the user has been found and the retrieved user is return as an output parameter.

```
MembershipUser user;
bool userExists = ExtendedMembership.TryGetUser("Foo"❶, true❷, out user❸);

Guid userKey = new Guid("0B489CD2-4840-4227-9A3E-D149FDF149E6");
user = ExtendedMembership.GetUserByKey(userKey❹);
userExists = ExtendedMembership.TryGetUser(userKey1, true, out user);
```

❶ The name of the user to get.

❷ A value indicating whether to update the user specific last-activity date/time stamp.

❸ A System.Web.Security.MembershipUser representing the membership user retrieved.

❹ The unique identifier for the membership user to get.

Note

`TryGetUser` is defined in iConnect.NET Membership facade class only, and there is not an equivalent one defined in System.Web.Security.Membership.

- Gets a collection of membership users

```
MembershipUserCollection users = Membership.FindUsersByEmail("%@ncsi.com.cn"❶);

int userCount;
users = Membership.FindUsersByEmail("%@ncsi.com.cn",0❷,10❸,out userCount❹);
```

Or

```
ExtendedMembershipUserCollection users = ExtendedMembership.FindUsersByEmail("%@ncsi.com.cn"❶);

int userCount;
users = ExtendedMembership.FindUsersByEmail("%@ncsi.com.cn", 0❷, 10❸, out userCount❹);
```

❶ The e-mail address of the membership users to search.

❷ The index of the page of results to return. Page index is zero-based.

❸ The size of the page of results to return.

❹ The total number of users.

- Unlock user

```
bool locked = ExtendedMembership.UnlockUser("Foo");
ExtendedMembershipUser user = ExtendedMembership.GetUser("Foo"❶);
ExtendedMembership.UnlockUser(user);
```

❶ The name of the user to unlock.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Note

The `UnlockUser` method is only defined in NCS.IConnect.Security.ExtendedMembership, and there is no equivalent one in System.Web.Security.Membership. This method is used to unlock a particular user account which was locked out for some reasons.

- Updates User

```
MembershipUser user = Membership.GetUser("Foo");
user.Email = "foo@ncsi.com.cn";
Membership.UpdateUser(user);
```

Or

```
ExtendedMembershipUser user = ExtendedMembership.GetUser("Foo");
user.Email = "foo@ncsi.com.cn";
ExtendedMembership.UpdateUser(user);
```

- Gets the number of users online

```
int numberOfOnlineUser = Membership.GetNumberOfUsersOnline();
```

Or

```
int numberOfOnlineUser = ExtendedMembership.GetNumberOfUsersOnline();
```

Note

The return value of `GetNumberOfUsersOnline` cannot really reflect the number of current online users, instead this method just returns the number of users whose last-activity date is greater than the current time less the *`UserIsOnlineTimeWindow`*.

- User account suspension

For the general active user account, it can be suspended in the following three manners:

- The administrator or other user with the appropriate privilege suspends a particular user account on demand.
- If the suspend duration (Suspend Start Date and Suspend End Date) is specified, in such a duration, the user account will be considered as Suspended.
- The user account will be automatically suspended if it has not logged on system in a given duration.

The following three methods will be added into iConnect.NET membership facade class: `ExtendedMembership`. **Activate|Deactivate** method is used to activate (deactivate) an inactive

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

(active) user account. `SetSuspendDuration` is used to set the Suspend Start Date and End Date.

```
public void Activate(string userName);
public void DeActivate(string userName);
public void SetSuspendDuration(string userName, DateTime startDate, DateTime endDate);
```

18.2.5. User Validation

- Authenticates the given user using the password specified.

```
bool valid = Membership.ValidateUser("Foo"❶, "Password+"❷);
```

Or

```
bool valid = ExtendedMembership.ValidateUser("Foo"❶, "Password+"❷);
```

❶ The name of the user to validate.

❷ The password for the user.

- Gets error information regarding user validation

```
string errorMessage =  ExtendedMembership.GetValidateUserErrorInfo("Foo","Password+");
```

Note

The `GetValidateUserErrorInfo` is exclusively defined in NCS.IConnect.Security.ExtendedMembership, and there is not equivalent one in System.Web.Security.Membership. Additionally, each invocation to `ValidateUser` with a passed incorrect password will cause the failed password attempt number increase by 1, and if this number exceeds the `PasswordAttemptWindow` set in your configuration, the current user will be locked out for security reason. However, invocation to `GetValidateUserErrorInfo` will not alter the failed validation attempt count.

18.2.6. Change and Retrieve Password

- Generates a random password

```
string password = Membership.GeneratePassword(16❶,3❷);
```

❶ The number of characters for the password generated.

❷ The minimum number of punctuation characters in the generated password.

- Changes password

```
ExtendedMembership.ChangePassword("Foo", "newPassword");
ExtendedMembership.ChangePassword("Foo", "oldPassword", "newPassword");
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Note

The `ChangePassword` method is exclusively defined in iConnect.NET membership only, and there is not equivalent one in System.Web.Security.Membership. The configuration property RequiresOldPassword determines whether you must provide the original password when your attempt to change the password.

- Gets the password for the specified user name

```
string password = ExtendedMembership.GetPassword("Foo"❶, "Little Q"❷);
```

❶ The user to retrieve the password for.

❷ The password answer for the user.

Note

Since the password is sensitive information, it is usually be encrypted or hashed in data persistence. Hashing algorithm is asymmetrical, so you cannot retrieve the original password as is if it is hashed. Additionally, the password can be retrieved only if the *`enablePasswordRetrieval`* is configured to true.

- Resets a user's password

```
string password = ExtendedMembership.ResetPassword("Foo"❶, "Little Q"❷);
```

❶ The user to retrieve the password for.

❷ The password answer for the user.

Note

Both of the two password related functionalities: password retrieval and password reset, are just only defined in NCS.IConnect.Security.ExtendedMembership, and there are not equivalent ones defined in System.Web.Security.Membership. Similar to password retrieval, the password is allowed to be reset only if *`enablePasswordReset`*is configured to true.

18.3. Role Manager

18.3.1. Introduction

Authentication and authorization are two important aspects for security system. Authentication verify the user is the one he or she claims to be, and authorization determines whether the user has full permission to the secured resource In iConnect.NET security component, membership is in charge of authentication and user management. As with authorization, iConnect.NET

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Role is the core module, since all of permissions are bound to a particular role. Other authorization , such as Code Table Authorization, Site Map Authorization, Rule-based and Business Action based Authorization are all role-centric.

Besides the basic authorization functionalities, iConnect.NET Roles also functions as a manager to maintain roles. Programming with iConnect.NET Role, you can easily create a new role; retrieve related roles based on different search criteria; delete an exiting role etc.

This section is dedicated to instruct you how to consume the functionalities provided by iConnect.NET Security Role. Starts by a simple walk through which will step you though a simple console application which utilizes the basic role functionalities, we then get a systemic introduction to how to program with iConnect.NET Role. The introduction to iConnect.NET Role usage in distributed scenario will be placed in the last part, as well as the introduction the design overview and role extensibility.

18.3.2. Getting Started

In this section,you will walk though a simple console application step by step, which will simulate the basic role related scenarios: Create a new user account; Create two new roles; Add the newly created user to one of the created roles; Check if the user is in the given role.

Step I: Create Application Record and Console Application

Similar to Membership Getting start demonstration, we should execute a SQL script insert a record, which stands for the current demonstrative application, into the T_IC_APP. You can get such a script from the Getting Started section of Membership, and we also name our application *SecurityDemo*. If such a record is here, this step can be ignored. And then create a Console Application, which has the assembly references.

Step II: Configure Role Management

Add an application configuration file and copy the following XML snippet to override the existing configuration. Do not forget to revise the connection string accordingly.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  <system.web>
    </membership>
      <roleManager defaultProvider="roleProvider" enabled="true">
      <providers>
        <add name="roleProvider"
             applicationName="SecurityDemo"
             database="iConnectDb"
             type="NCS.IConnect.Security.DbRoleProvider,
                   NCS.IConnect.Security.Server" />
      </providers>
  </system.web>
</configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step III: Program Role Management

Write the following source to fulfill these tasks: `User creation`; Role creation; Adding user to role(s); Checking if user in role(s).

```
//Create two roles: Managers & Administrators
Roles.CreateRole("Managers");
Roles.CreateRole("Administrators");

//Add the newly created user Foo in the role of Managers.
Roles.AddUserToRole("Alice", "Managers");

//Check if the newly created user Foo is in the roles:  Managers & Administrators
bool isManager = Roles.IsUserInRole("Alice", "Managers");
bool isAdministrator = Roles.IsUserInRole("Alice", "Administrators");

Console.WriteLine("Is Alice a manager? {0}", isManager ? "Yes" : "No");
Console.WriteLine("Is Alice an administraor? {0}", isAdministrator ? "Yes" : "No");
```

Run the console application and you can get the following output.

```
Is Alice a manager? Yes
Is Alice an administraor? No
```

18.3.3. Configuration

You can set the role manager and role provider in your configuration file as below. All of configuration is defined in System.Web.Configuration.RoleManagerSection.

```
<system.web>
    <roleManager defaultProvider="DbRoleProvider"❶
                 enabled="true"❷
                 cacheRolesInCookie="true"❸
                 cookieName=".ASPROLES"❹
                 cookieTimeout="30"❺
                 cookiePath="/"❻
                 cookieRequireSSL="false"❼
                 cookieSlidingExpiration="true"❽
                 cookieProtection="All"❾>
      <providers>
        <clear/>
        <add name="DbRoleProvider"
             applicationName="ApplicationName"
             database="iConnectSqlServer"
             type="NCS.IConnect.Security.DbRoleProvider,
                   NCS.IConnect.Security.Server" />
      </providers>
    </roleManager>
</system.web>
```

❶ The name of the default provider the role manager uses.

❷ The value indicating whether the role management feature is enabled

❸ The value indicating whether the current user's roles are cached in cookie.

❹ The name of the cookie caching the roles.

❺ The number of minutes before the role cookie expires.

❻ The virtual path of the role cookie.

❼ The value indicating whether Secure Sockets Layer (SSL) connection is need to send role cookie to the server.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❽ The value indicating whether the role cookie will be reset periodically.

❾ The type of security that is used to protect the role cookie.

As with `WcfRoleProcider`, which is used at client side in distributed application. Just only the *`appicationName`* and *`endpoint`* need to be configured.

```
<system.web>
 <roleManager defaultProvider="wcfProvider">
  <providers>
   <add name="wcfProvider"
        type="NCS.IConnect.Security.WcfRoleProvider,
              NCS.IConnect.Security.Client"
        applicationName="appName"
        endpoint="roleService"/>
  </providers>
 </roleManager>
</system.web>
```

18.3.4. Role Maintenance

- Creates a new role

```
Roles.CreateRole("Managers"❶);
```

❶ The name of the role to create.

The `CreateRole` with a literal role name as its input parameter is only defined in the ASP.NET static facade class: System.Web.Security.Roless, and there is not an equivalent one in iConnect.NET specific facade class (`NCS.IConnect.Security.ExtendedRoles`). iConnect.NET create a new type to represent the "role": NCS.IConnect.Security.ExtendedRole. In ExtendedRoles, two overloaded CreateRole methods are added to create iConnect.NET specific role. The first one accepts a ExtendedRole object as its input parameter:

```
ExtendedRole role = new ExtendedRole("Managers"❶,"Role desctiption"❷);
role = ExtendedRoles.CreateRole(role);
```

❶ The name of the role to create.

❷ The description of the role to create.

You can also invocate another overloaded `CreateRole` method with role related properties directly specified in the parameters:

```
ExtendedRole role = ExtendedRoles.CreateRole("Managers", "Role description", "Role flag"❶);
```

❶ The flag of the role to create.

Besides creating a single role once, another method `CreateRoles` is added in NCS.IConnect.Security.ExtendedRoles class to enable you to create a batch of roles once.

```
ExtendedRoleCollection roles = new ExtendedRoleCollection();
roles.Add(new ExtendedRole("Managers","Role description"));
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
roles.Add(new ExtendedRole("Administrators","Role description"));
roles = ExtendedRoles.CreateRoles(roles);
```

- Deletes an existing role

```
Roles.DeleteRole("Managers"❶);
Roles.DeleteRole("Administrator", true❷);
```

❶ The name of the role to delete.

❷ A value indicating whether to throw an exception if given role name has one or more members.

Note

If the second parameter is true, an exception will be thrown and if the role to delete has one or more members. If false, the role will always be deleted.

There is another `DeleteRole`, which accepts a `ExtendedRole` object, defined in the NCS.IConnect.Security.ExtendedRoles . You can invoke it as below:

```
ExtendedRole role = ExtendedRoles.GetRole("Managers");
ExtendedRoles.DeleteRole(role);
```

Additionally, another `DeleteRole` overloaded method is defined in ExtendedRoles, which you can invoke to delete an existing role by directly specifying the role name, and version no for concurrent operation checking.

```
ExtendedRoles.DeleteRole("Managers", 1❶, true❷);
```

❶ The version no of the role to delete.

❷ A value indicating whether to throw an exception if given role has one or more members.

- Checks role existence

```
bool roleExists = Roles.RoleExists("Managers"❶);
```

Or

```
bool roleExists = ExtendedRoles.RoleExists("Managers"❶);
```

❶ The name of the role to perform existence checking.

- Get all roles

```
string[] roles = Roles.GetAllRoles();
```

Or

```
string[] roles = ExtendedRoles.GetAllRoles();
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Besides return all role names in the form of string array, NCS.IConnect.Security.ExtendedRoles defines an additional method `GetAllRolesAsCollection` to return all roles in the form of NCS.IConnect.Security.ExtendedRoleCollection collection.

```
ExtendedRoleCollection roles = ExtendedRoles.GetAllRolesAsCollection();
```

- Gets or Finds role(s)

ExtendedRoles provides a series of APIs to retrieve expected roles based on variety of search criteria, and the retrieved role(s) is represented in the form of NCS.IConnect.Security.ExtendedRole or NCS.IConnect.Security.ExtendedRoleCollection object.

`Gets role based on the role name specified:`

```
ExtendedRole role = ExtendedRoles.GetRole("Managers");
```

`Finds roles based on the Flag specified:`

```
ExtendedRoleCollection roles = ExtendedRoles.FindRolesByFlagAsCollection("manager%");
```

Finds roles based on the role name with wildcard (%) :

```
ExtendedRoleCollection roles = ExtendedRoles.FindRolesByNameAsCollection("manager%");
```

- Updates existing role(s)

```
ExtendedRole role = new ExtendedRole("Managers","Role description");
role = ExtendedRoles.UpdateRole(role);

role = ExtendedRoles.UpdateRole("Managers"❶, "Role description"❷, "Role flag"❸, 1❹);
```

❶ The role name that represents the role to update.
❷ The role description which will be updated to.
❸ The role flag which will be updated to.
❹ The version no of the role in the data source.

Except for updating a single role once, ExtendedRoles provides another method `UpdateRoles` to perform batch updating. `UpdateRoles` accepts an ExtendedRoleCollection instance as passed parameter.

```
ExtendedRoleCollection roles = new ExtendedRoleCollection();
roles.Add(new ExtendedRole("Managers", "Role decription"));
roles.Add(new ExtendedRole("Administrators", "Role decription"));
roles = ExtendedRoles.UpdateRoles(roles);
```

18.3.5. Authorization

- Adds user(s) to role(s)

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
Roles.AddUserToRole("Foo"❶, "Managers"❷);
Roles.AddUserToRoles("Foo", new string[] { "Managers", "Administrator" }❸);

Roles.AddUsersToRole(new string[] { "Foo", "Bar" }❹, "Managers");
Roles.AddUsersToRoles(new string[] { "Foo", "Bar" }, new string[] { "Managers", "Administrator" });
```

Or

```
ExtendedRoles.AddUserToRole("Foo"❶, "Managers"❷);
ExtendedRoles.AddUserToRoles("Foo", new string[] { "Managers", "Administrator" }❸);

ExtendedRoles.AddUsersToRole(new string[] { "Foo", "Bar" }❹, "Managers");
ExtendedRoles.AddUsersToRoles(new string[] { "Foo", "Bar" }, new string[] { "Managers",
"Administrator" });
```

❶ The user name to add to the specified role(s).

❷ The role to add the specified user name to.

❸ A string array of user names to add to the specified roles.

❹ A string array of roles to add the specified user name to.

Note

If adding a user into a role whose members has includes such a user, an exception will be thrown.

Generally speaking, once a role is assigned to a user account, the user account will be granted the role based privilege. However, there are lots scenarios that we do not want to make the role based privilege take effect on demand when adding the user account into the role. For example an employee is promoted to director; however, the formal appointment will go into effect next month. Now the administrator will add the user account into director related role in advance. To address the above scenario, we can invoke another another `AddUser(s)ToRole(s)` overloaded method and specify the effective start date and end date.

```
AddUserToRole(string user, string role, DateTime startDate, DateTime endDate)
AddUserToRoles(string user, string[] roles, DateTime startDate, DateTime endDate)
AddUsersToRole(string[] users, string role, DateTime startDate, DateTime endDate)
AddUsersToRoles(string[] users, string[] roles, DateTime startDate, DateTime endDate)
```

- Removes user(s) from role(s)

```
Roles.RemoveUserFromRole("Foo"❶, "Managers"❷);
Roles.RemoveUserFromRoles("Foo", new string[] { "Managers", "Administrators" }❸);

Roles.RemoveUsersFromRole(new string[] { "Foo", "Bar" }❹, "Managers");
Roles.RemoveUsersFromRoles(new string[] { "Foo", "Bar" }, new string[] { "Managers",
"Administrators" });
```

Or

```
ExtendedRoles.RemoveUserFromRole("Foo"❶, "Managers"❷);
ExtendedRoles.RemoveUserFromRoles("Foo", new string[] { "Managers", "Administrators" }❸);

ExtendedRoles.RemoveUsersFromRole(new string[] { "Foo", "Bar" }❹, "Managers");
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
ExtendedRoles.RemoveUsersFromRoles(new string[] { "Foo", "Bar" }, new string[] { "Managers",
"Administrators" });
```

❶ The name of user to remove from the role(s) specified

❷ The name of role from which the user(s) is removed.

❸ A string array representing the users to be removed from the roles specified.

❹ A string array representing the roles from which the user(s) is removed.

Note

If removing a user from the role which does not have such a member, an exception will be thrown.

- Gets all roles for a user

```
string[] roles = Roles.GetRolesForUser()❶;
roles = Roles.GetRolesForUser("Foo")❷;
```

Or

```
string[] roles = ExtendedRoles.GetRolesForUser()❶;
roles = ExtendedRoles.GetRolesForUser("Foo")❷;
```

❶ Gets a list of the roles that the currently logged-on user is in.

❷ The name of user form whom the roles is retrieved.

For the above `GetRolesForUser`, the roles retrieved are returned as a string array. The following two methods, `GetRolesForUserAsCollection`, return the retrieved roles in the form of a `ExtendedRoleCollection` object.

```
ExtendedRoleCollection roles = ExtendedRoles.GetRolesForUserAsCollection();
roles = ExtendedRoles.GetRolesForUserAsCollection();
```

Effective duration is applied to the user and role assignment, and the above method returns the currently effective roles for the user. You can invoke another overloaded `GetRolesForUser` and `GetRolesForUserAsCollection` methods to the effective roles at a particular date specified, or in the period specified.

```
GetRolesForUser(DateTime startDate, DateTime endDate)
GetRolesForUser(string user ,DateTime startDate, DateTime endDate)

GetRolesForUserAsCollection(DateTime startDate, DateTime endDate)
GetRolesForUserAsCollection(string user ,DateTime startDate, DateTime endDate)
```

- Checks if a particular user is in the given role(s)

```
bool isManager = Roles.IsUserInRole("Managers"❶)❷;
isManager = Roles.IsUserInRole("Foo"❸,"Managers");
```

Or

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
bool isManager = ExtendedRoles.IsUserInRole("Managers"❶)❷;
isManager = ExtendedRoles.IsUserInRole("Foo"❸, "Managers");
```

❶ The name of the role

❷ This method gets a value indicating whether the currently logged-on user is in the role specified.

❸ The name of the user

The above method check if the role specified is in the currently effective role list for the user. You can also invoke another overloaded `IsUserInRole` method to check if the user is an effective member at a particular date or in the period specified.

```
public static bool IsUserInRole(string roleName, DateTime atTime);
public static bool IsUserInRole(string roleName, DateTime fromTime, DateTime toTime);
public static bool IsUserInRole(string userName, string roleName, DateTime atTime);
public static bool IsUserInRole(string userName, string roleName, DateTime fromTime, DateTime
toTime);
```

- Gets user(s) in the given role

```
string[] users = Roles.GetUsersInRole("Managers");
```

Or

```
string[] users = ExtendedRoles.GetUsersInRole("Managers");
```

Besides just only returning the user name list, an additional method `GetUsersInRoleAsCollection` is defined in ExtendedRoles to get the users as an ExtendedMembershipUserCollection instance:

```
ExtendedMembershipUserCollection users = ExtendedRoles.GetUsersInRoleAsCollection("Managers");
```

The above method the the currently effective members of the role specified. Yon can invoke another overloaded `GetUsersInRole` and `GetUsersInRoleAsCollection` methods to get the effective members of the role at the a particular date or in the period specified:

```
GetUsersInRole(string role, Datetime date)
GetUsersInRole(string role, DateTime startDate, DateTime endDate)
GetUsersInRoleAsCollection(string role, Datetime date)
GetUsersInRoleAsCollection(string role , DateTime startDate, DateTime endDate)
```

18.4. Authorization

18.4.1. Introduction

To a degree, iConnect.NET Role can help us implement many authorization related functionalities. In iConnect.NET Role, we perform authorization by checking if a particular user is in the given role. iConnect.NET Authorization gives us two more sophisticated authorisation solutions: Rule based authorization and Business Action based authorization.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Rule based authorization is directly inherited from *Enterprise Library Security Application Block*. You can define the authorization rule in an evaluation expression, E.g. This authorization rule expression "Administrator AND R: Manager" means the authorized user must be either in the role Administrator or Manager. Because iConnect.NET adopts the rule based authorization of Enterprise Library Application Block as is, you can refer to detailed information from officer site of Microsoft Patters and Practice: http://msdn2.microsoft.com/en-us/practices/default.aspx.

Besides the above rule based authorization, iConnect.NET Authorization provides another business action based authorization. In iConnect.NET, business action represents a secured action user perform in applications. By associating a business action with a series of roles, iConnect.NET can authorize a particular user against a given business action.

This section is dedicated to instruct you how to consume the functionalities provided by iConnect.NET Security Authorization. Starts by a simple walk through which will step you though a simple console application which utilizes the basic authorization functionalities, we then get a systemic introduction about how to program iConnect.NET Authorization. The introduction to iConnect.NET Authorization usage in distributed scenario will be placed in the last part.

18.4.2. Getting Started

In this section,you will walk though a simple console application step by step, which will simulate the basic business action authorization related scenarios: Create a new user account; Create two new roles; Add the newly created user to one of the created roles; Create two new business actions and Authorize the created user against the given business action(s).

Step I: Create Application Record and Console Application

Similar to Membership Getting start demonstration, we should execute a SQL script insert a record, which stands for the current demonstrative application, into the T_IC_APP. You can get such a script from the Getting Started section of Membership, and we also name our application *SecurityDemo*. If such a record is here, this step can be ignored. And then create a Console Application, which has the assembly references.

Step II: Configure Role Management

Add an application configuration file and copy the following XML snippet to override the existing configuration. Do not forget to revise the connection string accordingly.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="securityConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Security.Configuration.SecuritySettings,
                   Microsoft.Practices.EnterpriseLibrary.Security"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  <system.web>
    <roleManager defaultProvider="roleProvider" enabled="true">
      <providers>
        <add name="roleProvider"
             applicationName="SecurityDemo"
             database="iConnectDb"
             type="NCS.IConnect.Security.DbRoleProvider,
                   NCS.IConnect.Security.Server" />
      </providers>
    </roleManager>
  </system.web>
  <securityConfiguration defaultAuthorizationInstance="DbBusinessActionProvider">
    <authorizationProviders>
      <add name="DbBusinessActionProvider"
           type="NCS.IConnect.Security.DbBusinessActionProvider,
                 NCS.IConnect.Security.Server"
           applicationName="SecurityDemo"
           database="iConnectDb" />
    </authorizationProviders>
  </securityConfiguration>
</configuration>
```

Step III: Program Authorization

Write the following source to fulfill these tasks: User creation; Role creation; Adding user to role(s); Business Action Creation; Adding Actions to role(s); Authorizing user(s) against action(s).

```
//Create two roles: Managers and Administrators
Roles.CreateRole("Managers");
Roles.CreateRole("Administrators");

//Add newly created user account Foo to the role of Managers
Roles.AddUserToRole("Bob", "Managers");

//Create two business actions: View Employee Information & System Configuration
BusinessActions.CreateAction("View Employee Information", "View basic information of all of the
employees", string.Empty);
BusinessActions.CreateAction("System Configuration", "Configuring system setting", string.Empty);

//Add the newly created business actions to appropriate roles: View Employee Informatio->Managers;
System Configuration->Administrators
BusinessActions.AddActionsToRoles(new string[] { "View Employee Information" }, new string[] {
"Managers" });
BusinessActions.AddActionsToRoles(new string[] { "System Configuration" }, new string[] {
"Administrators" });

//Check if the user Foo have the appropriate permissions to the two business actions.
bool canViewEmployeeInfo = BusinessActions.Authorize("Bob", "View Employee Information");
bool canConfigureSystem = BusinessActions.Authorize("Bob", "System Configuration");

Console.WriteLine("Does Bob have the permission to view employee information? {0}", canViewEmployeeInfo
? "Yes" : "No");
Console.WriteLine("Does Bob have the permission to configure system setting? {0}", canConfigureSystem
? "Yes" : "No");
```

Run the console application by pressing F5, and you can get the following output:

```
Does Bob have the permission to view employee information? Yes
Does Bob have the permission to configure system setting? No
```

18.4.3. Configuration

We just need configure the appropriate `AuthorizationProvider`. The following code snippet represents the EnterLib security configuration with iConnect.NET `DbBusinessActionProvider`

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

as the default `AuthorizationProvider`. For this configuration, we can see just only the *`applicationName`* and *`database`* (the configuration element name of connection string).

```
<securityConfiguration defaultAuthorizationInstance="DbActionProvider">
    <authorizationProviders>
      <add name="dbAuthorizationRuleProvider"
           type="NCS.IConnect.Security.DbBusinessActionProvider,
                 NCS.IConnect.Security.Server"
           database="iConnectDb"
           applicationName="iconnect" />
    </authorizationProviders>
</securityConfiguration>
```

As with `WcfBusinessActionProvider`, which is used at client side in distributed application, we just need to configure the *`applicationName`* and *`endpoint`*.

```
<securityConfiguration defaultAuthorizationInstance="WcfActionProvider">
  <authorizationProviders>
   <add name="WcfActionProvider"
        type="NCS.IConnect.Security.WcfBusinessActionProvider,
              NCS.IConnect.Security.Client"
        applicationName="appName"
        endpoint="BusinessAction.svc" />
  </authorizationProviders>
</securityConfiguration>
```

18.4.4. Business Action Maintenance

- Creates new business action(s)

```
BusinessActions.CreateAction("System Configuration"❶, "Action description"❷, "Action flag"❸);
```

❶ Business action code.

❷ Business action description.

❸ Business action flag for extensibility purpose.

In iConnect.NET business action authorization, the type NCS.IConnect.Security.BusinessAction is defined to represent iConnect.NET specific business action and NCS.IConnect.Security.BusinessActionCollection represents action collection. Using the collection type BusinessActionCollection, you can invocate another method `CreateActions` to create a group of business actions once.

```
BusinessActionCollection actions = new BusinessActionCollection();

actions.Add(new BusinessAction("System Configuration", "Action description"));
actions.Add(new BusinessAction("Sales Report View","Action description"));
BusinessActions.CreateActions(actions);
```

Tip

The action(s) creation operation is executed in a single transaction, and any encountered error will cause the transaction roll back. If the business action you create is existent in the data source, an exception will be thrown with a friendly indicated error message.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Deletes existing business action(s)

```
BusinessActions.DeleteAction("System Configuration", 1❶);

BusinessActionCollection actions = new BusinessActionCollection();
actions.Add(new BusinessAction("System Configuration", "Action description"));
actions.Add(new BusinessAction("Sales Report View", "Action description"));
BusinessActions.DeleteActions(actions);
```

❶ Version number for the purpose of concurrent invocation detection.

- Checks business action existence

```
bool actionExists = BusinessActions.ActionExists("System Configuration"❶);
```

❶ The code of the business action you check.

- Gets existing business actions

Get a business action based on the specified action code:

```
BusinessAction action = BusinessActions.GetAction("System Configuration");
```

Tip

If no business action can be found in data source based on your specified action code, no exception will be found and this method returns null instead.

Get all existing business actions:

```
string[] actionCodes = BusinessActions.GetAllActions();❶
BusinessActionCollection actions = BusinessActions.GetAllActionsAsCollection();❷
```

❶ This method just only returns action code list in the form of string array.

❷ This method returns all of the existing business actions in the form of BusinessActionCollection, each element of which is a BusinessAction instance encapsulating all related action related information.

BusinessActions define another method `FindActionsByCodeAsCollection` to perform action code based search. Unlike `GetAction` method with an exact action code as its parameter, this method accepts a wildcard (%) action code as its parameter and returns a BusinessActionCollection instance.

```
BusinessActionCollection actions = BusinessActions.FindActionsByCodeAsCollection("%config%");
```

- Updates existing business action(s)

```
BusinessActionCollection actions = new BusinessActionCollection();
actions.Add(new BusinessAction("System Configuration", "Action description"));
actions.Add(new BusinessAction("Sales Report View", "Action description"));
BusinessActions.UpdateActions(actions);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Tip

The action(s) updating operation is executed in a single transaction, and any encountered error will cause the transaction roll back. If the business action you create is not existent in the data source, an exception will be thrown with a friendly indicated error message.

18.4.5. Authorization via Business Action

- Adds action(s) to Role(s)

```
BusinessActions.AddActionsToRoles(new string[] { "System Configuration", "Sales Report View" },
new string[] { "Managers", "Administrators" });
```

Tip

The batch processing is enlisted into a single transaction, and any encountered error will cause the transaction roll back. If the business action you create is not existent in the data source, an exception will be thrown with a friendly indicated error message. However, considering decoupling the dependence between Business Action Authorization module and Role module, no exception will be thrown in the situation that the given role can not be found in role related data source. In other words, the role existence validation will never be taken into the account of this method.

- Authorizes a user to a business action

```
IPrincipal principal = Thread.CurrentPrincipal;
bool isAuthorized = BusinessActions.Authorize(principal❶, "System Configuration"❷);
isAuthorized = BusinessActions.Authorize("Foo"❸, "System Confiruation");
```

❶ A principal instance derived System.Security.Principal.IPrincipal interface, which represents the security context information regarding the user you authorize.
❷ The code of the business action.
❸ The user name you will authorize.

Tip

If the business action you authorize against cannot be found in the data source, an exception will be thrown with a friendly message.

Note

A principal object represents the security context of the user on whose behalf the code is running, including that user's identity (System.Security.Principal.IPrincipal) and any roles to which they belong. All

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

principal objects are required to implement the System.Security.Principal.IPrincipal interface.

- Gets business action(s) in a role(s)

```
string[] actionCodes = BusinessActions.GetActionsInRole("Managers");
BusinessActionCollection actions = BusinessActions.GetActionsInRoleAsCollection("Managers");
```

Tip

Similar to `AddActionsToRoles`, in order to make business action authorization component loosely couple with Role component, this method will never check if the role really exists in the role data source. So, no exception will be thrown when passing an invalid non-existent role name to this method.

BusinessActions defines two additional methods: `FindActionsInRole` and `FindActionsInRoleAsCollection`, which enable you to perform actions search based on the combination of exact role name and wildcard (%) action code. The first one returns the all action codes in the form of string array, and the other one returns a BusinessActionCollection instance.

```
string[] actionCodes = BusinessActions.FindActionsInRole("Managers", "%Configuration%");
BusinessActionCollection actions = BusinessActions.FindActionsInRoleAsCollection("Managers",
"%Configuration%");
```

- Gets business action(s) for a user

Business action authorization is based on the association between role and business action, and role is further associated with a particular user. Accordingly, `BusinessActions` defines two methods `GetActionsForUser` and `GetActionsForUserAsCollection` to get a business actions list on which the given user account has the appropriate permissions.

```
string[] actionCodes = BusinessActions.GetActionsForUser("Foo");
BusinessActionCollection actions = BusinessActions.GetActionsForUserAsCollection("Foo");
```

The above method returns the business actions which are added in the currently effective roles for the user specified. You can invoke another overloaded `GetActionsForUser` and `GetActionsForUserAsCollection` methods to get effective actions for the user at a particular date or in the period specified.

```
GetActionsForUser(string user, Datetime date);
GetActionsForUser(string user, DateTime startDate, DateTime endDate);

GetActionsForUserAsCollection(string user, Datetime date);
GetActionsForUserAsCollection(string user, DateTime startDate, DateTime endDate);
```

- Removes business action(s) from role(s)

```
BusinessActions.RemoveActionsFromRoles(new string[] { "System Configuration" }, new string[] {
"Managers", "Administrators" });
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Tip

For the following situations, an exception will be thrown with an indicated error message: The business action specified cannot be found in data source; the association between business action(s) and role(s) you specified does not exist in data source. However, if the role name specified can not be found, no exception will be thrown.

- Checks if a given business action is associated with a particular role

```
bool isActionInRole = BusinessActions.IsActionInRole("System Configuration", "Managers");
```

Tip

If the business action specified is not existent, an exception will be thrown with an indicated error message. However, the check regarding role existence will never be taken into account.

- Gets role list which a given business action is in:

```
string[] roleNames = BusinessActions.GetRolesForAction("System Configuration");
ExtendedRoleCollection roles = BusinessActions.GetRolesForActionAsCollection("System Configuration");
```

18.4.6. `AuthorizeCallHandler`

The `AuthorizeCallHandler` is used to preform authorization in an *AOP* manner. Its specific `AuthorizeCallHandlerAttribute` can be applied to the class or method. Both business action and rule based authorization are supported by `AuthorizeCallHandler`, and the business action code and authorization rule name are separately specified by two custom attributes, `BusinessActionAttribute` and `AuthorizationRuleAttribute`. The following code snippet illustrates the definition of the two attributes.

```
[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class, Inherited=true, AllowMultiple=false)]
public class BusinessActionAttribute : Attribute
{
    public BusinessActionAttribute(string actionCode);
    public string ActionCode { get; }
}

[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class, Inherited=true, AllowMultiple=false)]
public class AuthorizationRuleAttribute : Attribute
{
    public AuthorizationRuleAttribute(string ruleName);
    public string RuleName { get; }
}
```

According to the definition of `AttributeUsageAttribute` applied to the two attributes, `BusinessActionAttribute` and `AuthorizationRuleAttribute` can be both applied to class and method level. If different the two attributes are applied to both class and method, the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

method specific attribute takes higher priority. If both attributes are applied to the same method or class, `BusinessActionAttribute` takes higher priority.

If both of the authorized method and class are not decorated with the two attributes, `AuthorizeCallHandler` will figure out two special business action codes, which separately correspond to current method and class. The following lists the class and method specific business action code's format. Once the above business action codes are dynamically figured out, `AuthorizeCallHandler` determines whether the current method invocation is authorized by checking if anyone of the two is authorized to the current user.

- Class: [{Type Assembly Qualified Name}].[*]
- Method: [{Type Assembly Qualified Name}].[Method Name]

The above two kinds of business action codes can also be generated by calling the `GetBusinessActionCode` method of the static class of `BusinessActions`. As illustrated in the following code snippet, there is an optional parameter *`method`* with default value "*". The class based business action code will be generated if we do not explicitly specify the parameter or give the default value, otherwise the method specific business action code is generated.

```
[Serializable]
public static class BusinessActions
{
    //Others
    public static string GetBusinessActionCode(Type type, string method = "*");
}
```

Class and method based role assignment can be achieved by calling the `AddType(s)ToRole(s)` and `AddMethod(s)ToRole(s)` methods of the static class of `BusinessActions`. The following code snippet illustrates the definition of the two methods.

```
[Serializable]
public static class BusinessActions
{
    //Others
    public static void AddTypesToRole(Type[] types, string role);
    public static void AddTypesToRoles(Type[] types, string[] roles);
    public static void AddTypeToRole(Type type, string role);
    public static void AddTypeToRoles(Type type, string[] roles);

    public static void AddMethodsToRole(Type type, string[] methods, string role);
    public static void AddMethodsToRoles(Type type, string[] methods, string[] roles);
    public static void AddMethodToRole(Type type, string method, string role);
    public static void AddMethodToRoles(Type type, string method, string[] roles);
}
```

There are other methods defined in the static class of `BusinessActions`. As illustrated in the following code snippet, the static `IsTypeInRole/IsMethodInRole` method is used to check if the given role is assigned to the specified type/method. The method `RemoveType(s)FromRole(s)` and `RemoveMethod(s)FromRole(s)` are used to cancel the role assignment.

```
[Serializable]
public static class BusinessActions
{
    //Others
    public static bool IsTypeInRole(Type type, string role);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public static bool IsMethodInRole(Type type, string method, string role);

    public static void RemoveTypeFromRole(Type type, string role);
    public static void RemoveTypeFromRoles(Type type, string[] roles);
    public static void RemoveTypesFromRole(Type[] types, string role);
    public static void RemoveTypesFromRoles(Type[] types, string[] roles);

    public static void RemoveMethodFromRoles(Type type, string method, string[] roles);
    public static void RemoveMethodsFromRole(Type type, string[] methods, string role);
    public static void RemoveMethodsFromRoles(Type type, string[] methods, string[] roles);
    public static void RemoveMmethodFromRole(Type type, string method, string role);
}
```

Except for rule and business action based authorization, including type/method based business action authorization, we can also directly specify the authorized user name and role list when applying the `AuthorizeCallHandlerAttribute` to a particular method or class. As following code snippet illustrates, the properties of Users and Roles are defined in `AuthorizeCallHandlerAttribute`. The two properties of string type are used to be specified authorized user name and role list, and the delimiter is comma (",").

```
[AttributeUsage(AttributeTargets.Property | AttributeTargets.Method | AttributeTargets.Class)]
public class AuthorizeCallHandlerAttribute : HandlerAttribute
{
    //Others
    public string Users { get; set; }
    public string Roles { get; set; }

    public bool Supress { get; set; }
}
```

As the above code snippet illustrates, there is an extra Suppress property of boolean type, which is used to switch on/off authorization. The authorization will be ignored if Suppress property is true. This property's default value is false.

18.5. Profile

18.5.1. Introduction

Personalization is growing more and more of an essential ingredient for many types of web applications. Without it, it's quite difficult to serve your customers efficiently. But building a personalization layer in your presentation tier can be a challenge. Thankfully, iConnect.NET profile provides new facilities to help you build your personalization layer with ease.

iConnect.NET profile is built on the ASP.NET profiling system. ASP.NET profile, as an important personalization feature, is designed for persistent storage of structured profile data. To make use of personalization features, your application defines its own model of personalized data, and the ASP.NET runtime does all of the rest by parsing and compiling that model into a class. Each member of the personalized data class corresponds to a piece of information specific to the current user. Loading and saving personalized data is completely transparent to end users.

The ASP.NET 4.0 Profile ships with a single provider – SqlProfileProvider. This provider is very flexible and automatically works with any serializable profile data. However the cost of

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

this flexibility is that all profile data is stored in either an **opaque `text blob`** or an **opaque binary blob** column. As a result there is no direct way to query profile data from inside of the database. In iConnect.NET, this type of profiling is also supported. Additionally, iConnect.NET adopts a new profiling solution to resolve the issue: **Single Table Profiling**. In the solution, you can customize your profile table schema. In other words, each profile property corresponds to a particular column. Since the profile table schema variates across different application, the stored procedure used for profile data access should changed changed accordingly. These stored procedures can be administratively configured in your application.

This section is dedicated to instruct you how to consume the functionalities provided by iConnect.NET Profile. Starts by a simple walk-through which will step you though a simple ASP.NET web site application which utilizes the basic profile functionalities, we will get a systemic introduction to the usage iConnect.NET Profile. The introduction to usage in distributed scenario will be placed in 3rd part. This section ends with introducing the design overview and profiling extensibility.

18.5.2. Getting Started

In this section, you will walk though a simple ASP.NET website application step by step, which will simulate the basic profiling scenarios: profile setting and retrieval. This walk-though approaches general profile other than single table based profile.

Step I: Create Application Record and Web Application

Similar to Membership Getting start demonstration, we should execute a SQL script insert a record, which stands for the current demonstrative application, into the T_IC_APP. You can get such a script from the Getting Started section of Membership, and we also name our application *SecurityDemo*.

Create a new ASP.NET empty web site using your Visual Studio. In the New Web Site dialog, you must select HTTP as the web location (the default is File System). After the web site is successfully created, add the reference to the assembly NCS.IConnect.Security.Server.dll, which is located in the /bin folder of iConnect.NET installation directory.

Step II: Configure Profile

Add an web configuration file and copy the following XML snippet to override the existing content of the configuration file. Do to forget to revise the connection string accordingly.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <configSections>
   <section name="iConnect.applicationContexts"
            type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                  NCS.IConnect.Core"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         providerName="System.Data.SqlClient"
connectionString="..." />
  </connectionStrings>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  <iConnect.applicationContexts defaultContextLocator="httpSessionStateContextLocator">
    <contextLocators>
      <add name="httpSessionStateContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.HttpSessionStateContextLocator,
                 NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>
  <system.web>
    <anonymousIdentification enabled="true" />
    <profile enabled="true"
             defaultProvider="dbProfileProvider"
             automaticSaveEnabled="true" >
      <properties >
        <add name="Age" type="int" allowAnonymous="true"/>
        <add name="Address" type="string" allowAnonymous="true"/>
      </properties>
      <providers>
        <add name="dbProfileProvider"
             type="NCS.IConnect.Security.DbProfileProvider,
                   NCS.IConnect.Security.Server"
             applicationName="SecurityDemo"
             database="iConnectDb" />
      </providers>
    </profile>
  </system.web>
</configuration>
```

Step III: Add Profile Page

Add a new web page, and design it according the structure illustrated in the following figure. The end user can use this page to view and maintain his or her Age and Address.

Figure 18.1. iConnect.NET Profile Demonstration

The following code snippet represents the three buttons (Save, Get and Clear) click event handler based code. After completing the programming, you can run this web site to view and maintain profile information.

```
protected void ButtonSave_Click(object sender, EventArgs e)
{
    this.Profile.Age = int.Parse(this.TextBoxAge.Text);
    this.Profile.Address = this.TextBoxAddress.Text;
}
protected void ButtonGet_Click(object sender, EventArgs e)
{
    this.TextBoxAge.Text = this.Profile.Age.ToString();
    this.TextBoxAddress.Text = this.Profile.Address;
}
protected void ButtonClear_Click(object sender, EventArgs e)
{
    this.TextBoxAge.Text = string.Empty;
    this.TextBoxAddress.Text = string.Empty;
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.5.3. Configuration

The following XML snippet represents the whole structure of ASP.NET profile, in which the `DbProfileProvider` is configured as the default `ProfileProvider`. For `DbProfileProvider`, we just need to configure the *`applicationName`* and *`database`* (the configuration name of connection string).

```
<system.web>
  <anonymousIdentification enabled="true" ❶/>
  <profile enabled="true"
           defaultProvider="DbGeneralProvider"
           automaticSaveEnabled="true" ❷>
   <properties >❸
    <add name="Age" type="int" allowAnonymous="true"/>
    <add name="Address" type="string" allowAnonymous="true"/>
   </properties>
   <providers>
    <add name="DbGeneralProvider"
        type="NCS.IConnect.Security.DbProfileProvider,
              NCS.IConnect.Security.Server"
         applicationName="SecurityDemo"
         database="iConnectSqlServer" />
   </providers>
  </profile>
  <compilation debug="true">
  </compilation>
</system.web>
```

❶ ASP.NET can profie the two kinds of users: authenticated user and anonymous user. The anonymous user based profile is by default disabled, we must explicitly enable anonymous identification.

❷ A value indicates whether the profile data will be automatically saved. If true, profile data will be automatically saved at the end of the execution of an ASP.NET page, otherwise you have to explicitly call the `Save` method to submit your profile information. The default value is true.

❸ Specify all of profile properties. You can store simple (scalar) values, collections and other complex types, even user-defined types. For detailed information regarding ASP.NET Profile Properties, you can refer to related MSDN online documentation via the URL: `http://msdn2.microsoft.com/en-us/library/d8b58y5d.aspx`.

The following represents the `SingleTableProfileProvider` configuration.

```
<system.web>
  <anonymousIdentification enabled="true" />
  <profile enabled="true" defaultProvider="DbGeneralProvider" automaticSaveEnabled="true" >
    <providers>
        <add name="DbTableProfileProvider"
              type="NCS.IConnect.Security.SingleTableProfileProvider, NCS.IConnect.Security.Server"
              setProcedure="P_IC_USER_INFO_SET_PROPERTIES" ❶
              readProcedure="P_IC_USER_INFO_GET_PROPERTIES" ❷
              deleteInactiveProcedure="P_IC_USER_INFO_DELETE_INACTIVE" ❸
              deleteProcedure="P_IC_USER_INFO_DELETE" ❹
              getInactiveNoProcedure="P_IC_USER_INFO_GET_INACTIVE_NO" ❺
              findProcedure="P_IC_USER_INFO_FIND" ❻
              applicationName="QuickStart"
              connectionStringName="SQLServer"/>
     </providers>
   <properties >
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    <add name="Age"
         type="int"
         allowAnonymous="true"
         customProviderData="p_age;int32;9"❼/>
    <add name="Address"
         type="string"
         allowAnonymous="true"
         customProviderData="@p_address;string;50"/>
   </properties>

  </profile>
  <compilation debug="true">
  </compilation>
</system.web>
```

❶ The stored procedure used to set all of profile properties.

❷ The stored procedure used to get the values of all profile properties.

❸ The stored procedure used to delete all inactive profile entries.

❹ The stored procedure used to delete the profile data specific to the given user.

❺ The stored procedure used to get the number of current inactive users.

❻ The stored procedure used to search profile list base on some query criteria.

❼ Provides the stored procedure with parameter information corresponding to the profile property, its format is "**parameter name; data type; length**"

18.5.4. Customize Profile

Single table based profiling make it possible to customize the profile table and make its schema reflects the profile properties definition. Since the profile schema varies for different applications, related stored procedures need to be manually customized. Assume we need maintain the two profile properties: Age, Address and Phone number, we need to create the profile table and create stored procedures as below:

- Profile table: T_USER_PROFILE

Table 18.1. The schema of T_USER_PROFILE table

Column Name	Data Type	Is Extension Column
USER_ID	VARCHAR	N
AGE	INT	Y
ADDRESS	VARCHAR	Y
PHONE	INT	Y
VERSION_NO	INT	N
TRANSACTION_ID	VARCHAR	N
CREATED_BY	VARCHAR	N
CREATED_TIME	DATETIME	N
LAST_UPDATED_BY	VARCHAR	N
LAST_UPDATED_TIME	DATETIME	N

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Stored procedures

P_IC_USER_INFO_GET_PROPERTIES: The stored procedure is used to get the profile data specific to the user specified.

```
CREATE  PROCEDURE P_IC_USER_INFO_GET_PROPERTIES
(
    @p_app_name             nvarchar(256),
    @p_user_name            nvarchar(256),
    @p_current_time_utc     datetime,
    @p_age                  int output,
    @p_address              nvarchar(50) output,
    @p_phone                int output,   ❶
)
AS
DECLARE  @v_user_id VARCHAR(50)
DECLARE  @v_app_id VARCHAR(50)
DECLARE  @v_error INT

BEGIN
SET NOCOUNT ON
SET XACT_ABORT ON

EXEC @v_error = P_IC_APP_GET_ID @p_app_name, @v_app_id OUTPUT
IF @v_error <> 0 RETURN @v_error

    SET @v_user_id = dbo.F_IC_GET_USER_ID_CORE(@p_app_name, @p_user_name)

    IF @v_user_id IS NULL RETURN

    SELECT @p_address = ADDRESS,@p_age = AGE, @p_phone = Phone
    FROM   T_IC_USER_INFO
    WHERE  [USER_ID] = @v_user_id

    -- update last activity time, no need to update the version and transaction id, etc.
    UPDATE T_IC_USERS
    SET    LAST_ACTIVITY_TIME = @p_current_time_utc
    WHERE  [USER_ID] = @v_user_id

END
GO
```

❶ Each profile property value is retrieved from the stored procedure as a output parameter. In the above stored procedure, the three output parameters correspond to the profile properties: Age, Address and Phone No.

P_IC_USER_INFO_SET_PROPERTIES: The stored procedure is used to update profile information specific to the user specified.

```
CREATE  PROCEDURE P_IC_USER_INFO_SET_PROPERTIES
(
    @p_app_name             nvarchar(256),
    @p_age                  int,
    @p_address              nvarchar(512),
    @p_phone                int,
    @p_user_name            nvarchar(256),
    @p_is_user_anonymous    varchar(1),
    @p_current_time_utc     datetime,
    @p_transaction_id       VARCHAR(50),
    @p_setted_by            NVARCHAR(256),
    @p_return_value         int OUTPUT
)
AS
BEGIN
SET NOCOUNT ON
SET XACT_ABORT ON
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
EXEC @v_error = P_IC_APP_GET_ID @p_app_name, @v_app_id OUTPUT
IF @v_error <> 0 RETURN @v_error

    SET @v_setted_by = dbo.F_IC_CHECK_USER_NAME(@p_setted_by)
    SET @v_user_id = dbo.F_IC_GET_USER_ID_CORE(@p_app_name,@p_user_name)

-- update last activity time, no need to update the version and transaction id, etc.
    UPDATE T_IC_USERS
    SET    LAST_ACTIVITY_TIME = @p_current_time_utc
    WHERE  [USER_ID] = @v_user_id

IF @@ROWCOUNT = 0 RETURN

    IF (EXISTS( SELECT 1 FROM   T_IC_USER_INFO WHERE  [USER_ID] = @v_user_id))
 BEGIN
  UPDATE T_IC_USER_INFO
  SET   AGE = @p_age, ADDRESS = @p_address, TRANSACTION_ID = @p_transaction_id, VERSION_NO =
VERSION_NO + 1,
        LAST_UPDATED_BY = @v_setted_by, LAST_UPDATED_TIME = @p_current_time_utc
  WHERE  [USER_ID] = @v_user_id
 END
 ELSE
  BEGIN
   INSERT INTO T_IC_USER_INFO([USER_ID], AGE, ADDRESS,
TRANSACTION_ID,VERSION_NO,CREATED_BY,CREATED_TIME,LAST_UPDATED_BY,LAST_UPDATED_TIME)
   VALUES
(@v_user_id,@p_age,@p_address,@p_transaction_id,1,@v_setted_by,@p_current_time_utc,@v_setted_by,@p_current_time_utc)

  END
END
GO
```

Tip

iConnect.NET security create a profile table used for single table profile: `T_USER_PROFILE`. You'd better add the new columns in this table to reuse the rest stored procedures which iConnect.NET provides for profile management. In other words, if you reuse the `T_USER_PROFILE`, you just only rewrite the above-mentioned stored procedures.

18.5.5. Profile setting and retrieval

Setting the profile information specific to a particular user, and retrieving the profile information regarding the given user from data source are two basic profile related functionalities. Both in website application and traditional web application, ASP.NET provides strongly typed profile to facilitate programming. The strongly typed profile based programming is implemented by generating the profile class whose structure strictly corresponds to your profile properties configuration.

```
this.Profile.Age = 26;
this.Profile.Address = "Room E101, Airport Rd #328, Suzhou Jiangsu Province, PRC.";

this.lblAge.Text = this.Profile.Age.ToString();
this.lblAddress.Text = this.Profile.Address;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Tip

Because Web Application Projects compile the code behind files in advance of ASP.NET they do not have access to dynamically generated types like the Profile object. You can install a Visual Studio plug in tool named **ASP.Net WebProfile Generator** to gain strongly typed programming support.

You can also adopt a general indexing programming style to operate profile.

```
ProfileBase userProfile = ProfileBase.Create("Foo"❶);
userProfile["Age"] = 26;
userProfile["Address"] = "Room E101, Airport Rd #328, Suzhou Jiangsu Province, PRC.";

this.lblAge.Text = userProfile["Age"];
this.lblAddress.Text = userProfile["Address"];
```

❶ The name of the user to create or retrieve a profile for.

Note

If you set the **AutomaticSaveEnabled** to false, you must explicitly call the Save method to submit the change.

18.5.6. Profile Maintenance

The static ProfileManager class is used to manage profile settings, search for user profiles, and delete user profiles that are no longer in use. The ProfileManager class provides static methods and properties that can be accessed by referencing the ProfileManager class in your application code.

- Gets All Profiles or Inactive Profiles

```
ProfileInfoCollection profiles;
int totalCount;

profiles = ProfileManager.GetAllProfiles(ProfileAuthenticationOption.Authenticated);
profiles = ProfileManager.GetAllProfiles(ProfileAuthenticationOption.All, 0, 10, out totalCount);

profiles = ProfileManager.GetAllInactiveProfiles(ProfileAuthenticationOption.Anonymous,
DateTime.Now.AddDays(-10));
profiles = ProfileManager.GetAllInactiveProfiles(ProfileAuthenticationOption.Anonymous,
DateTime.Now.AddDays(-10), 0, 10, out totalCount);
```

- Find profiles or inactive profile

```
ProfileInfoCollection profiles;
int totalCount;

profiles = ProfileManager.FindProfilesByUserName(ProfileAuthenticationOption.Authenticated, "Foo%");
profiles = ProfileManager.FindProfilesByUserName(ProfileAuthenticationOption.Authenticated, "Foo%",
0, 10, out totalCount);

profiles = ProfileManager.FindInactiveProfilesByUserName(ProfileAuthenticationOption.Authenticated,
"Foo%", DateTime.Now.AddDays(-10));
profiles = ProfileManager.FindInactiveProfilesByUserName(ProfileAuthenticationOption.Authenticated,
"Foo%",
    DateTime.Now.AddDays(-10), 0, 10, out totalCount);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Delete existing profiles

```
ProfileInfoCollection profiles;
int totalCount;

ProfileManager.DeleteProfile("Foo");
ProfileManager.DeleteProfiles(new string[]{ "Foo", "Bar" });
profiles = ProfileManager.FindProfilesByUserName(ProfileAuthenticationOption.All,"Foo%");
ProfileManager.DeleteProfiles(profiles);

ProfileManager.DeleteInactiveProfiles(ProfileAuthenticationOption.All,
ProfileAuthenticationOption.All, DateTime.Now.AddDays(-10));
```

18.6. Site Map

18.6.1. Introduction

Any website that is composed of more than one page needs some sort of navigation user interface. A navigation user interface might be as simple as static hyperlinks to the other pages in the site, or might involve the usage of menus or trees. Prior to ASP.NET 2.0, developers typically rolled their own site navigation solutions. ASP.NET 2.0, however, makes defining a site's structure and implementing it using common navigation user interface elements a walk in the park. Implementing site navigation in ASP.NET 4.0 is a breeze, thanks to the building site navigation features. Internally, ASP.NET offers a programmatic API that enables the site map to be queried. ASP.NET does not require a particular format for specifying the site map, although it does provide a default choice that uses an XML-formatted file. The details on how the site map is serialized can be customized because the site navigation feature of ASP.NET 2.0 uses the provider model. The provider model enables developers to customize the inner workings of a particular ASP.NET subsystem as long as they keep the forward-facing API the same.

You can use ASP.NET 2.0's default XML-based method for specifying your web site's site map or, with a bit of code, you can use your existing, custom method, or some other approach When displaying site navigation, both the `TreeView` and Menu controls use the `SiteMapDataSource` control to read the contents of the site map. Underneath the covers, these controls call ASP.NET 2.0's site navigation API. Since the site navigation piece is implemented using the provider model, the controls are oblivious to the inner details on how the site map is serialized. That is, regardless of whether you use the default site map or roll your own custom site map logic, the navigation controls can be used to work against your site map of choice.

By default, Microsoft provides the `XmlSiteMapProvider` which will use the XML files as data source. But in fact, the most projects will put the menu data in the database. So we customize a site map provider which will retrieve menu data from database. We will store the role information for every menu data in order to do authorization. The customized site map provider can communicate with WCF service. We will also provide some APIs to return menu data as XML format in order to provide data for `XmlSiteMapDataSource` control.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.6.2. Getting Started

In this section, you will walk though a simple ASP.NET website application step by step to build the menu using iConnect.NET SiteMap.

Step I: Prepare Menu Data

Execute the following SQL snippet to insert a record into the T_IC_APP table, and build a simple two level of menu.

```
declare @v_app_id varchar(50)
set @v_app_id=NEWID()
declare @v_current_time_utc datetime
set @v_current_time_utc = getdate()
EXEC P_IC_APP_CREATE_APP 'MenuDemo','*',@v_current_time_utc,@v_app_id output

INSERT INTO T_IC_SITE_MAP
           (SITE_MAP_ID,APP_ID,SITE_MAP_TYPE,SITE_MAP_TITLE,SITE_MAP_DESC,URL,ORDER_BY,

VERSION_NO,TRANSACTION_ID,CREATED_BY,CREATED_TIME,LAST_UPDATED_BY,LAST_UPDATED_TIME,SHOW_IN_MENU)
VALUES ('00bac050-f1e5-4cff-a07c-94ff293d41f11', @v_app_id,'Web','Home','Home','#',2,
        1,'*','test',getdate(),'test',getdate(),1)

INSERT INTO T_IC_SITE_MAP_IN_ROLE
           (SITE_MAP_ID,ROLE_NAME,LOWERED_ROLE_NAME,VERSION_NO,TRANSACTION_ID,
            CREATED_BY,CREATED_TIME,LAST_UPDATED_BY,LAST_UPDATED_TIME)
VALUES ('00bac050-f1e5-4cff-a07c-94ff293d41f11','*','*',1,'*',
        'test',getdate(),'test',getdate())

INSERT INTO T_IC_SITE_MAP
           (SITE_MAP_ID,APP_ID,SITE_MAP_TYPE,SITE_MAP_TITLE,SITE_MAP_DESC,URL,SITE_MAP_PARENT,

ORDER_BY,VERSION_NO,TRANSACTION_ID,CREATED_BY,CREATED_TIME,LAST_UPDATED_BY,LAST_UPDATED_TIME,SHOW_IN_MENU)
VALUES ('2e41f7d4-53b3-45c9-94ad-9cf80d7590001',@v_app_id,'Web','Product Category','Product Category',

'~/Default.aspx','00bac050-f1e5-4cff-a07c-94ff293d41f11',12,1,'*','test',getdate(),'test',getdate(),1)

INSERT INTO T_IC_SITE_MAP_IN_ROLE
          (SITE_MAP_ID,ROLE_NAME,LOWERED_ROLE_NAME,VERSION_NO,TRANSACTION_ID,CREATED_BY,CREATED_TIME,
            LAST_UPDATED_BY,LAST_UPDATED_TIME)
VALUES ('2e41f7d4-53b3-45c9-94ad-9cf80d7590001','*','*',1,'*','test',getdate(),'test',getdate())

INSERT INTO T_IC_SITE_MAP
           (SITE_MAP_ID,APP_ID,SITE_MAP_TYPE,SITE_MAP_TITLE,SITE_MAP_DESC,URL,SITE_MAP_PARENT,

ORDER_BY,VERSION_NO,TRANSACTION_ID,CREATED_BY,CREATED_TIME,LAST_UPDATED_BY,LAST_UPDATED_TIME,SHOW_IN_MENU)
VALUES ('ee2d1ebe-ed8e-44df-a722-20dddbd031441',@v_app_id,'Web','My Shopping Cart','My Shopping Cart',

'~/MyShoppingCart.aspx','00bac050-f1e5-4cff-a07c-94ff293d41f11',14,1,'*','test',getdate(),'test',getdate(),1)

INSERT INTO T_IC_SITE_MAP_IN_ROLE
          (SITE_MAP_ID,ROLE_NAME,LOWERED_ROLE_NAME,VERSION_NO,TRANSACTION_ID,CREATED_BY,CREATED_TIME,
            LAST_UPDATED_BY,LAST_UPDATED_TIME)
VALUES ('ee2d1ebe-ed8e-44df-a722-20dddbd031441','*','*',1,'*','test',getdate(),'test',getdate())

EXEC P_IC_SITE_MAP_MAP_PATH_UPDATE 'MenuDemo',NULL
```

Step II: Create a Web Site and Configure SiteMap

Create a new ASP.NET empty web site using your Visual Studio. In the New Web Site dialog, you must select HTTP as the web location (the default is File System). After the web site is successfully created, add the reference to the assembly NCS.IConnect.Security.Server.dll, which is located in the /bin folder of iConnect.NET installation directory.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Add a web.config and copy the following XML snippet to override the existing content of the configuration file. Do not forget to revise the configuration string to connect the appropriate database.

```
<?xml version="1.0"?>
<configuration>
  <configSections>
    <section name="iConnect.applicationContexts"
             type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                   NCS.IConnect.Core"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         providerName="System.Data.SqlClient"
         connectionString=""/>
  </connectionStrings>
  <iConnect.applicationContexts defaultContextLocator="httpSessionStateContextLocator">
    <contextLocators>
      <add name="httpSessionStateContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.HttpSessionStateContextLocator,
                 NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>
  <system.web>
    <siteMap defaultProvider="dbMenuProvider">
      <providers>
        <clear/>
        <add name="dbMenuProvider"
             type="NCS.IConnect.Security.DbMenuProvider,
                   NCS.IConnect.Security.Server"
             applicationName="MenuDemo"
             menuType="web"
             database="iConnectDb"/>
      </providers>
    </siteMap>
  </system.web>
</configuration>
```

Step III: Add Home Page

Add a new web page, and drag a `SiteMapDataSource` from ToolBox and drop into the page. Drag and drop a `Menu` control into the page, and set the added `SiteMapDataSource` as the DataSource. Run the application, you will see a menu tree is built in the default page.

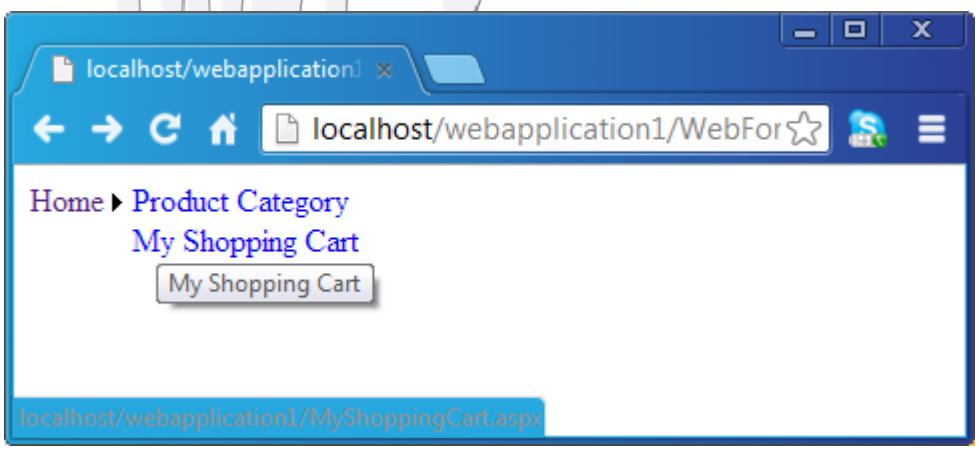


Figure 18.2. Menu Tree

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.6.3. Site Map Data V.S. Menu Data

The menu data is the one that will be displayed in `Menu` control or `TreeView` control. The site map data is used to navigate. So the menu data belongs to site map data. Some data which will not be displayed in the menu control also belongs to site map data. Because some page will not be requested by menu control, but will be requested by clicking button. So in order to distinguish menu data we add the `SHOW_IN_MENU` column in `T_IC_SITE_MAP` table. If the data will be displayed in the menu control, you need to specify the value of `SHOW_IN_MENU` column as "1". But for the `SiteMapMapth` control it will display the site map data not only menu data.

The `URL` of site map data is unique. If you type the same URL it will throw an exception. In the application side, there must be a root node whose `SITE_MAP_PARENT` is null. While inserting a site map data you need to check three columns: `DEPTH`, `PATH_ID` and `MAP_PATH`. The value of `DEPTH` and `MAP_PATH` columns are generated by computing according to the relationship between current site map data and parent site map data. The `DEPTH` is the count of the levels from root node to current node. The root node's `DEPTH` is 0. The `PATH_ID` is an auto increase field. The `MAP_PATH` will connect all `PATH_ID` from root node to current node by ".".

The `ORDER_BY` is used to specify the rank sequence. If the menu data need to be splited by different module, you can specify the `SITE_MAP_TYPE` to distinguish it. In the configuration of site map provider there is a *`menuType`* attribute to specify. While loading data it will read the settings to filter data. If you don't specify the attribute, it will load all site map data under the special application name.

18.6.4. SiteMapDataSource based Binding

If your `Menu` control or `TreeView` control uses `SiteMapDataSource` as data source control, it will bind the menu data from `T_IC_SITE_MAP` to menu control. The `SiteMapDataSource` will wrap every menu data as a `SiteMapNode` instance. The `TreeView` control is constructed by a hierarchical `TreeNode` instance. While binding data to `TreeView` it will put the value of `SiteMapNode`'s property to corresponding `TreeNode`'s property for every record. The default binding relationship is as below:

Table 18.2. The default binding relation

TreeNode Property	SiteMapNode Property	T_IC_SITE_MAP Column
NavigateUrl	Url	URL
Text	Title	SITE_MAP_TITLE
ToolTip	Description	SITE_MAP_DESC

Now if I want to bind a `TARGET` column in `T_IC_SITE_MAP` to `Target` property of `TreeNode`, it seems that the site map node can't work because the `SiteMapNode` don't exist the `Target` property. In order to resolve the problem we define a `ExtendedSiteMapNode` class which inherit from `SiteMapNode` to extend it. For the most column in `T_IC_SITE_MAP` table there is a

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

corresponding property in `ExtendedSiteMapNode` class. You can specify the binding relationship in the `TreeView` control as following steps:

- Right click the `TreeView` control and select "Edit TreeNode DataBindings"
- Click the Add button in `TreeView` Databindings Editor window
- It will display a Data binding properties in the right side
- In the `DateMember` property you type "ExtendedSiteMapNode"
- In the `TextField` property you type "Title"
- In the `NavigateUrlField` property you type "Url"
- In the *`TargetField`* property you type "Target"

The above setting is used to specify binding relationship between `ExtendedSiteMapNode` and `TreeNode`. You can customize it according to your project requirement.

18.6.5. XmlDataSource based Binding?

The above section demonstrates how to bind menu data to `SiteMapDataSource` control. In fact the `XmlDataSource` can also bind menu data to `TreeView` control. The `XmlDataSource` needs the menu data in XML format. So we provide some APIs to return menu data from `T_IC_SITE_MAP` table as XML format. In fact we will retrieve menu data to `T_IC_MENUS` `DataTable` of `MenuDataSet`, then covert all data to xml format. You can specify the binding relationship in the `TreeView` control as following steps:

- Right click the `TreeView` control and select "Edit TreeNode DataBindings"
- Click the Add button in `TreeView` Databindings Editor window
- It will display a Data binding properties in the right side
- In the `DateMember` property you type "T_IC_MENUS"
- In the `TextField` property you type "SITE_MAP_TITLE"
- In the `NavigateUrlField` property you type "URL"

In fact the binding relationship is between columns of `T_IC_MENUS` `DataTable` and `TreeNode's` field property. After creating binding relationship you also need to retrieve data and bind to `TreeView` control. You can refer the following code:

```
XmlDataSource1.Data = Menus.GetXmlMenusByRole("*").InnerXml;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.6.6. SiteMap Structure Maintenance

The API defined in the facade class Menus can be used to maintain the sitemap structure, and the following code snippet lists related methods. The static method UpdateMenus is used to update the sitemap structure based on given sitemap data, which is a strongly typed DataSet (the type is MenuDataSet). The other two methods can be used to get the sitemap data to update. As its names suggests, GetAllMenus method only retrieve the menu based site map data.

```
public static class Menus
{
    //...
    public static MenuDataSet GetAllMenus(string menuType = "*", string siteMapId = null, int depth
= -1);
    public static MenuDataSet GetAllSiteMap(string menuType = "*");
    public static MenuDataSet UpdateMenus(MenuDataSet menuDataSet);
}
```

The following figure represents the structure of MenuDataSet, which contains two tables. All of the sitemap structure data is stored in the T_IC_MENUS table. As with the other table, T_IC_SITE_MAP_IN_ROLE, as its name suggests, store the mapping between site map node and role. It serves for site map authorization.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

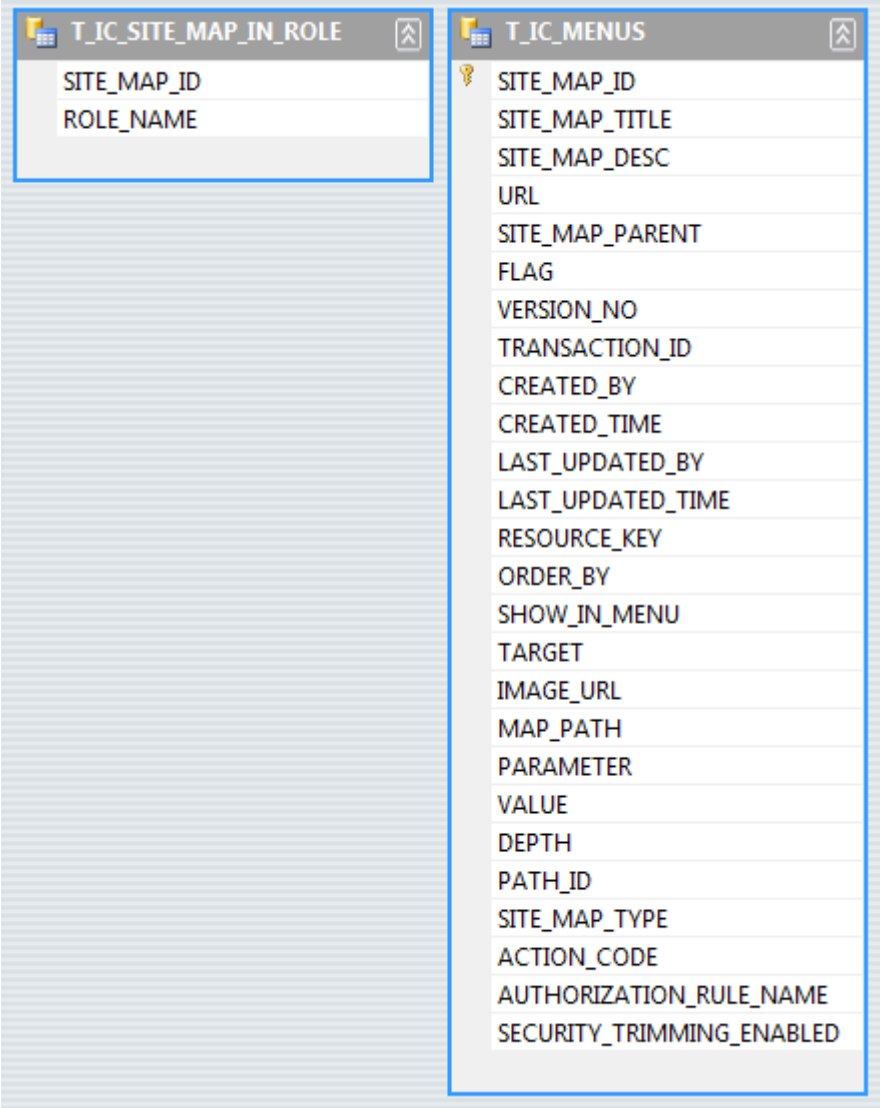


Figure 18.3. structure of MenuDataSet

SiteMap is of a tree-like hierarchical structure, and each node is represented by an SiteMapNode object. There are other methods defined in the facade Menus type, and they retrieve the sitemap node information as a single SiteMapNode object or a SiteMapNodeCollection. As illustrated in the following code snippet, GetAllSiteMapNode method returns the whole sitemap trees as a SiteMapNodeCollection, and GetChildNodes can be used to get all of the child nodes of the given parent node. GetParentNode method gets the parent of the given child node. GetSiteMapNodeByUrl gets the sitemap node by the given URL.

```
public static class Menus
{
    //...
    public static SiteMapNodeCollection GetAllSiteMapNode();
    public static SiteMapNodeCollection GetChildNodes(SiteMapNode node);

    public static SiteMapNode GetParentNode(SiteMapNode node);
    public static SiteMapNode GetSiteMapNodeByUrl(HttpRequest request);
    public static SiteMapNode GetSiteMapNodeByUrl(string applicationPath, string url);
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.6.7. Authorization

The site map also provides the authorization feature. If you want to do authorization, you should put corresponding page into `T_IC_SITE_MAP`. If you don't put it, by default we will do authorization for these pages. You can specify role name for every site map data. The role is stored in the `T_IC_SITE_MAP_IN_ROLE`. We provide some APIs in `Menus` class to assign roles to site map record or remove roles from site map record. The site map will check whether the menu data is allowed to display according the user's role and site map's role. If you want to let every one to access you can set the role name as "*".

It's not safe if only controlling the menu. Somebody can request some pages which are not allowed to access by directly typing the request URL in address bar. In order to resolve the problem we provide a `AuthorizationHttpModule` to prevent the above action. You can register the `AuthorizationHttpModule` in your web.config as below:

```
<system.web>
    <httpModules>
      <add name="HttpModule"
           type="NCS.IConnect.Security.AuthorizationHttpModule,
                 NCS.IConnect.Security.Client"/>
    </httpModules>
<system.web>
```

The `AuthorizationHttpModule` will find the corresponding `SiteMapNode` according to request `URL`, then it will check whether request is allowed to access according to the user's role and the `SiteMapNode's` role. While finding the `SiteMapNode` according to request URL it will ignore the query string of the URL. If you want to do authorization according to different query string, you can use the URL re rewriter. The detailed information of URL re rewriter you can refer as below:
http://weblogs.asp.net/scottgu/archive/2007/02/26/tip-trick-url-rewriting-with-asp-net.aspx.
The AuthorizationHttpModule is compatible with URL re rewriter.

Warning

You need to check the whether the `securityTrimmingEnabled` attribute is set as "true" in the site map provider. If the value is set as "true", it will enable the authorization checking.

18.6.7.1. Role based Authorization

There are two solutions to achieve sitemap authorization. The first one is directly to map a sitemap node with a role. The role assignment can be performed by invoking the following AssignRoles method of the facade Menus. The other two ones (RemoveAllRoles & RemoveRoles) is used to remove the existing sitemap-node-role mapping.

```
public static class Menus
{
    //...
    public static void AssignRoles(string siteMapId, string[] roles);
    public static void RemoveAllRoles(string siteMapId);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public static void RemoveRoles(string siteMapId, string[] roles);
}
```

The following two GetMenus methods return the menu modes which are all authorized by the given user. All of the menu nodes are retrieved in the form of MenuDataSet object.

```
public static class Menus
{
    //...
   public static MenuDataSet GetMenus(IPrincipal principal, string menuType = "*", string siteMapId
= null, int depth = -1);
    public static MenuDataSet GetMenus(string userName, string menuType = "*", string siteMapId =
null, int depth = -1);
}
```

18.6.7.2. Business Action Authorization

Generally speaking, in other authorization scenarios, like UI element authorization, service operation authorization, etc., business action based authorization is better choice. In order to make it possible to use consistent authorization in the whole project, the business action authorization is applied to site map.

As illustrated in the above MenuDataSet structure, there is an ACTION_CODE field in the T_IC_MENU table. This field represents the business action code to which the sitemap node is mapped. Is such a field is not null, it will be always used for authorization, otherwise the pure role based authorization is used.

Warning

The table T_IC_MENU in the MenuDataSet, which also corresponds to the database table, includes an extra SECURITY_TRIMMING_ENABLED field. This field is a sitemap node specific switch to enable and disable authorization. Only if it is specified true, the sitemap node can be authorized. In other words, the sitemap whose SECURITY_TRIMMING_ENABLED is false is always accessible for all users.

18.6.8. Site Map Provider

The site map component provides three providers. You can switch these providers according to every project requirement in the configuration.

- Database Provider

```
<siteMap defaultProvider ="DbSiteMapProvider">
   <providers>
      <clear/>
      <add name="DbSiteMapProvider"
           type="NCS.IConnect.Security.DbMenuProvider,
                 NCS.IConnect.Security.Server"
           applicationName="UnitTest"
           connectionStringName="OraConnString" />
   </providers>
<siteMap>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- WCF provider

```
<siteMap defaultProvider ="WcfProvider">
   <providers>
      <clear/>
      <add name="WcfProvider"
           type="NCS.IConnect.Security.WcfMenuProvider, NCS.IConnect.Security.Client"
           applicationName="UnitTest"
           endpoint="Menu.svc"/>
   </providers>
<siteMap>
```

In a WCF application, above provider configuration is configured in the client side. You also need to configured corresponding WCF service in server side as below:

```
<system.serviceModel>
   <services>
      <service behaviorConfiguration="myBehavior" name="NCS.IConnect.Security.MenuWcfService">
         <endpoint binding="basicHttpBinding"
                   bindingConfiguration="myBasicHttpBinding"
                  contract="NCS.IConnect.Security.IMenu" />
   </services>
</system.serviceModel>
```

18.7. SecureConnect

18.7.1. Introduction

With internet technology development, more and more people prefer looking up information through website, also using software based on web pages over internet. Therefore, it's essential that sometimes users need to pass sensitive data to web server, e.g. financial data, private credentials. Of course, we will say we can encrypt our input in order to avoid hack.

So SecureConnect is used to encrypt data at client side and the encrypted data also can be decrypted at server side. It's guaranteed that the post data will be encrypted and although hackers can intercept the data through special ways, they just can get encrypted data, but it is not easy to be decrypted by hackers.

In detail, at server side SecureConnect will generate RSACryptoServiceProvider based on an existing private key/public key stored in database or memory. At client side, if users want to pass data to server. At first, SecureConnect will sponsor a session based on current Session Id, and generate a RA (Registration Authority) which is used to verify the session is valid, then SecureConnect will generate a CSK based on configuration which is used to encrypt data using AES algorithm. At the same time, the CSK will be encrypted using public key from server; the RA will also be encrypted using CSK. The users will post encrypted data to server with the encrypted CSK, encrypted RA, at server side, SecureConnect will first decrypt the CSK, then using CSK the RA will be recover, and it will be used to verify the session is valid. If the session is valid, the encrypted data will be decrypted using the CSK and RSA private key.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

18.7.2. Getting Started

In this section, you will walk through a simple ASP.NET website application step by step to encrypt/decrypt data using SecureConnect.

Step I: Create a new ASP.NET web application and add assembly reference.

At first, you need to add core assembly "NCS.IConnect.Security.SecureConnect.dll".

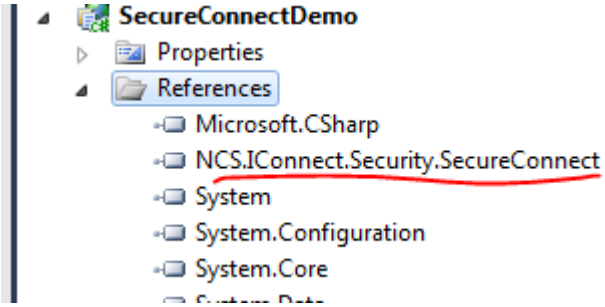


Figure 18.4. Add assembly

Step II: Create necessary controls.

Add two textbox controls, one is used to input plain text, another one is used to display cipher text (encrypted text), this control is read only.

Add two button controls, one is used to trigger encrypt action, another is used to decrypt cipher text.

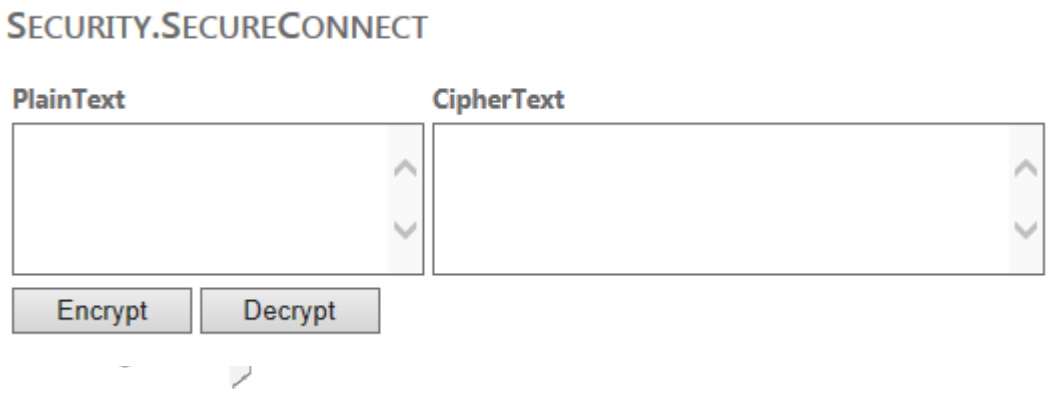


Figure 18.5. Create controls

Step III: Configure SecureConnect.

As quick demo, you can just configure store RA and CSP in memory. For the real usage, can configure them to store in database.

```
<configSections>
  <!-- Security.SecureConnect Configuration -->
  <section name="iConnect.security"
           type="NCS.IConnect.Security.SecureConnect.Configuration.SecuritySettings,
                 NCS.IConnect.Security.SecureConnect"/>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  </configSections>

  <!-- Security.SecureConnect Configuration -->
  <iConnect.security  defaultSecureConnectManager="testing">
    <secureConnectManagers>
      <add name="testing"
           type="NCS.IConnect.Security.SecureConnect.SecureConnectManager,
NCS.IConnect.Security.SecureConnect"
           dwKeySize="1024"
           cspId="8CF39497-1FC7-4DC8-8FC7-875AF8D93A40"
           cspStore="cspInMemoryStore"
           raStore="raInMemoryStore"/>
    </secureConnectManagers>
    <cspStores>
      <add name="cspInMemoryStore"
           type="NCS.IConnect.Security.SecureConnect.CSPInMemoryStore,
NCS.IConnect.Security.SecureConnect"/>
    </cspStores>
    <raStores>
      <add name="raInMemoryStore"
           type="NCS.IConnect.Security.SecureConnect.RegistrationAuthorityInMemoryStore,
NCS.IConnect.Security.SecureConnect"/>
    </raStores>
  </iConnect.security>
```

Step IV: Add necessary code lines to perform encrypt and decrypt.

Due to perform encryption at client side, you need to register JavaScript resources on the page. Then the page needs to inherit SecurityPage class which will help to register necessary JavaScript and internal logic. Finally, Add code lines to perform encryption/decryption.

Add JavaScript:

```
<%--Javascripts required for UI--%>
<script type="text/javascript">
    function Window_Load() {
        $('<%= btnEncrypt.ClientID %>').onclick = OnClientEncrypt;
    }

    window.onload = Window_Load;
</script>
```

Add code logic:

```
using System;
using System.Collections.Generic;
using System.Web.UI;

using NCS.IConnect.Security.SecureConnect.ASPNET;
using NCS.IConnect.Security.SecureConnect;

namespace SecureConnectDemo
{
    public partial class _Default : SecurityPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            this.RegisterControlsToEncrypt(new List<Control> { this.tbxPlainText });
        }

        protected void btnEncrypt_Click(object sender, EventArgs e)
        {
            this.tbxCipherText.Text = this.GetCipherText(this.tbxPlainText);
        }

        protected void btnDecrypt_Click(object sender, EventArgs e)
        {
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            this.tbxPlainText.Text =
SecureConnectManagerFactory.CreateSecureConnectManager().Decrypt(this.GetEncryptedContent(this.tbxPlainText));
        }
    }
}
```

Figure 18.6. Show demo

18.7.3. Programming

CSPStore: it is used to store CSP which include private key and public key. That is, as soon as the application started, SecureConnect will get keys from persistence layer, and construct RSA instance which is used to encrypt/decrypt data. So for different project, the persistence layer is different. iConnect.NET includes two kinds of persistence ways: InMemory (CSPInMemoryStore) and Database (CSPDbStore).

- CSPInMemoryStore: it is used to store CSP in memory, normally it's for testing.
- CSPDbStore: it is used to store CSP in database.

RAStore: RA (registration authority) is used to verify whether the session is valid, generally you need to store RA information into a place; SecureConnect provides memory store and database store.

- RegistrationAuthorityInMemoryStore: it is used to store RA in memory, normally it's for testing.
- RegistrationAuthorityDbStore: it is used to store RA in database.

ISecureConnectManager: the interface is used to generate SecureToken and perform decryption using managed code.

```
public interface ISecureConnectManager
{
    SecureToken GenerateSecureToken();
    string Decrypt(EncryptedContent encryptedContent);
    void ScavengeRA();
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- SecureConnectManager: usually its usage is under the non-distributed environment, e.g. you just needs to do encryption/decryption on one web server.
- WcfSecureConnectManager: you can use this "Manager" to do encryption/decryption under the distributed environment.

SecureConnectManagerFactory: Generally, above are internal objects, developers don't care them, as façade class, developers just use SecureConnectManagerFactory to create ISecureConnectManager instance to do decryption, because encryption is done at client side.

SecurityPage: web page needs to inherit this class in order to register necessary JavaScript and hidden fields.

PageExtension: SecureConnect provides some useful *extension methods* which help developers to register controls to be ready to encrypt, get encrypted data etc.

- static void RegisterControlsToEncrypt(this Page page, List<Control> controlsToEncrypt)

  It's used to register controls to encrypt at client side.

- static EncryptedContent GetEncryptedContent(this Page page, Control control)

  It's used to get encrypted data including encrypted sensitive data, encrypted CSK, and encrypted RA.

- static string GetCipherText(this Page page, Control control)

  It's used to get cipher text by the specific control.

JavaScript: SecureConnect embeds many JavaScript resources to do encryption at client side. Just as the following scripts:

```
//function to allow pages to encrypt the controls before postback
function OnClientEncrypt() {
    // Create a Confidential Level Session Key, CSK to use for encryption
    var cskBytes = CreateCsk();❶

    // Encrypt the generated CSK and RA.
    EncryptCskandRa(cskBytes);❷

    var clientIdsForEncryption = $('__securitymanagerClientIdsForEncryption').value;

    // For each input that requires encryption, encrypt them with CSK
   // and disable the input containing the value to be encrypted so that it is not posted back in clear
    if (clientIdsForEncryption != null && clientIdsForEncryption != '') {
        var clientIds = clientIdsForEncryption.split('|');

        for (var i = 0; i < clientIds.length; i++) {
            var client = $(clientIds[i]);

            if (client != null && client.value != null && client.value != '') {
                var hiddenEncrypted = $('__securitymanager_' + clientIds[i] + '_encrypted');

                hiddenEncrypted.value = AesEncrypt(client.value, cskBytes);❸
              // Note: after encrypting value, it's better to disable the control so that the plain
value won't be post.
                client.setAttribute('disabled', true);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            }
        }
    }
}
```

❶ CreateCsk(): create a new CSK.

❷ EncryptCskandRa(cskBytes): encrypt CSK and RA according to the generated CSK.

❸ AesEncrypt(client.value, cskBytes): encrypt data according to the generated CSK.

18.7.4. Configuration

```
<configSections>
    <!-- Default Database Setting -->
    <section name="dataConfiguration"
type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
Microsoft.Practices.EnterpriseLibrary.Data" />
    <!-- Security.SecureConnect Configuration -->
    <section name="iConnect.security"
type="NCS.IConnect.Security.SecureConnect.Configuration.SecuritySettings,
NCS.IConnect.Security.SecureConnect"></section>❶
  </configSections>

  <!-- Database Connection String -->
  <connectionStrings>
    <add name="SqlServerDatabase" connectionString="Server=.\SQLExpress; Database=SecureConnectDB;
Uid=[user name]; Pwd=[password]" providerName="System.Data.SqlClient" />
  </connectionStrings>

  <!-- Default Database Setting -->
  <dataConfiguration defaultDatabase="SqlServerDatabase" />

  <!-- Security.SecureConnect Configuration -->
  <iConnect.security  defaultSecureConnectManager="testing">
    <secureConnectManagers>
      <add name="default"
           type="NCS.IConnect.Security.SecureConnect.SecureConnectManager,
NCS.IConnect.Security.SecureConnect"❷
           dwKeySize="1024"
           cspId="FACEC53D-9EFF-4F26-8976-41E14C7A0152"
           cspStore="cspDbStore"
           raStore="raDbStore"/>
      <add name="testing"
           type="NCS.IConnect.Security.SecureConnect.SecureConnectManager,
NCS.IConnect.Security.SecureConnect"
           dwKeySize="1024"
           cspId="8CF39497-1FC7-4DC8-8FC7-875AF8D93A40"
           cspStore="cspInMemoryStore"
           raStore="raInMemoryStore"/>
      <add name="remoteservice"
           type="NCS.IConnect.Security.SecureConnect.WcfSecureConnectManager,
NCS.IConnect.Security.SecureConnect"❸
           endpoint="WcfSecureConnectManagerService"/>
    </secureConnectManagers>
    <cspStores>
      <add name="cspDbStore"
          type="NCS.IConnect.Security.SecureConnect.CSPDbStore, NCS.IConnect.Security.SecureConnect"❹
           database="SqlServerDatabase"❺/>
      <add name="cspInMemoryStore"
           type="NCS.IConnect.Security.SecureConnect.CSPInMemoryStore,
NCS.IConnect.Security.SecureConnect"❻/>
    </cspStores>
    <raStores>
      <add name="raDbStore"
           type="NCS.IConnect.Security.SecureConnect.RegistrationAuthorityDbStore,
NCS.IConnect.Security.SecureConnect"❼
           database="SqlServerDatabase"❽/>
      <add name="raInMemoryStore"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            type="NCS.IConnect.Security.SecureConnect.RegistrationAuthorityInMemoryStore,
NCS.IConnect.Security.SecureConnect"❾/>
    </raStores>
  </iConnect.security>
```

❶ SecureConnect configuration section.

❷ The built-in SecureConnectManager to handle CSPStore and RAStore, also configure dwKeySize which is used by encryption/decryption, and CSPId is used by CSPStore.

❸ The built-in WcfSecureConnectManager to construct distributed service, developers can use this "Manager" to handle encryption/decryption using wcf service.

❹ To store CSP in database.

❺ Connection string name is used to connect database.

❻ To store CSP in memory.

❼ To store RA in database.

❽ Connection string name is used to connect database.

❾ To store RA in memory.

18.7.5. Best Practice

Although iConnect.NET provides so many "Store", "Manager" to perform end to end security feature, you still need to have a good configuration or combination to reach your best secure result.

At first, we need to have a clear understanding to know what's the secure action in the application.

In web applications, for sensitive data, we won't let anybody intercept the plain content anytime. It means that we have to keep data protected at the beginning or user input step, meanwhile every pipeline (connection) keeps data protected as well.

So, we draw a picture to illustrate the secure process as the following one.

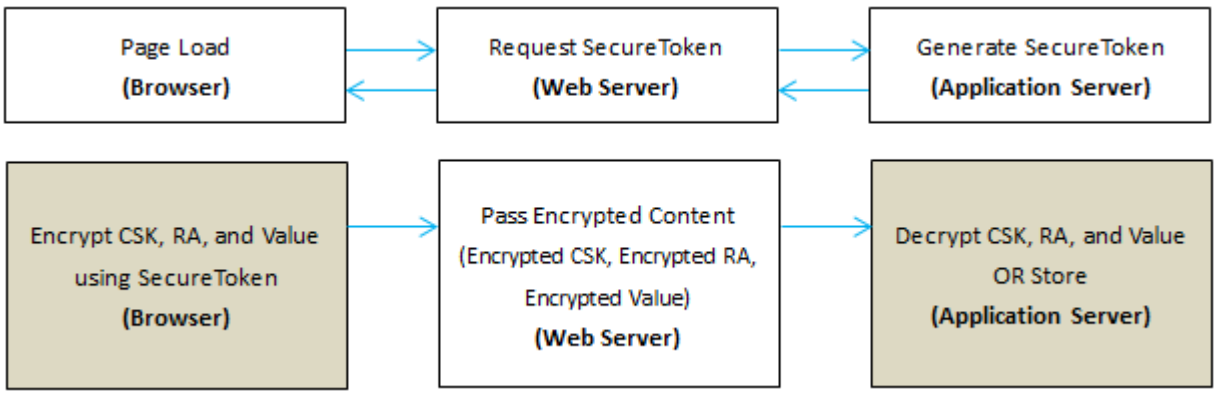


Figure 18.7. Secure Process

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

From above picture, it assumed that there are Web Server and Application Server, of course, it also includes user input side (browser). Then we'll illustrate the process steps one by one:

- *Page Load*: when the page loads, it will request SecureToken through Web Server at first.
- *Request SecureToken*: actually the action to request SecureToken happened at Application Server, Web Server just have a request to Application Server.
- *Generate SecureToken*: Application Server will generate SecureToken which includes PublicKey, Registration Authority and send back to client.
- *Encrypt CSK, RA, and Value using SecureToken*: when the page wants to handle business, it will generate CSK, encrypt CSK, RA and Value using SecureToken, all of these actions are just done by client/browser not related to Web Server.
- *Pass Encrypted Content (Encrypted CSK, Encrypted RA, Encrypted Value)*: it's noted that Web Server doesn't take responsibility for decrypt value, it just pass encrypted content to Application Server.
- *Decrypt CSK, RA, and Value OR Store*: Application Server received the encrypted content, one case is to decrypt CSK, RA and Value, or store the encrypted Value and CSK in database or other persistence layer.

According to the secure process, we find that the plain Value to handle just appears at client/browser or Application Server, at transfer pipeline or Web Server, it's always kept protected. Obviously it will be secure better.

As for how to implement code using the above secure process, iConnect.NET QuickStart "Security"->"End to end encryption" has a good sample to demonstrate it as your reference.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 19. Batch Job

19.1. Introduction

Batch Job is used to schedule and run jobs at per-determined time in the scheduler server. Except for the basic functionalities, job scheduling and execution, other related functionalities are also supported, E.G. Viewing the job status and running history, Notification and Failover, etc.

19.2. Getting Started

In this section,you will walk though a simple application step by step, which will simulate the basic batch job related work, including: Create a custom job; Schedule and run with batch job tool provided by iConnect.NET.

Step I: Create a Custom Job

Create a Class Library project using your Visual Studio, and name this newly created project "*MyBatchJob*". Then add the assembly reference to NCS.IConnect.BatchJob,which is located in the /bin folder of the iConnect.NET installation directory. Add a new .cs file to define such following custom job, `ExecutableBatchJob`.

```
using System;
using System.Threading;
using NCS.IConnect.BatchJob;
namespace MyBatchJob
{
    public class ExecutableBatchJob : BatchJobBase
    {
        protected override void DoWork()
        {
            try
            {
                for(int i=0;i<100;i++)
                {
                    Percentage = i;
                    Thread.Sleep(500);
                }
                AddNote("This is batch job demo");
            }catch(Exception ex)
            {
                SendMessage(ex.Message,true);
            }
        }
    }
}
```

Step II: Deploy and Schedule the Created Job

Compile the above project and copy the built assembly, `MyBatchJob.dll` into the iConnect.NET installation directory. Than launch iConnect.NET studio, and open the batch job type maintenance window by selecting the menu *Tools/Batch Job/Maintain Batch Job Type*. The following *Batch Job Type Add* window will be pop-up after clicking the "Add" button. Specify

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

the appropriate type name and assembly qualified name for the custom job created above, and then click the *Add* button.

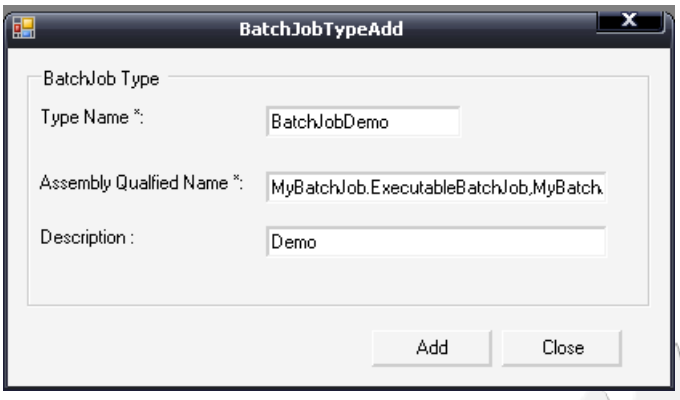


Figure 19.1. Add batch job type

After the batch job type is successfully added we can leverage iConnect.NET studio to schedule it. Select the menu item *Tools/Batch Job/Maintain Batch Job* to show the batch job maintenance window. The following Add *Batch Job* dialog box will be shown after the *Add* button clicked. You can see the batch job type created above from the *Batch Job Type* `ComboBox`. You need to specify a unique job name, and set the scheduling.

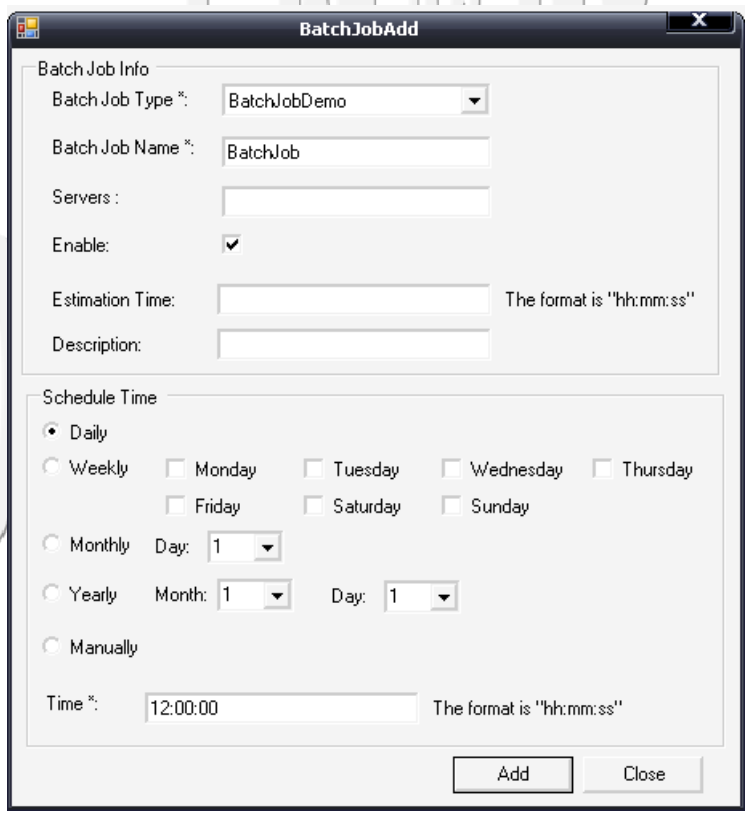


Figure 19.2. Add batch job

Step 3: Run the Scheduled Batch Job

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

So far, all of the preparation are done, you can run the batch job by launching the executable of NCS.IConnect.BatchJob.Executor.exe , which is resided in the installation directory of iConnect.NET. In this demonstration application, we will launch the executable file from Visual Studio. If the project created in step 1 is closed, re-open it. Right click the project in the solution explorer window, and select "Properties" option in the pop-up context menu. Activate the Debug table in the Project Property windows and specify the path of `NCS.IConnect.BatchJob.Executor.exe` as the "Start external program" as below. In addition, specify the "Command line arguments" as "fb07b0ad-3118-4775-b4ff-60fa05ec5c58 Manually {Version No}".

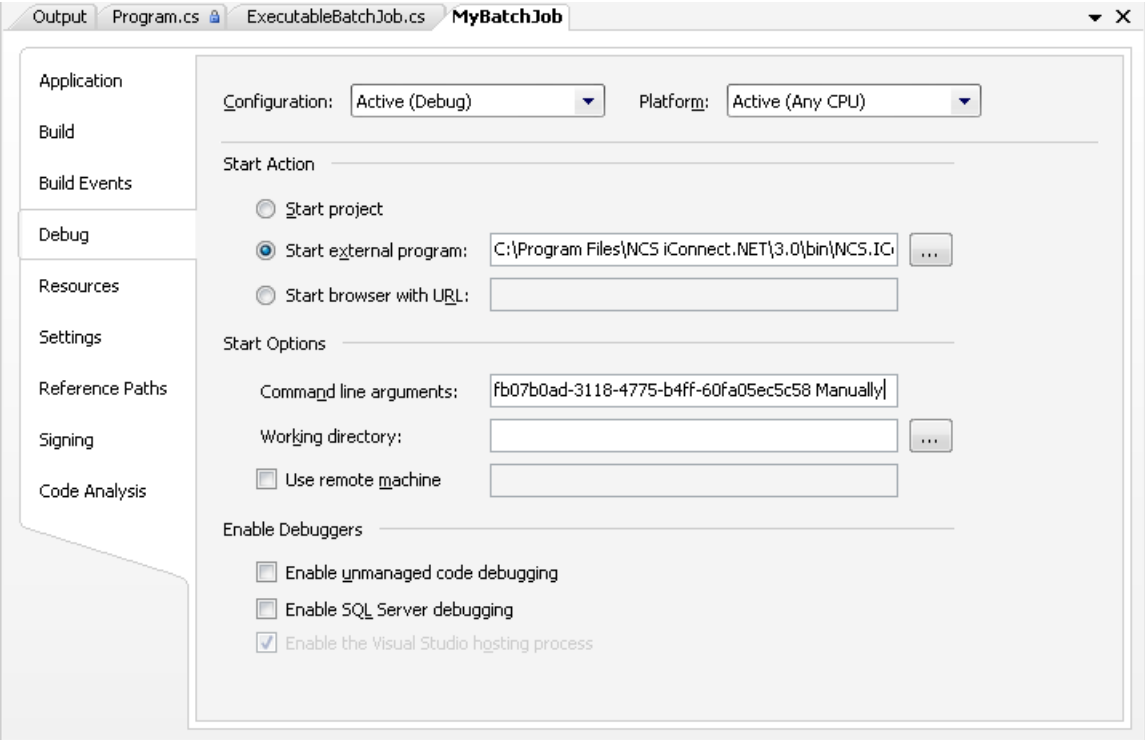


Figure 19.3. MyBatchJob property settings

Set `MyBatchJob` project as the startup project, you will get the following output from console after running the project.

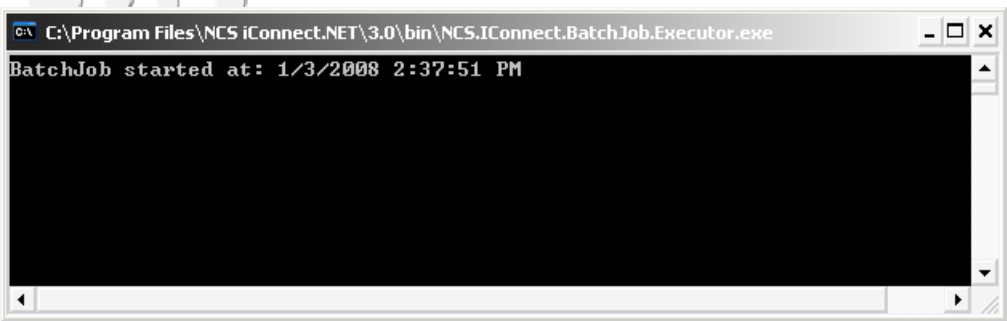


Figure 19.4. My own job is running

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Keep the application running, you can view the running history information and check the status of each job from Batch Job History Viewer of iConnect.NET studio. You can show this by selecting Tools/Batch Job/Maintain Batch Job History.

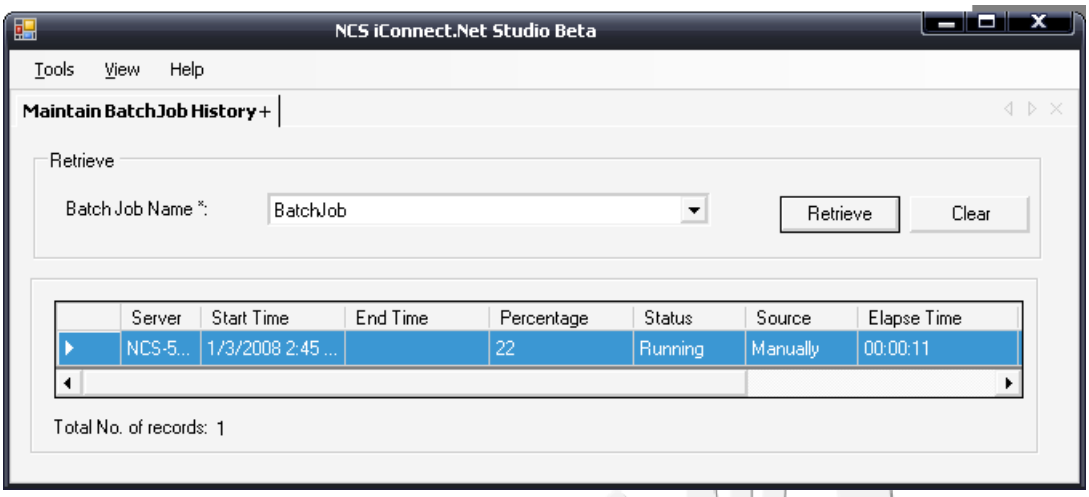


Figure 19.5. Batch job running status

19.3. Scheduling Service

A job can be either executed at a determined time once, or executed at regular intervals. For example, we can schedule a job and make it executed at certain time every day, week or month. We call such jobs as Daily Job, Weekly Job and Monthly Job. The above Getting-Started application demonstrates how to manually run batch job, however, in most situations, the batch job need to be executed in an automatic fashion. iConnect.NET leverages a Windows Service to load the scheduling setting and automatically launch job according to the setting.

So, when consuming iConnect.NET Batch Job, you firstly need to guarantee such a Windows Service, whose name is "*iConnect Batch Job Service*", is successfully started. You can check the status of the batch job service in Windows Service Manager if it is installed. Generally speaking, this service is installed in the progress of iConnect.NET installation. If you cannot get the service entry from Service Manager Window, you need to manually install it. You can execute the `InstallUtil` command line to install iConnect.NET batch job service like this:

```
InstallUtil "C:\Program Files\NCS iConnect.NET\{version no}\bin\NCS.IConnect.BatchJob.Service.exe"
```

Before introducing the components involved in batch job, let's first take a glance at the physical architecture of Batch Job (note that the schedule service server side will monitor all the batch jobs and run them automatically):

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

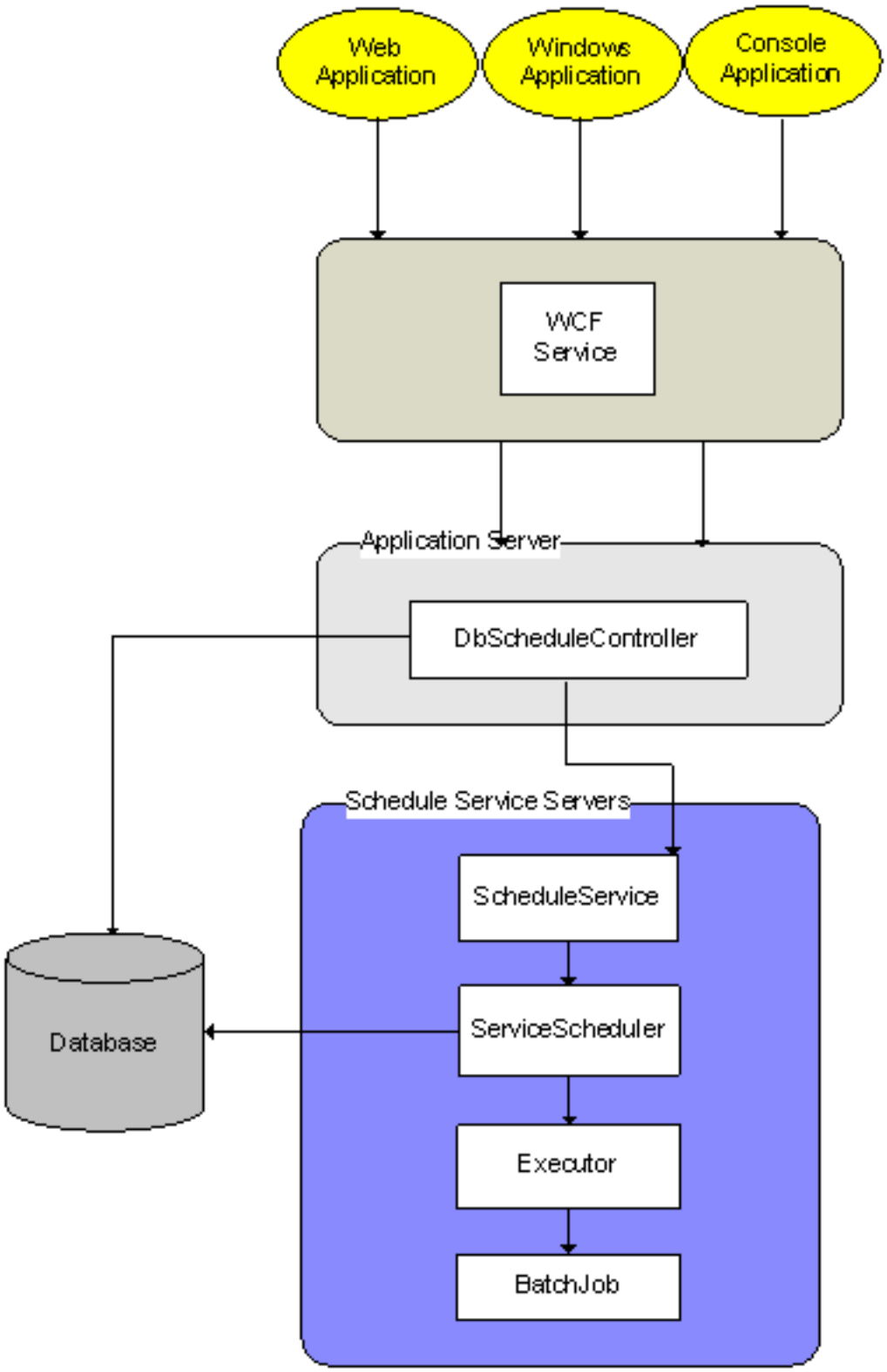


Figure 19.6. Physical batch job architecture

The following 4 kinds of services or components are hosted at schedule server: `SchduleService`, `ServiceScheduer`, `Executor` and `BatchJob`. The interaction between them is illustrated in the following sequence diagram.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

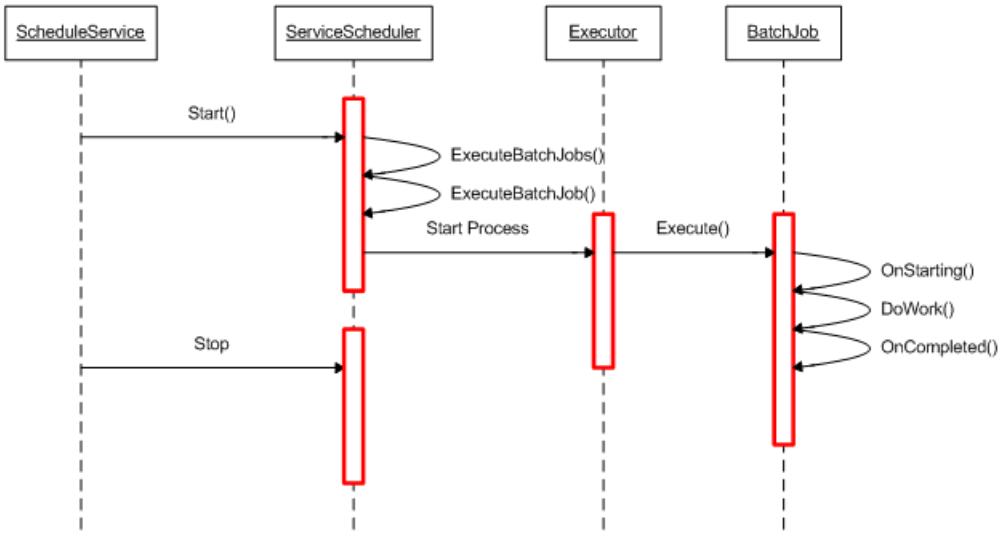


Figure 19.7. Sequence diagram in scheduler server

`ServiceScheduler` start a single process to run each job. Once the batch job windows service launched, the `Start` method of `ServiceScheduler` class will be invoked to start a scheduler. Client code can use `Stop` method to stop the scheduler. In addition, there are also many other methods defined in `ServiceScheduler` class, which can be used to control the service scheduler.

iConnect.NET provides us a set of configuration to perform setting for batch job and batch job service. The schema of the configuration is illustrated in the following code snippet.

```
<configuration>
   <configSections>
      <section name="iConnect.batchJob"
               type="NCS.IConnect.BatchJob.Configuration.BatchJobSettings,
                     NCS.IConnect.BatchJob" />
      <section name="iconnect.batchjob.service"
               type="NCS.IConnect.BatchJob.Configuration.BatchJobServiceSettings,
                     NCS.IConnect.BatchJob" />
   </configSections>

   <iConnect.batchJob defaultScheduleController="dbScheduleController"❶
                      executorPath="C:\Program Files\NCS iConnect.NET\4.3\bin"❷
                      defaultSystemUser="systemUser"❸>
      <scheduleControllers>
        <add name="dbScheduleController"
             type="NCS.IConnect.BatchJob.DbScheduleController,
                   NCS.IConnect.BatchJob.Server"
            applicationName=”Demo”
            database="iConnectSqlServer"/>
      </scheduleControllers>
   </iConnect.batchJob>
   <iconnect.batchjob.service schedulerInterval="30"❹
                              failOverInterval="180"❺
                              enableNotification="true"❻
                              handleFailover="true"❼/>
</configuration>
```

❶ The name of default `ScheduleController`, which is configured in the <scheduleControllers/> configuration section.

❷ The path of the iConnect.NET batch job executor.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❸ The name of system user account

❹ The interval between two refresh operations, which is used to get the current job scheduling information.

❺ The batch job's failover interval.

❻ A Boolean value indicating whether to send notification when the batch jobs don't run at the scheduled time

❼ A Boolean value indicating whether to run the batch jobs which haven't run at the scheduled time.

Tip

The attribute "*`enableNotification`*" depends on `Logging Application Block` to send the notification, so please add logging settings and add email trace listener for "`BatchJob`" category for notification purpose and "`TraceBatchJob`" category for trace purpose. If want to get more information about logging, we can refer `Microsoft Enterprise Library Logging Application Block`.

19.4. ScheduleController

In iConnect.NET Batch Job, most of batch job management operations are controlled by `ScheduleController`, which is an abstract type implementing `IScheduleController` interface. `ScheduleController` is the base type of all of concrete schedule controller. There are totally two built-in `ScheduleController` defined in iConnect.NET, and they are `DbScheduleController` and `WcfScheduleController`.

19.4.1. Database Schedule Controller

`IDbScheduleController` manages batch job by directly getting access to database, and another one needs to invocate WCF service. The following code snippet describes the schema of `DbScheduleController` based configuration. As with the implication of involved configuration properties, refer to 3.1.

```
<configuration>
   <configSections>
        <section name="iConnect.batchJob"
                 type="NCS.IConnect.BatchJob.Configuration.BatchJobSettings,
                       NCS.IConnect.BatchJob" />
   </configSections>
   <iConnect.batchJob defaultScheduleController="dbScheduleController"
                      executorPath="C:\Program Files\NCS iConnect.NET\{version no}\bin"
                      defaultSystemUser="systemUser">
      <scheduleControllers>
         <add name="dbScheduleController"
              type="NCS.IConnect.BatchJob.DbScheduleController,
                    NCS.IConnect.BatchJob.Server"
            applicationName="Demo" database="iConnectSqlServer"/>
      </scheduleControllers>
  </iConnect.batchJob>
</configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

19.4.2. WCF Schedule Controller

`WcfScheduleController` is used in distributed scenario with WCF as the communication infrastructure. The following illustrates the `WCFScheduleController` related configuration at client side. The configuration property of endpoint indicates the configuration name of endpoint to consume the batch job WCF service.

```
<add name="wcfScheduleController"
     type="NCS.IConnect.BatchJob.WcfScheduleController,
           NCS.IConnect.BatchJob"
     applicationName="Demo"
     endpoint="batchSchedulerService" />
```

At client side, we need to add the above configuration, and do necessary WCF configurations; At server side, we also need to do necessary WCF configurations, and add the *`dbScheduleController`* configuration (refer 2.2.2), in addition, we also need to add one file called `BatchJob.svc`, the content of it is:

```
<%@ ServiceHost language="C#" Service="NCS.IConnect.BatchJob.ScheduleControllerWcfService" %>
```

19.5. Get Job Information

You can call appropriate methods of facade class `BatchJobManager` to get job related information.

```
IList<BatchJobTypeInfo> jobType           = BatchJobManager.GetAllJobTypeInfo();❶
IList<BatchJobInfo> jobInfo               = BatchJobManager.GetAllJobInfo();❷
IList<BatchJobParameter> jobParameter     = BatchJobManager.GetParameters(jobId);❸
IList<BatchJobRunningInfo> jobRunningInfo = BatchJobManager.GetJobHistoryByJobId(jobId);❹
```

❶ Get all available batch job types.

❷ Get all batch job info.

❸ Get the parameter by the batch job id.

Tip

The batch job can run with different parameters. We can use parameters in the our own batch job business logic according to the requirements. (If want to add parameters for one batch job, we can use studio function)

❹ Get the history information by the batch job id.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 20. File Uploader

20.1. Introduction

In some projects there would be a requirement to perform data migration task. Extract data from some sources, E.G. Excel, Txt, Csv and so on. And save the extracted data to database.

iConnect.NET File Uploader is used to extract data from source files, and upload the data to database. It standardize extracting interface, provide predefined extract components include **Csv Extract Component**, **Excel Extract Component**, **Text Extract Component**, and provide a GUI based configuration tool to configure the column mappings, the mapping files can be saved and reloaded for task executing also.

20.2. Getting Started

Assume that we have the following excel file:

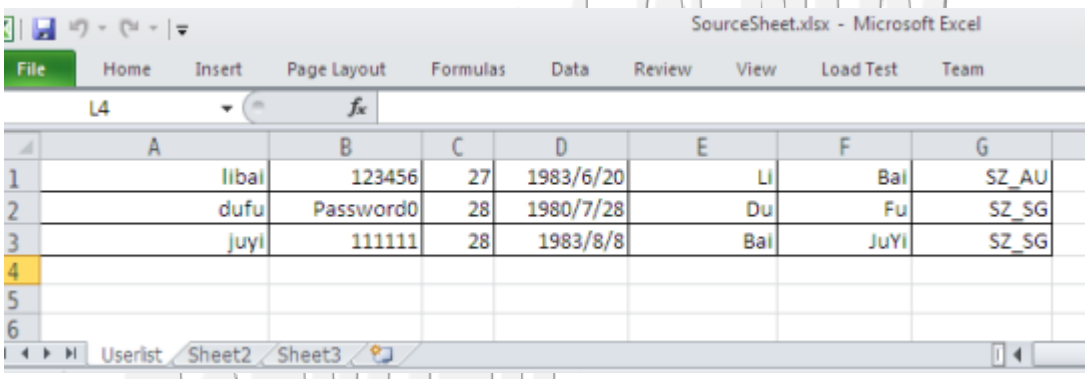


Figure 20.1. excel souce file

and we need to migrate them to the following database table:

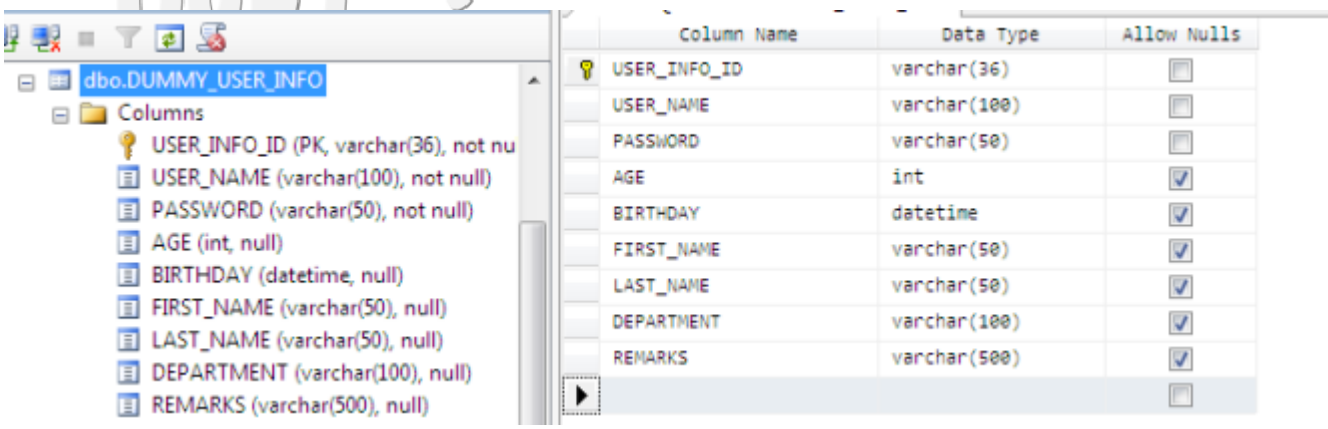


Figure 20.2. destination databse table struct

Notes: you can create the sample table by following schema:

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
CREATE TABLE DUMMY_USER_INFO (
 [USER_INFO_ID] VARCHAR(36) PRIMARY KEY,
 [USER_NAME] VARCHAR(100) NOT NULL,
 [PASSWORD] VARCHAR(50) NOT NULL,
 [AGE] INT, [BIRTHDAY] DATETIME,
 [FIRST_NAME] VARCHAR(50),
 [LAST_NAME] VARCHAR(50),
 [DEPARTMENT] VARCHAR(100),
 [REMARKS] VARCHAR(500)
)
```

Following list described the detail business rule that will be used in the sample:

1. "*USER_INFO_ID*" : we will use system generated GUID.

2. "*USER_NAME*" : import from excel column `A` (User Name).

3. "*PASSWORD*" : import from excel column `B` (Password).

4. "*AGE*" : import from excel column `C` (Age).

5. "*BIRTHDAY*" : import from excel column `D` (Birthday).

6. "*FIRST_NAME*" : import from excel column `E` (First Name).

6. "*LAST_NAME*" : import from excel column `F` (Last Name).

7. "*DEPARTMENT*" : import from excel column `G` (Department).

8. "*REMARKS*" : content will be kept as NULL because there is no corresponding column in the original excel.

The following steps guide you how to extract the data from the excel source and upload them to database table:

- Step 1 : Create a new blank solution and add the following projects in it:
  - Client : This is a *Console Application* project used to execute the file upload task. This project should reference **NCS.IConnect.Core** and **NCS.IConnect.FileUploader** .

    Create two folder under the Client project "Source\SourcePath" , "Source\DTSettingPath". Put the "SourceSheet.xlsx" under "Source\SourcePath" and set the property "Copy to Output Directory" as "Copy always".

  - Entity : This is a *Class Library* project that is used for configure column mapping source columns with database table columns. Entity project is referenced by Client project.
- Step 2 : Create a dataset named "*DataSetDummyUserInfo*" for the database table, drag table "DUMMY_USER_INFO" from "Server Explorer" to this dataset. Build out the dll of the Entity project and copy it to bin folder under iConnect.NET installation folder.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Step 3 : Open **NCS.IConnect.FileUploader.DTSettingTool** from bin folder under iConnect.NET installation folder. Select the assembly for the dataset "Entity" in "Data table mapping" region. Configure "Source reader option", "Data table mapping", "Data import extension" parts as the following figure.

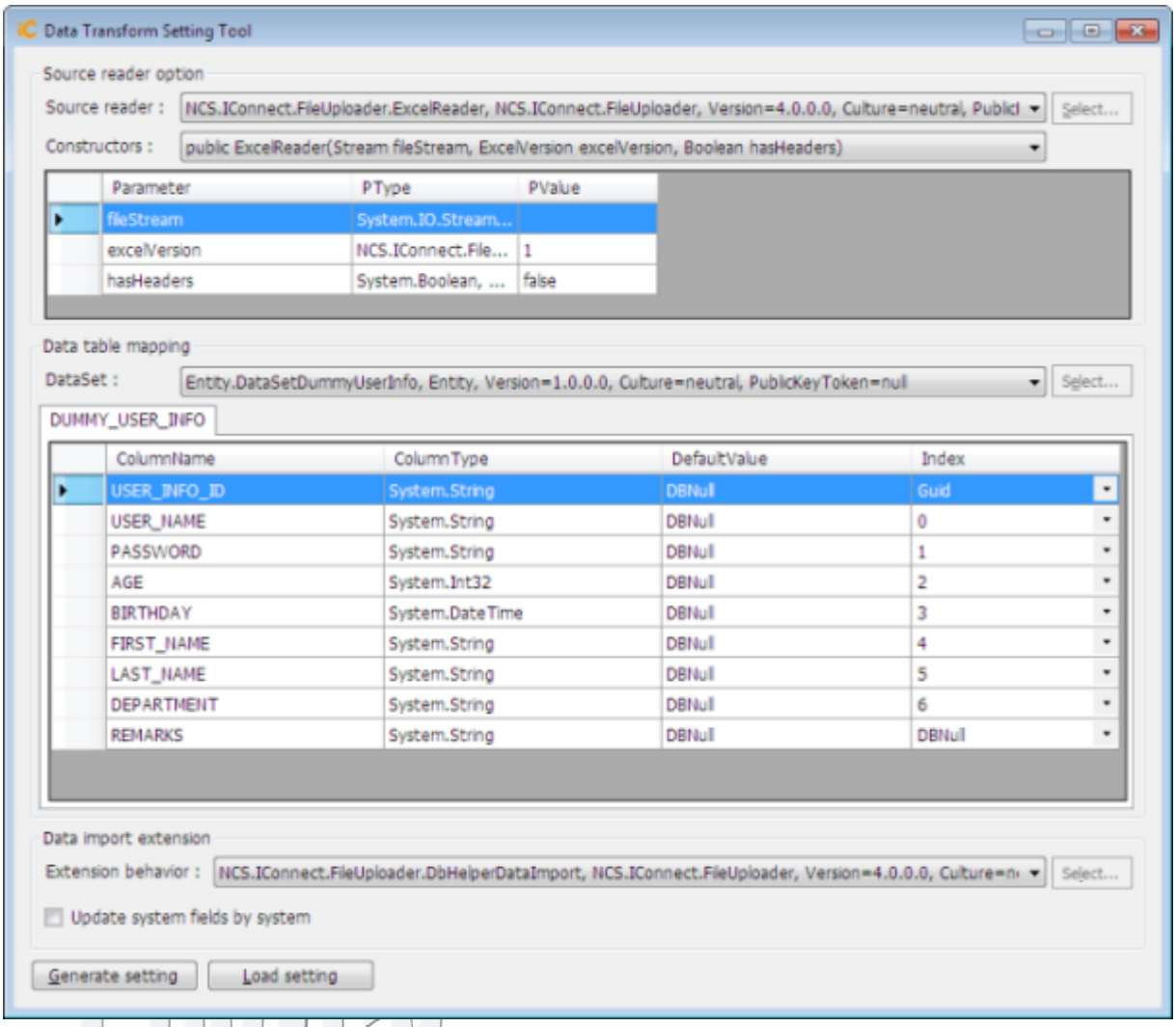


Figure 20.3. File Uploader configuration detail

Click "Generate setting" to generate "EntitySetting.xml" saved in the "Source\DTSettingPath" created in Step 1. Include it in project, set the property "Copy to Output Directory" as "Copy always".

The following figure illustrates the structure of the whole demonstration solution:

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

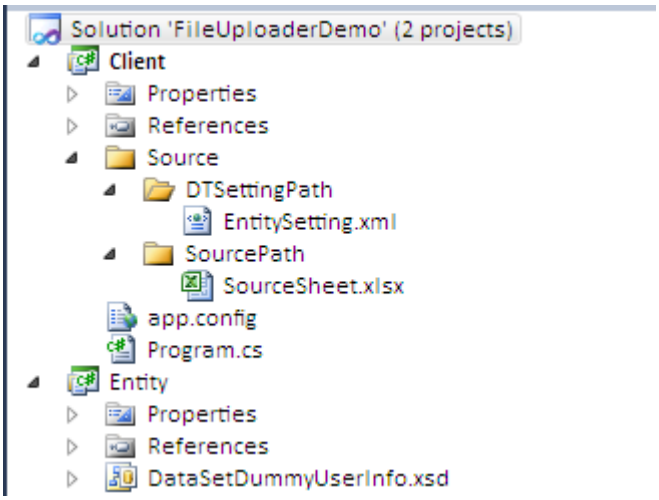


Figure 20.4. The structure of the File Uploader demonstration solution

- Step 4 : Use **NCS iConnect.Net Studio** to generate the standard insert script for table DUMMY _USER_INFO and execute in the target database.
- Step 5 : Create other 3 folders in the Client project output folder : "Source\SuccessPath", "Source\ErrorPath", "Source\BackupPath".
- Step 6 : Configure the app.config for the application. Source location configuration has the following four parts:

  dtSettingPath : used to get the setting file.

  sourcePath : used to get source file.

  backupPath : used as backup file path.

  successPath : when task executed successfully, there will be a source file copy here.

  errorPath : when task executed but encounter error, there will be a source file copy here.

```
<configuration>
  <configSections>
    <section name="dataConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
                   Microsoft.Practices.EnterpriseLibrary.Data"/>
    <section name="iConnect.dataAccess"
             type="NCS.IConnect.Helpers.Data.Configuration.DataAccessSettings,
                   NCS.IConnect.Helpers.Data"/>
    <section name="iConnect.fileUploader"
             type="NCS.IConnect.FileUploader.Configuration.FileUploaderServiceSettings,
                   NCS.IConnect.FileUploader"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectSqlServer"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  <dataConfiguration defaultDatabase="iConnectSqlServer"/>

  <!-- iConnect.dataAccess -->
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  <iConnect.dataAccess>
    <systemFields>
      <add name="VersionNo"  fieldName="VERSION_NO" />
      <add name="TransactionId"  fieldName="TRANSACTION_ID" />
      <add name="CreatedBy"  fieldName="CREATED_BY" />
      <add name="CreatedTime"  fieldName="CREATED_TIME" />
      <add name="LastUpdatedBy"  fieldName="LAST_UPDATED_BY" />
      <add name="LastUpdatedTime"  fieldName="LAST_UPDATED_TIME" />
    </systemFields>
    <commands>
      <add name="Select" format="P_{TableName}_S" />
      <add name="Delete" format="P_{TableName}_D" />
      <add name="Update" format="P_{TableName}_U" />
      <add name="Insert" format="P_{TableName}_I" />
    </commands>
    <parameters>
      <add name="Current" format="p_{DataColumnName}" />
      <add name="Original" format="o_{DataColumnName}" />
    </parameters>
    <schemaMappings>
      <add name="dataTableName" format="{TableName}" />
      <add name="dataColumnName" format="{DataColumnName}" />
    </schemaMappings>
  </iConnect.dataAccess>
  <!-- File Uploader -->
  <iConnect.fileUploader defaultProvider="DbProvider">
    <fileUploaderProviders>
      <add name="DbProvider"
           type="NCS.IConnect.FileUploader.FileUploaderDbProvider,
                 NCS.IConnect.FileUploader"
           applicationName="WingHingApp"
           database="iConnectSqlServer"/>
    </fileUploaderProviders>
    <sourceLocation❶ dtSettingPath="Source\DTSettingPath\"
                     sourcePath="Source\SourcePath\"
                     backupPath="Source\BackupPath\"
                     successPath="Source\SuccessPath\"
                     errorPath="Source\ErrorPath\"/>
  </iConnect.fileUploader>

  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>
</configuration>
```

❶ Source location configuration:

dtSettingPath : used to get the setting file.

sourcePath : used to get source file.

backupPath : used as backup file path.

successPath : when task executed successfully, there will be a source file copy here.

errorPath : when task executed but encounter error, there will be a source file copy here.

- Step 7 : the C# code for the demo is as follows:

```
static void Main(string[] args)
{
    var fileSource = @"Source\SourcePath\SourceSheet.xlsx";
    var fileInfo = new FileInfo(fileSource);
    var taskInfo = new TaskInfo(
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                        fileInfo.Name.Remove(fileInfo.Name.LastIndexOf(".")),
                        fileInfo.Extension.Remove(0, 1),
                        (int)fileInfo.Length,
                        "EntitySetting.xml",
                        "File Uploader Demo");

    //Create Task Info
    taskInfo = FileUploaderManager.CreateTask(taskInfo);
    Console.WriteLine("Task with id {0} was created", taskInfo.TaskID);
    Console.Write("Press any key to execute the task");
    Console.Read();

    //Exectute Task
    FileUploaderManager.ExecuteTask(taskInfo.TaskID);
    Console.WriteLine("uploaded to database already");
}
```

- Run this demonstration application, here is the executing result

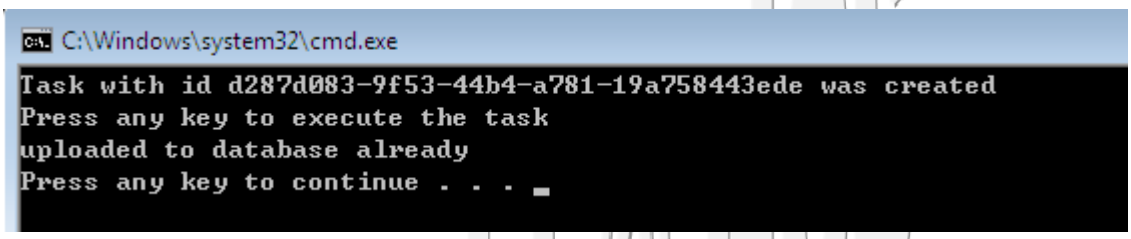
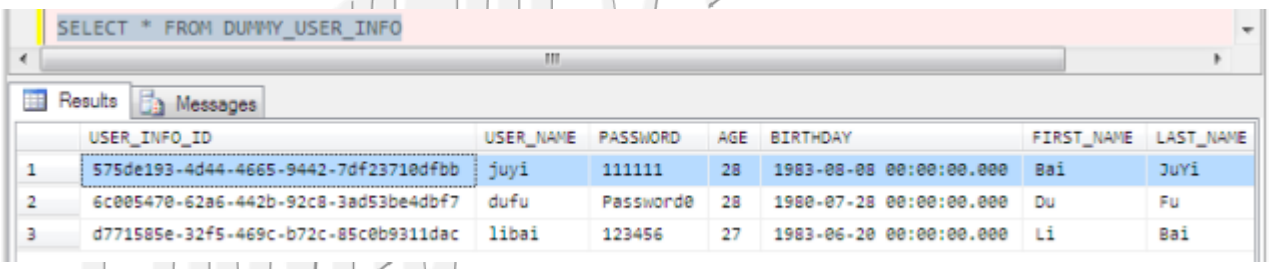


Figure 20.5. Console application execution result

We can also check the result of database table.



	USER_INFO_ID	USER_NAME	PASSWORD	AGE	BIRTHDAY	FIRST_NAME	LAST_NAME
1	575de193-4d44-4665-9442-7df23710dfbb	juyi	111111	28	1983-08-08 00:00:00.000	Bai	JuYi
2	6c005470-62a6-442b-92c8-3ad53be4dbf7	dufu	Password0	28	1980-07-28 00:00:00.000	Du	Fu
3	d771585e-32f5-469c-b72c-85c0b9311dac	libai	123456	27	1983-06-20 00:00:00.000	Li	Bai

Figure 20.6. Database table "DUMMY_USER_INFO" data

20.3. How to manage the task

We can use the `FileUploaderManager` to manage the file uploading task.

- Create a new task.

```
var taskInfo = new TaskInfo(...);
FileUploaderManager.CreateTask(taskInfo);
```

- Delete a existing task.

```
var taskId = ...;
FileUploaderManager.DeleteTask(taskId);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Update a existing task.

```
FileUploaderManager.UpdateTask(taskInfo);
```

- Get task by known task id.

```
var taskId = ...;
FileUploaderManager.GetTaskById(taskId);
```

- Get all tasks for a whole application.

```
IList<TaskInfo> taskList =
FileUploaderManager.GetAllTasks();
```

- Execute a existing task.

```
var taskId = ...;
FileUploaderManager.ExecuteTask(taskId);
```

20.4. Inside File Uploader

The process flow of the File Uploader is designed to be 3 parts which is a implementation of ETL (Extract, transform, and load).

Figure 20.7. File Uploader process flow figure

First , Extract data from different type of source files, implement the interface of `IDataReader` is enough. The predefined supported source in iConnect.NET 4.0 is **Excel**, **Txt** and **Csv**.

Second, Transform data in `FileUploader` is based on setting files, process the data from the source and save it in configured strong typed dataset.

At last, It will Load the data in dataset to database table, there is also a interface `IDataImport` can be used, File Uploader will use the default `DbHelperDataImport` to load the transformed data in dataset to database.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

20.5. Database File Uploader Uploader

iConnect.NET provide two build-in `FileUploader` Provider (inherited `FileUploaderProviderBase`) that used to archieve management of the task. One is `FileUploaderDbProvider`, another is `FileUploaderWcfProvider`.

The following code snippet describe the schema of `FileUploaderDbProvider` based configuration.

```
<configSections>
  <section name="iConnect.fileUploader"
           type="NCS.IConnect.FileUploader.Configuration.FileUploaderServiceSettings,
                 NCS.IConnect.FileUploader"/>
</configSections>
<iConnect.fileUploader defaultProvider="DbProvider">
  <fileUploaderProviders>
    <add name="DbProvider"
         type="NCS.IConnect.FileUploader.FileUploaderDbProvider,
               NCS.IConnect.FileUploader"
         applicationName="WingHingApp"
         database="iConnectSqlServer"/>
  </fileUploaderProviders>
  <sourceLocation dtSettingPath="Source\DTSettingPath\"
                  sourcePath="Source\SourcePath\"
                  backupPath="Source\BackupPath\"
                  successPath="Source\SuccessPath\"
                  errorPath="Source\ErrorPath\"/>
</iConnect.fileUploader>
```

20.6. WCF File Uploader Uploader

`FileUploaderWcfProvider` is used in distributed scenario with WCF as communication infrastructure, the following illustrates the related configuration at client side. Service side also need the essential WCF configuration and `FileUploaderDbProvider` configuration as section 3.3.

```
<iConnect.fileUploader defaultProvider="WcfProvider">
  <fileUploaderProviders>
    <add name="WcfProvider"
         type="NCS.IConnect.FileUploader.FileUploaderWcfProvider,
               NCS.IConnect.FileUploader"
         applicationName="QuickStart"
         endpoint="FileUploaderEndpoint"❶/>
  </fileUploaderProviders>
</iConnect.fileUploader>
```

❶ *`FileUploaderEndpoint`* is the endpoint name that WCF service will be consumed.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 21. Data Transfer

21.1. Introduction

Some project may need a solution to upload or download big files. Uploading and downloading task would be interrupted by user or unexpected reasons, E.G. Computer electric down, and the interrupted task need to be resumed and continued at the point of interruption.

iConnect.NET Data Transfer provide a solution to support transmission resuming at break-points. There are two modes in this solution, upload mode and download mode, the whole state related with the task at break-point will be kept in database, so the transfer can stop at any time, and be continued again by reading the state in database.

21.2. Getting Started

Assume there is a file "TestFile.rar" in "D:\Source\", and we will get the help of Data Transfer to upload this file to "D:\DataTransfer\UploadLocation".

- Step 1 : Create a blank solution and add a WinForm Project "WinClient", add the references **NCS.IConnect.Core** and **NCS.IConnect.DataTransfer**.

  Delete the default `Form1.cs` and and a new `AppForm.cs` Windows Form Item.

- Step 2 : Design the UI for the added form.

  Add the following controls and set the listed properties:

  Table 21.1.

Control Type	Property	Value
Button	(Name)	buttonExecute
	(Text)	Create && Execute Transfer Task
Button	(Name)	buttonStop
	(Text)	Stop Task
Button	(Name)	buttonContinue
	(Text)	Continue Task
ProgressBar	(Name)	progressBar
TextBox	(Name)	textBox
	(Multiline)	True
	(ReadOnly)	True

  Adjust the alignment for the controls, following is the result.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

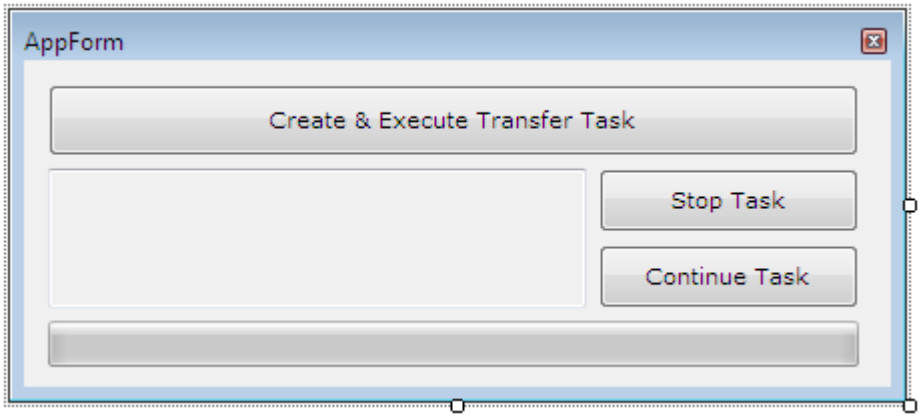


Figure 21.1. Final user interface

- Step 3 : Add the following schema in app.config.

```
<configSections>
    <section name="dataConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Data.Configuration.DatabaseSettings,
                   Microsoft.Practices.EnterpriseLibrary.Data"/>
    <section name="iConnect.applicationContexts"
            type="NCS.IConnect.Core.ApplicationContexts.Configuration.ApplicationContextSettings,
                   NCS.IConnect.Core"/>
    <section name="iConnect.dataTransfer"
             type="NCS.IConnect.DataTransfer.Configuration.DataTransferServerSettings,
                   NCS.IConnect.DataTransfer"/>
  </configSections>

  <!-- ConnectionStrings -->
  <connectionStrings>
    <add name="iConnectSqlServer"
         connectionString="..."
         providerName="System.Data.SqlClient" />
  </connectionStrings>
  <dataConfiguration defaultDatabase="iConnectSqlServer"/>

  <!-- iConnect.NET: ApplicationContext -->
  <iConnect.applicationContexts defaultContextLocator="callContextLocator">
    <contextLocators>
      <add name="callContextLocator"
           type="NCS.IConnect.Core.ApplicationContexts.CallContextLocator,
                 NCS.IConnect.Core"/>
    </contextLocators>
  </iConnect.applicationContexts>

  <!-- iConnect.NET: DataTransfer -->
  <iConnect.dataTransfer defaultProvider="DbProvider">
    <dataTransferProviders>
      <add name="DbProvider"
           type="NCS.IConnect.DataTransfer.DataTransferDbProvider, NCS.IConnect.DataTransfer"
           applicationName="TestApp"
           database="iConnectSqlServer"/>
    </dataTransferProviders>
    <transferLocation basePath=""
                      uploadPath="D:\DataTransfer\UploadLocation\"
                      downloadPath="D:\DataTransfer\DownloadLocation\"/>
  </iConnect.dataTransfer>
```

- Step 4 : Add the C# code in `AppForm.cs`.

  Add the following namespace reference:

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
using NCS.IConnect.DataTransfer;
```

Add 3 properties:

```
public TransInfo TransInfo { get; set; }
public TransferComponent TransferComponent { get; set; }
public String TransId { get; set; }
```

Add the following at the end position of `AppForm` method:

```
buttonContinue.Enabled = false;
buttonStop.Enabled = false;
```

Add the following code in `buttonExecute_Click` method:

```
buttonExecute.Enabled = false;
buttonStop.Enabled = true;
String file = @"D:\Source\TestFile.rar";
FileInfo fileInfo = new FileInfo(file);
TransInfo = new TransInfo();
TransId = Guid.NewGuid().ToString();

TransInfo.TransId = TransId;
TransInfo.FileName = fileInfo.Name.Remove(fileInfo.Name.LastIndexOf("."));
TransInfo.FileType = fileInfo.Extension.Remove(0, 1);
TransInfo.NumberOfBlocks = TransInfo.GenerateNumberOfBlocks(fileInfo.Length);
TransInfo.TransDirection = TransDirection.Upload;
TransInfo.Initialize();

//Create Transfer Task
TransInfo = DataTransferManager.CreateTransfer(TransInfo);

//Create Transfer Component
TransferComponent = DataTransferManager.BuildTransferComponent(file, TransInfo);
TransferComponent.OnError += new TransferErrorHandler(TransferComponent_OnError);
TransferComponent.OnNotify += new TransferNotifyHandler(TransferComponent_OnNotify);

//Transfer Execution
DataTransferManager.DoTransfer(TransferComponent);
```

The code in the method `TransferComponent_OnError` is:

```
MessageBox.Show(transError.Error.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

The code in the method `TransferComponent_OnNotify` is:

```
progressBar.Value = (int)(100 * (transInfo.NumberOfBlocks - transInfo.BlocksLeft) /
transInfo.NumberOfBlocks);
textBox.Clear();
textBox.Text += String.Format("File : {0}.{1}\r\n", transInfo.FileName, transInfo.FileType);
textBox.Text += String.Format("Total Blocks : {0}\r\n", transInfo.NumberOfBlocks);
textBox.Text += String.Format("Blocks Left : {0}\r\n", transInfo.BlocksLeft);
textBox.Text += String.Format("Transfer Status : {0}", transInfo.TransStatus.ToString());

if (progressBar.Value == 100)
{
    buttonContinue.Enabled = false;
    buttonStop.Enabled = false;
}
```

Code for `buttonStop_Click` is:

```
buttonStop.Enabled = false;
buttonContinue.Enabled = true;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
TransferComponent.Stop();
```

Code for `buttonContinue_Click` is:

```
buttonContinue.Enabled = false;
buttonStop.Enabled = true;

DataTransferManager.DoTransfer(TransferComponent);
```

- Step 5 : Build and run the program, we can see the running status as follows:

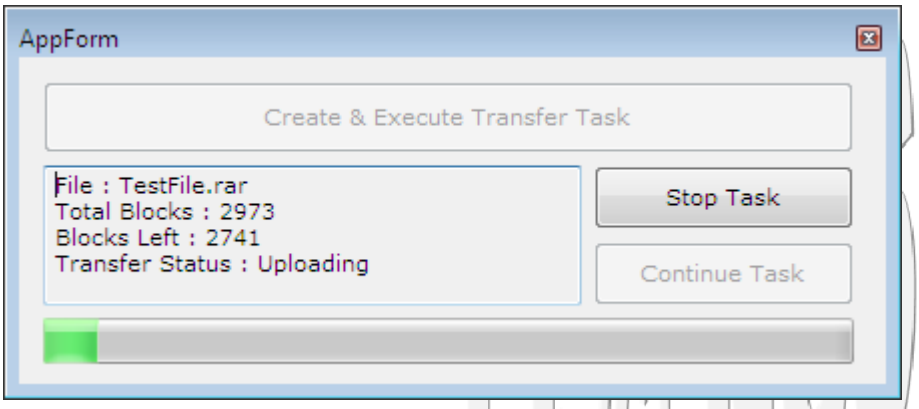


Figure 21.2. Click "Create & Execute Transfer Task" button

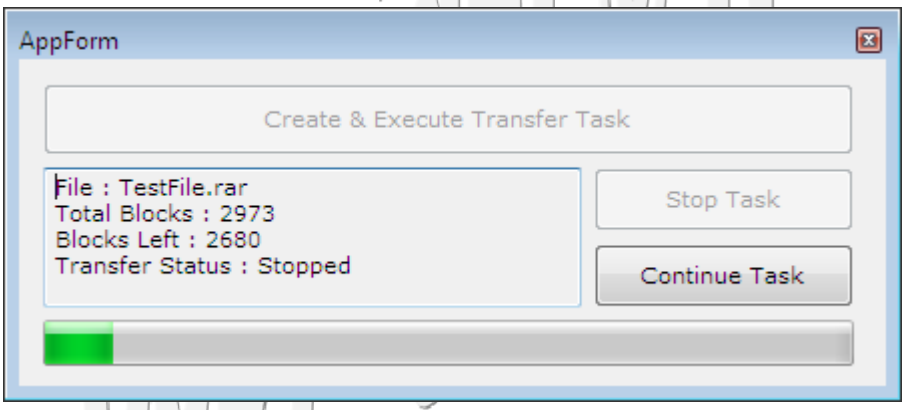


Figure 21.3. Stop the transfer task by clicking "Stop Task"

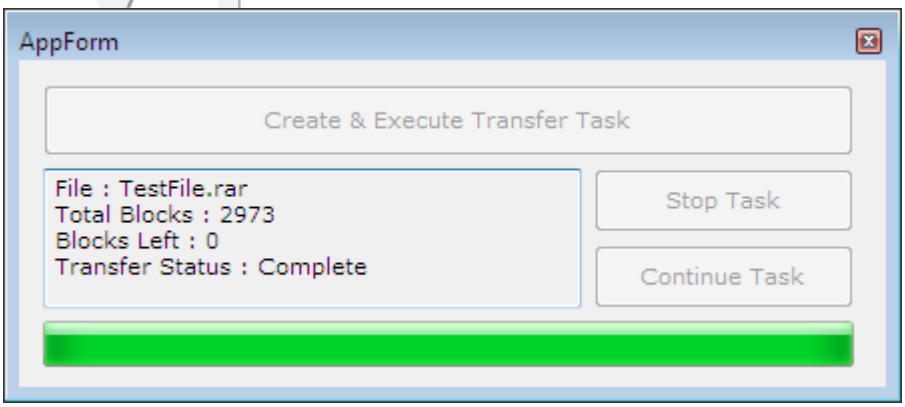


Figure 21.4. Click "Continue Task" and waiting the transfer finished

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

We can also check that the TestFile.rar is already in "D:\DataTransfer\UploadLocation".

21.3. Programming

21.3.1. DataTransferManager

`DataTransferManager` is the facade for manage data transfer task.

- Create a new data transfer task.

```
TransInfo transInfo = new TransInfo();
transInfo.TransId = Guid.NewGuid().ToString(); ;
transInfo.FileName = ...;
transInfo.FileType = ...;
transInfo.NumberOfBlocks = transInfo.GenerateNumberOfBlocks(fileInfo.Length);
transInfo.TransDirection = TransDirection.Upload;
transInfo.Initialize();
transInfo = DataTransferManager.CreateTransfer(transInfo);
```

- Delete a existing data transfer task.

```
DataTransferManager.DeleteTransfer(transId, versionNo);
```

- Get a data transfer task.

```
TransInfo transInfo = DataTransferManager.GetById(transId);
```

- Get all data transfer tasks.

```
IList<TransInfo> transInfoList = DataTransferManager.GetAll();
```

- Execute a data transfer task.

```
TransferComponent transferComponent = DataTransferManager.BuildTransferComponent(fileName,
transInfo);
DataTransferManager.DoTransfer(transferComponent);
```

- Stop a data transfer task.

```
DataTransferManager.StopTransfer(transId);
```

TransferComponent also provide a way to stop a transfer task.

```
transferComponent.Stop();
```

21.3.2. TransferComponent

TransferComponent is a abstract class **DownloadTask** and **UploadTask**, TransferComponent provide two Event that can be subscribed also.

OnError : this event is raised when there is a error in the transfer process.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

OnNotify : this event is raised at the time that every block data was transfered .

21.3.3. DataTransferDbProvider

iConnect.NET provide two build-in Data Transfer Providers that used to archieve the transfer of data blocks. One is DataTransferDbProvider, another is DataTransferWcfProvider.

The following code snippet describe the schema of DataTransferDbProvider based configuration.

```
<configSections>
  <section name="iConnect.dataTransfer"
           type="NCS.IConnect.DataTransfer.Configuration.DataTransferServerSettings,
                 NCS.IConnect.DataTransfer"/>
</configSections>
<iConnect.dataTransfer defaultProvider="DbProvider">
  <dataTransferProviders>
    <add name="DbProvider"
         type="NCS.IConnect.DataTransfer.DataTransferDbProvider,
               NCS.IConnect.DataTransfer"
         applicationName="WingHingApp"
         database="iConnectSqlServer"/>
  </dataTransferProviders>
  <transferLocation basePath=""❶
                    uploadPath="D:\DataTransfer\UploadLocation\"❷
                    downloadPath="D:\DataTransfer\DownloadLocation\"❸/>
</iConnect.dataTransfer>
```

❶ basePath : is used as base path that will combined with uploadPath or downloadPath when they are not root path.

❷ uploadPath : upload path, can be root path.

❸ downloadPath : download path, can be root path.

21.3.4. DataTransferWcfProvider

DataTransferWcfProvider is used in distributed scenario with WCF as communication infrastructure, the following illustrates the related configuration at client side. Service side also need the essential WCF configuration and DataTransferDbProvider configuration as section 3.3.

```
<iConnect.dataTransfer defaultProvider="WcfProvider">
  <dataTransferProviders>
    <add name="WcfProvider"
         type="NCS.IConnect.DataTransfer.DataTransferWcfProvider,
               NCS.IConnect.DataTransfer"
         applicationName="TestApp"
         endpoint="DataTransfer"❶/>
  </dataTransferProviders>
</iConnect.dataTransfer>
```

❶ DataTransfer is the endpoint name that WCF service will be consumed.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

21.4. Design Overview

The following figure illustrates the main elements in Data Transfer module and the relationship between each other.

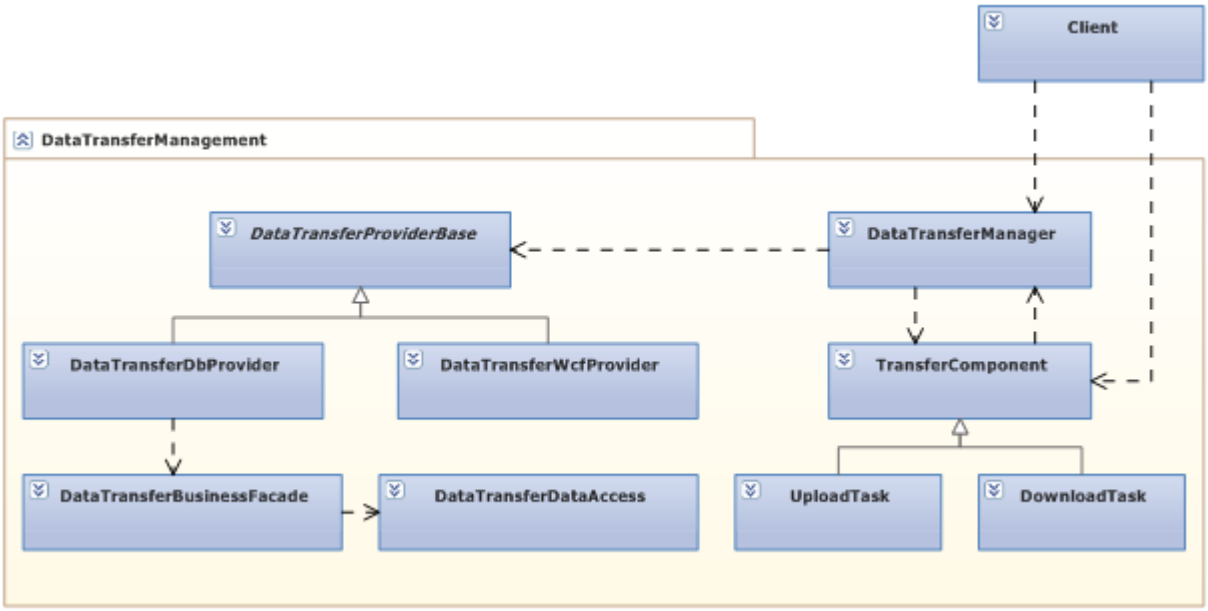


Figure 21.5. Design overview of Data Transfer

DataTransferManager

This is the facade class that provide the method to manage the transfer task, E.G. Create a transfer task, delete a transfer task, execute a transfer task, stop a transfer task, get all transfer tasks.

DataTransferProviderBase

It is a base class that is used to provide the core function used in this module, it has two implementations, one is **DataTransferDbProvider** , another is **DataTransferWcfProvider**.

TransferComponent

Currently, there are two transfer mode, Upload and Download, there are two corresponding transfer components, **UploadTask** and **DownloadTask**.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 22. Controls

22.1. Introduction

If you are an ASP.NET web application developer, you must be familiar with validator control. From the first version, ASP.NET provides us with a series of special web server controls, which are used to validate the input value entered into the control. These kinds of web server controls are called Validator Control. Validator controls make it easy to perform input control based validation. To apply such an easy-to-use validation mode to Windows forms based application, iConnect.NET creates a set of similar validator controls. All of these validator controls, which are defined in the `NCS.IConnect.Controls.WinForm` assembly, are listed below:

- `RequiredFieldValidator`: This validator control is used to ensure a value must be inputted into the control to validate;
- `RegularExpressionValidator`: This validator control is used to ensure what is inputted into the control to validate must match the given regular expression validation pattern;
- `RangeValidator`: This validator control is used to ensure the value inputted into the control to validate must fall into the specified range;
- `CompareValidator`: This validator control is used to ensure the value inputted into the control to validate must match the specified data type or comparison rule;
- `SGInfoValidator`: This validator control is used to ensure the value inputted into the control to validate must be a valid NRIC, FIN, etc.
- `CustomValidator`: If the above built-in validator controls cannot meet your validation requirements, you can use this one to define your own validation rule;
- `FormValidator`: One Windows form can have one `FormValidator` control. Once invoked, which will automatically invoke all of the validator controls in the form to perform form-wide validation.

Note

EntLib Validation Application Block provides us with a new validation solution. The validation policy is directly applied to the entity class by decorating related validator attributes or configuring. Validation Application Block implements the integration with Windows Forms application and ASP.NET web application to achieve UI based input validation. In addition, WCF integration is also implemented for automatic input parameter validation. This is good choice which can be considered to be used in your application. For more information about EnterLib

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Validation Application Block, please refer to the EntLib developer's guide, which can be downloaded from EntLib official web site.

22.2. Getting Started

This section will give you a sample to guide you to use the above mentioned validator controls to validate the value inputted into the control.

Step I: Load Validator Controls into Visual Studio's Toolbox

Create a Windows Form project in the Visual Studio. Keep the form designer is opened, and right click the blank area of the ToolBox dialog box. Choose the "Add tab" item from the contextual menu to add a new tab. You can give the newly added tab a readable name, E.G.iConnect.NET Validator Controls. Then right click the blank tab, and select the "Choose items" contextual menu item. The following "Choose Toolbox Items" dialog box will be popped up. Now you should click the Browse button to load the NCS.IConnect.Controls.Windows.dll from the /bin folder of iConnect.NET installation directory. Once the given assembly is successfully loaded, iConnect.NET validator controls will appear in the ToolBox.

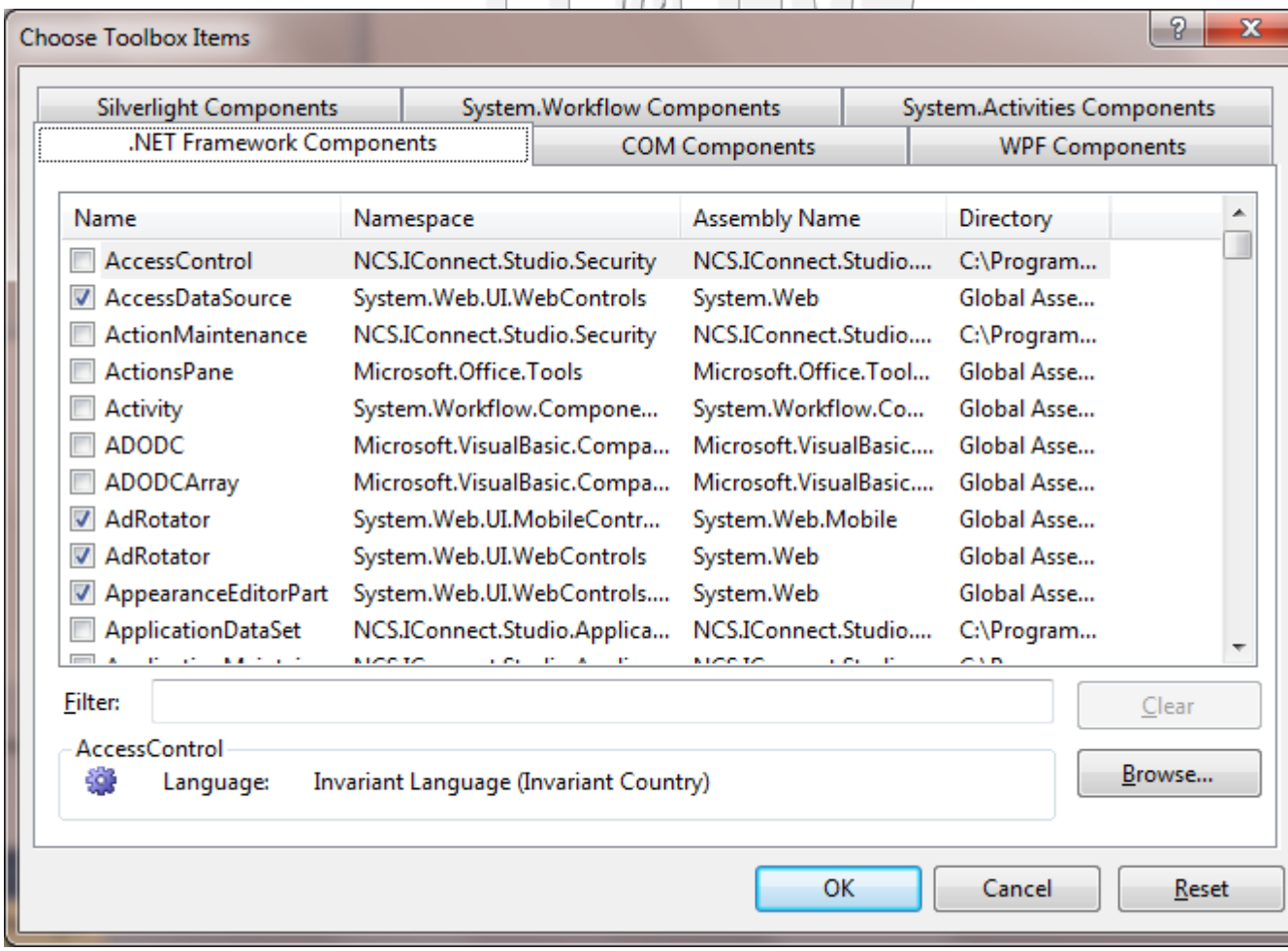


Figure 22.1. Choose Toolbox Items Dialog Box

Step II: Design the Form

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

We will add three simple fields to validate. They are *First Name*, *Last Name* and *Age*. They are all mandatory fields. The age of integer type must be between 18 and 50. Then design the Form1 to be like the following figure.

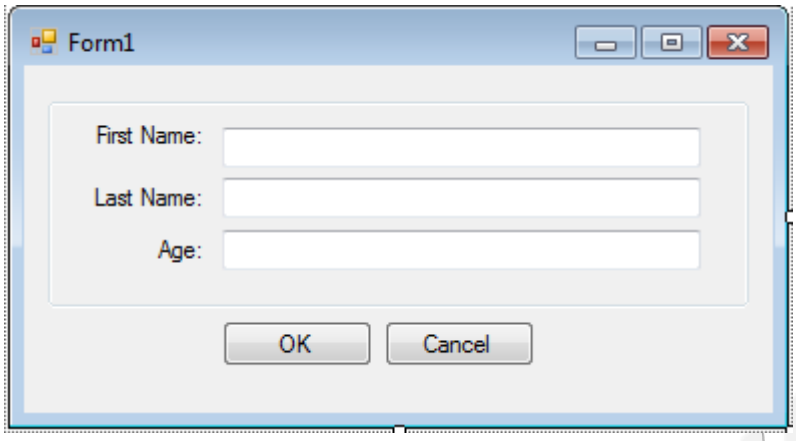


Figure 22.2. Validation Form

Since three fields are all mandatory ones, we drag and drop three `RequiredFieldValidator` from "iConnect.NET Validator Controls" toolbox to validate them. After giving them a meaningful name, E.G. *rfvFirstName*, *efvLastName* and *rfvAge*, you need to set the `ControlToValidate` property as name of the corresponding `Textbox` to validate. Finally set the `ErrorMessage` properties of the three as "First Name/Last Name/Age is mandatory!".

Then drag and drop a `RangeValidator` validate the Age's range (18~50). Besides to set the `ControlToValidate` and `ErrorMessage`, E.G. "The Age must between 18 and 50!", you need to set the following properties as well.

- Type: Integer
- MinimumValue: 18
- MaximumValue: 50

The last validator control needed is `FormValidator`, and you need not to set any property. Now, you can write your validation code in the OK button's click event handler. You just need to invoke the Validate method of the added `FormValidator`.

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    private void btnOk_Click(object sender, EventArgs e)
    {
        this.formValidator.Validate();
    }
}
```

Run your application, you will see a read icon shown near of the corresponding TextBox if pressing the OK button with invalid inputs. If you hover the mouse on the read icon, you will

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

see your specified error message in the form of tip. The following is screen shot for validation error.

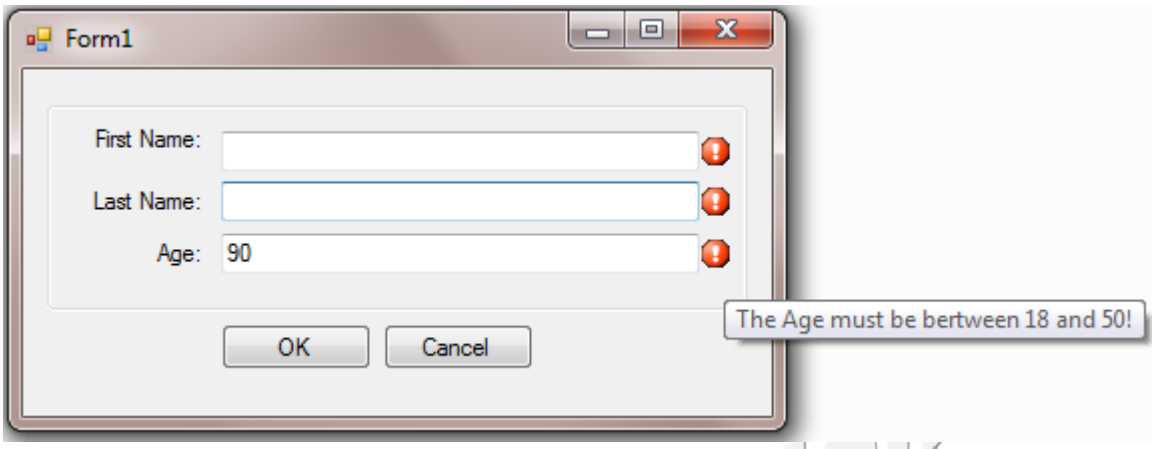


Figure 22.3. Validation Error

22.3. RequiredFieldValidator

If the input control corresponds to a mandatory data field, you need to use `RequiredFieldValidator` to ensure the end user input a value. Except for such two common properties: `ControlToValidate` and `ErrorMessage`, there is an important property: `InitialValue`. If the field is initialized as a given value, and you need the end user to input another value which is must different from this value, you can the set InitialValue property. The default value for this property is null.

22.4. RegularExpressionValidator

The `RegularExpressionValidator` control is used to determine whether the value of an input control matches a pattern defined by a regular expression. This type of validation allows you to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers, postal codes, and so on. After figuring out the regular expression as the validation rule for a given data field, you just need to use it to evaluate `RegularExpressionValidator's` `ValidationExpression`.

22.5. RangeValidator

Like the field of Age in the above demonstration, if you must ensure its value must fall into a given range, you need the `RangeValidator` to validate the input control specific to such a filed. You must specify the field's data type by setting the `Type` property. The combination between `MinimumValue` and `MaximumValue` defines the valid data range. The data type of `RangeValidator`'s Type property is `NCS.IConnect.Controls.WinForm.ValidationDataType`. `ValidationDataType` is an enumeration. Its definition is represented in the following code snippet.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public enum ValidationDataType
{
    Currency,
    Date,
    Double,
    Integer,
    String
}
```

22.6. CompareValidator

There are commonly used validation rules:

- What is entered into a given input control must match the specified data type;
- The value inputted into a given control must meet the pre-defined comparison rule.

For the first validation scenario, you just need to specify the `Type` property, whose data type is the above mentioned `ValidationDataType`. If you need to perform comparison rule based validation, you first need to choose an operator via its Operator property. The data type of Operator property is the following `NCS.IConnect.Controls.WinForm.ValidationCompareOperator` enumeration.

```
public enum ValidationCompareOperator
{
    DataTypeCheck,
    Equal,
    GreaterThan,
    GreaterThanEqual,
    LessThan,
    LessThanEqual,
    NotEqual
}
```

What is used to compare with the value of input control to validate can be the value of another control, or a given hard coded value. For the first comparison case, you need to set the `ControlToCompare` property as the name of a given control in the same form. In the fixed value comparison scenario, you just set `ValueToCompare` property.

22.7. CustomValidator

If you need special validation rules. E.G. The validation rule involves complicated business logic, the `CustomValidator` enable you to directly program your validation rule. When programming `CustomValidator`, you generally need to code your validation rule by registering its `Validating` event. The following code snippet demonstrates you how to program `CustomValidator`'s Validating event handler. As the code snippet represents, this `Validating` event handler accepts an event argument of `ValidatingCancelEventArgs` type. `ValidatingCancelEventArgs`'s Valid property indicates whether the input is valid or not.

```
void cstmAge_Validating(object sender, ValidatingCancelEventArgs e)
{
    try
    {
        // Must be 18 or older
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        DateTime dob = DateTime.Parse(((Control)e.ControlToValidate).Text);
        DateTime legal = DateTime.Now.AddYears(-18);
        e.Valid = ( dob < legal );
    }
    catch( Exception ex )
    {
        e.Valid = false;
    }
}
```

22.8. SGInfoValidator

The `SGInfoValidator` is used to validate Singapore common informations, like NRIC,FIN etc. It has the following three properties:

- *NationalityControl*: Can be (none) or the name of the control that contains the nationality information;
- *NationalityValue*: If the `NationalityControl` is not (none), and when the value of the specified `NationalityControl` is equal to the specified `NationalityValue`, the validation on the `ControlToValidate` will work. When the NationalityControl is (none), the validation on the `ControlToValidate` will always work. It is strongly recommended to set the property value at run time, but at design time;
- *ValidationType*: The type of the validation, for example: NRIC validation, Singaporean telephone number validation. Notes that different validation type has different validation rules.

The data type of `ValidationType` property is the following a `NCS.IConnect.Controls.WinForm.RegularExpressionType` enumeration.

```
public enum RegularExpressionType
{
    NRIC,
    FIN,
    PostalCode,
    PagerNumber,
    TelephoneNumber,
    Fax,
    MobilePhone
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 23. Calendar

23.1. Introduction

Calendar is mainly used to calculate date and relevant functions. As basic functions for projects, Calendar is to provide `WorkdayCalendar` function for iConnect.NET. Of course, Calendar is designed to be flexible and extensible. You can create your own components to extend Calendar which is more compatible to your projects.

23.2. Getting Started

This section gives you a demonstrative sample application to guide you program Calendar, including how to create or load `WorkdayCalendar`, and how to perform workday calculation based on the loaded `WorkdayCalendar`.

Step I: Create Application Record

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You first need to execute the following SQL script against the this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named " *`CalendarDemo`* ".

```
INSERT INTO [dbo].[T_IC_APP]
           ([APP_ID]
           ,[APP_NAME]
           ,[LOWERED_APP_NAME]
           ,[APP_DESC]
           ,[FLAG]
           ,[VERSION_NO]
           ,[TRANSACTION_ID]
           ,[CREATED_BY]
           ,[CREATED_TIME]
           ,[LAST_UPDATED_BY]
           ,[LAST_UPDATED_TIME])
     VALUES
           ('3B9413F6-3644-4733-8CD6-DA08B203D751'
           ,'CalendarDemo'
           ,'calendardemo'
           ,''
           ,''
           ,1
           ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
           ,'Sys'
           ,GETDATE()
           ,'Sys'
           ,GETDATE())
GO
```

Step II: Create a Windows Form Application and Design the Form

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Using your Visual Studio, you first create a new Windows Form project and add the reference to the `NCS.IConnect.Calendar` assembly, which is located in the installation directory. In the automatically created Windows Form (Form1), drag and drop necessary Controls into the form and design it according to the following screen shot.

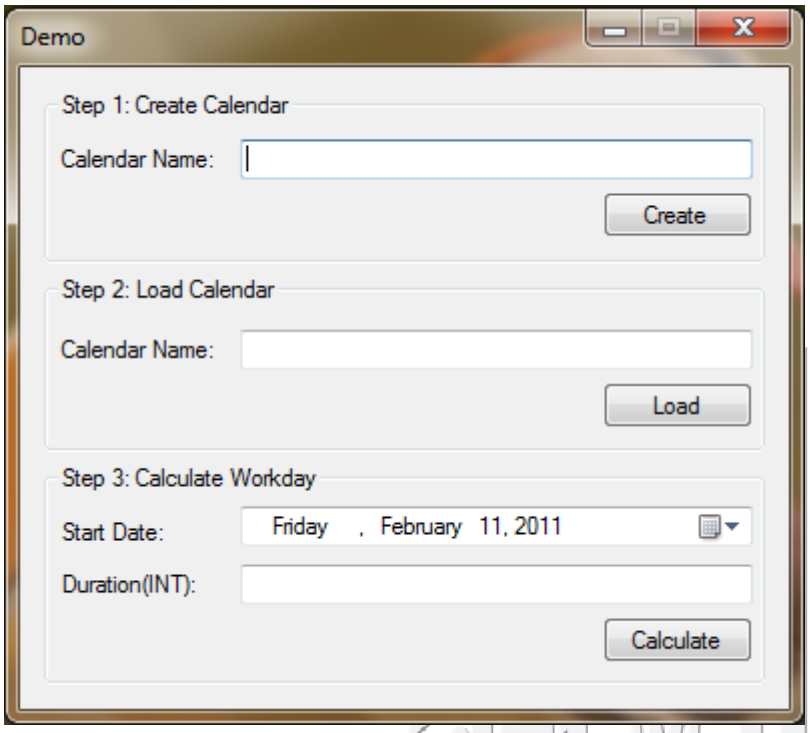


Figure 23.1. Calendar Demonstration Form

Step III Configuring Calendar

When the above demonstration form is designed, add an app.config file into the project. You then directly copy the configuration illustrated in the following XML snippet into this newly added configuration file. Don't forget to revise the connection string to connect to your specified database.

```
<configuration>
  <configSections>
    <section name="iConnect.calendar"
             type="NCS.IConnect.Calendar.Configuration.CalendarSettings,
                   NCS.IConnect.Calendar"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         providerName="System.Data.SqlClient"
         connectionString="..."/>
  </connectionStrings>
  <iConnect.calendar defaultProvider="dbProvider">
    <storageProviders>
      <add name="dbProvider"
           type="NCS.IConnect.Calendar.DbStorageProvider,
                 NCS.IConnect.Calendar"
           applicationName="CalendarDemo"
           database="iConnectDb"/>
    </storageProviders>
  </iConnect.calendar>
</configuration>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step IV: Programing Calendar

Double click the windows form and show the code. As illustrated in the following code snippet, we define a private _calendar field of the `WorkdayCalendar` type.

```
public partial class WorkdayCalculateForm
{
    private WorkdayCalendar _calendar = null;

    //Others
}
```

Double click `Create` button and copy the following code as the *Create* button's click event handler. As illustrated in the following code snippet, the static method Create of `WorkdayCalendar` is called to create workday calendar, whose name is specified in the Calendar name `TextBox`.

```
private void createButton_Click(object sender, EventArgs e)
{
    try
    {
        WorkdayCalendar calendar = WorkdayCalendar.Create(this.newCalendarNameTextBox.Text, new
List<CalendarDate>());
        calendar.Insert();

        MessageBox.Show(string.Format("Calendar \"{0}\" is created successfully.",
this.newCalendarNameTextBox.Text),
                        "Information",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Operation error: {0}.", ex.Message),
                        "Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
}
```

The following code snippet represents the Load button's click event handler, in which a `WorkdayCalendar` is directly instantiated by specifying the calendar name.

```
private void loadButton_Click(object sender, EventArgs e)
{
    try
    {
        _calendar = new WorkdayCalendar(this.calendarNameTextBox.Text);

        MessageBox.Show(string.Format("Calendar \"{0}\" is loaded successfully.",
this.calendarNameTextBox.Text),
                        "Information",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Operation error: {0}.", ex.Message),
                        "Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

When both Start Date and Duration are specified, you can click the Calculate button to list all of workday in the specified duration. The following code snippet represents the Calculate button click event handler.

```
private void calculateButton_Click(object sender, EventArgs e)
{
    try
    {
        if (_calendar != null)
        {
            IList<DateTime> datelist = _calendar.GetWorkdays(this.startDateTimePicker.Value,
                this.startDateTimePicker.Value.AddDays(Int32.Parse(this.durationTextBox.Text)));
            StringBuilder output = new StringBuilder();
            output.AppendLine("The workday list is:");
            foreach (DateTime date in datelist)
            {
                output.AppendLine(date.ToString("yyyy-MM-dd"));
            }
            MessageBox.Show(output.ToString(), "Information", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show(string.Format("Calendar \"{0}\" is not loaded.",
this.newCalendarNameTextBox.Text),
                            "Error",
                            MessageBoxButtons.OK,
                            MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(string.Format("Operation error: {0}.", ex.Message),
                        "Error",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
}
```

23.3. StorageProvider

Similar to other iConnect.NET component, provider based design is applied into Calendar. In order to support multiple calendar data storage, abstract `StorageProvider` is created. There are two built-in `StorageProvider`: `DbStorageProvider` and `WcfStorageProvider`. As their names suggests, `DbStorageProvider` is specific to database storage, and `WcfStorageProvider` implements calendar data retrieval by remotely invoking calendar service. Which kind of `StorageProvider` is by default used is controlled by configuration. The following is a demonstrative configuration snippet, in which the `DbStorageProvider` is configured as used as the default `StorageProvider`.

Besides the common provider configuration properties, name and type, you just need to specify the *`applicationName`* and *`database`* (the name of connection string).

```
<configuration>
  <configSections>
    <section name="iConnect.calendar"
             type="NCS.IConnect.Calendar.Configuration.CalendarSettings,
                   NCS.IConnect.Calendar" />
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        connectionString="..."
        providerName="..."/>
  </connectionStrings>
  <iConnect.calendar defaultProvider="dbStorageProvider" >
    <storageProviders>
      <add name="dbStorageProvider"
           type="NCS.IConnect.Calendar.DbStorageProvider,
                 NCS.IConnect.Calendar"
           applicationName="demoApp"
           database="iConnectDb" />
    </storageProviders>
  </iConnect.calendar>
  .....
</configuration>
```

`WcfStorageProvider` is usually applied to distributed scenario, it provides opportunity for you to build multi-layer application. As with `WcfStorageProvider` related configuration, the name of the endpoint to invocate calendar service must be specified. The more than one `WcfStorageProvider`, which come from different applications, to invocate the same service, the `applicationName` should be configured accordingly. Otherwise, the *`applicationName`* property can be ignored.

```
<configuration>
    <configSections>
      <section name="iConnect.calendar"
               type="NCS.IConnect.Calendar.Configuration.CalendarSettings,
                     NCS.IConnect.Calendar" />
    </configSections>
    <iConnect.calendar defaultProvider="WcfStorage" >
      <storageProviders>
        <add name="WcfStorage "
             type="NCS.IConnect.Calendar.WcfStorageProvider,
                   NCS.IConnect.Calendar "
             applicationName="QuickStart"
             endpoint="calendarService" />
      </storageProviders>
    </iConnect.calendar>
    <system.serviceModel>
      <client>
         <endpoint name="calendarService"
                   address="http://localhost/ICQuickStartService/Calendar.svc"
                   binding="wsHttpBinding"
                   contract="NCS.IConnect.Calendar.IWcfStorageProvider"/>
      </client>
    </system.serviceModel>
    .....
  </configuration>
```

23.4. WorkdayCalendar

Calendar have only `WorkdayCalendar` as build-in component of iConnect.NET, it is mainly used to calculate workday, also supports day level T+n algorithm. Below the table lists the functional APIs related to `WorkdayCalendar`.

Method Name	Description
Constructor	The constructor of `WorkdayCalendar` is used to load an existing workday calendar.
Create	This is a static method which is for creating a new workday calendar.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Method Name	Description
NextWorkday	The property is to calculate next workday based today.
PrevWorkday	The property is to calculate previous workday based today.
IsWorkday	Check whether the specified date is workday.
GetWorkday	Get a workday based specific date or day offset.
GetWorkdays	Get workday list through series of conditions.
GetWeekWorkdays	Get workday list based current week.
GetMonthWorkdays	Get workday list based current month.
GetYearWorkdays	Get workday list based current year.
GetWorkdayCount	Calculate workday count through series of conditions.
GetWeekWorkdayCount	Calculate workday count based current week.
GetMonthWorkdayCount	Calculate workday count based current month.
GetYearWorkdayCount	Calculate workday count based current year.
Calendar	It's a property, also a business object to handle calendar meta data.
Insert	Insert a new workday calendar (record).
Update	Update an existing workday calendar (record).
Delete	Delete an existing workday calendar (record).

The following code lines illustrate how to use these APIs in common cases:

```
WorkdayCalendar calendar = WorkdayCalendar.Create("CalendarName", new List<CalendarDate>());
// Create a new WorkdayCalendar instance which is based default calendar provider, meanwhile needs
to specify calendar name
// and particular day set (The particular day is, if today is a workday, you can set today to holiday).

WorkdayCalendar calendar = WorkdayCalendar.Create("WcfStorage", "CalendarName", new
List<CalendarDate>());
// Create a new WorkdayCalendar instance which is based specified calendar provider name configured
in app.config
// or web.config.

WorkdayCalendar calendar = new WorkdayCalendar("MyWorkdayCalendar");
// Load an existing WorkdayCalendar by specific calendar name.

WorkdayCalendar calendar = new WorkdayCalendar("DatabaseStorage", "MyWorkdayCalendar");
// Load an existing WorkdayCalendar by specified calendar provider name and calendar name.

DateTime nextWorkday = calendar.NextWorkday;
// Calculate next workday based today through WorkdayCalendar instance.
DateTime nextWorkday = calendar.PrevWorkday;
// Calculate previous workday based today through WorkdayCalendar instance.

DateTime? specificWorkday = calendar.GetWorkday(7);
// Calculate the specific workday based the day after seven days.
... ...

calendar.Calendar.ApplyParticularDay(new CalendarDate(DateTime.Parse("2011-06-01"),
WorkdayCalendarTag.Holiday));
calendar.Insert();
// Commit a new WorkdayCalendar into specified calendar storage.

calendar.Calendar.ApplyParticularDay(new CalendarDate(DateTime.Parse("2011-06-02"),
WorkdayCalendarTag.Holiday));
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
calendar.Update();
// Update an existing WorkdayCalendar into specified calendar storage.
... ...
```

23.5. Extensibility

Although Calendar have only one build-in `WorkdayCalendar`, but actually it provides many points to extend or customize your own Calendar component. Calendar component is basically composed of `CalendarBuilder` and `CalendarTag`.

- `CalendarTag`: `CalendarTag` actually is related to `CalendarDate` object. Calendar treats everyday as an object which is easy to be used by developers. So `CalendarDate` needs to represents what day is the specified day, `CalendarTag` just play such a role for `CalendarDate`. e.g. `WorkdayCalendarTag` consists of "Workday" and "Holiday", even "Unspecified".
- `CalendarBuilder`: `CalendarBuilder` is used to build `CalendarDate` list by specific start date, end date and particular day set. It also provides the function to check whether the specified day is particular day.

`CalendarTag` and `CalendarBuilder` are core of one Calendar component. Except for both of them, it's better to create a facade class to encapsulate `CalendarTag` and `CalendarBuilder`, just as `WorkdayCalendar` class. The facade class is friendly and distinct for developers to program Calendar functions. Generally speaking, we are about to implement custom Calendar component through the following steps:

23.5.1. Step I: Create Custom CalendarTag

At first, you needs to define custom `CalendarTag` to identify the specific day is "Workday", "Holiday" or others. The custom `CalendarTag` has to derive from typed class "`CalendarTag`". Just like the demo below:

```
using System;
using System.Collections.Generic;
using NCS.IConnect.Calendar;

namespace CustomCalendar
{
    public class CustomCalendarTag : CalendarTag
    {
        public override IEnumerable<string> CalendarTags ❶
        {
            get { throw new NotImplementedException(); }
        }

        public override string DefaultTag ❷
        {
            get { throw new NotImplementedException(); }
        }
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❶ `CalendarTags` is used to list all of tags to represent `CalendarDate` is "Holiday", "Workday" or others. It is the basic element of Calendar component, and is used by `CalendarBuilder` and other objects.

❷ As internal mechanism of Calendar component, `CustomCalendarTag` must specify a default tag, it's noted that `DefaultTag` must be in the list of `CalendarTags`.

23.5.2. Step II: Create Custom CalendarBuilder

`CalendarBuilder` is the core portion of Calendar component to fulfil date calculation. In fact, it also decides which type of Calendar. To implement custom `CalendarBuilder`, the class must inherit interface "`ICalendarBuilder`" to expose necessary behaviors. The following code lines show "How To":

```
using System;
using System.Collections.Generic;
using NCS.IConnect.Calendar;

namespace CustomCalendar
{
    public class CustomCalendarBuilder : ICalendarBuilder
    {
        public IEnumerable<CalendarDate> Build(DateTime startDate,
                                               DateTime endDate,
                                               IEnumerable<CalendarDate> particularDates) ❶
        {
            throw new NotImplementedException();
        }

        public CalendarTag CalendarTag ❷
        {
            get { throw new NotImplementedException(); }
        }

        public string CalendarType ❸
        {
            get { throw new NotImplementedException(); }
        }

        public bool IsGeneralDay(CalendarDate calendarDate) ❹
        {
            throw new NotImplementedException();
        }
    }
}
```

❶ "Build" method is used to build date list by start date, end date and particular day set. Here you have to understand what's particular day set, it's collection of particular day. As for particular day, it is mentioned above.

❷ `CalendarBuilder` needs to specify `CalendarTag`, property "`CalendarTag`" is just the role.

❸ Actually property "`CalendarType`" is just a tag but real typed class, and it exactly represents which type of Calendar through it.

❹ Calendar component will utilize method "`IsGeneralDay`" to do check whether the specific day is particular day.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

23.5.3. Step III: Create Custom Calendar

Apart from above 2 portions, it's necessary to make developers program with distinct guideline. You need a custom calendar facade class. The class inherits class "`CalendarBase<CustomCalendarBuilder>`" which gives basic features. See below as a sample facade class:

```
using System;
using NCS.IConnect.Calendar;

namespace CustomCalendar
{
    public class CustomCalendar : CalendarBase<CustomCalendarBuilder>
    {
        protected override CustomCalendarBuilder CalendarBuilder ❶
        {
            get { throw new NotImplementedException(); }
        }
    }
}
```

❶ It's an internal object, and you can utilize such a property to construct your calendar calculation logic in facade class "`CustomCalendar`". By the way, as base class "`CalendarBase<CustomCalendarBuilder`", it expose "`CalendarContent`" object and other useful properties/methods to help you construct calendar calculation logic.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 24. Caching

24.1. Introduction

Caching is a preferred means to improve the performance and robustness of applications. However, it is difficult to synchronize the cached data across servers in load balance scenario. In order to address cache synchronization, the following cache item expiration policy can be approached:

- Dependency based Expiration: Create the same dependency for all machine specific cache, and approach notification mechanism to make the cache item expire once the dependency changes.
- Absolute Time based Expiration: Make all of the machine specific cache expire at an absolute time specified.

We will approach the above solutions to resolve cache synchronization issue in load balance scenario.

24.2. Getting Started

iConnect.NET directly utilizes Microsoft Enterprise Library Caching Application Block (CAB) as the underlying caching framework. CAB controls the cache item expiration using a special object: `CacheItemExpiration`, which implements `ICacheItemExpiration` interface. The code snippet in the following sample is the definition of the `ICacheItemExpiration` interface.

```
public interface ICacheItemExpiration
{
    bool HasExpired();❶
    void Initialize(CacheItem owningCacheItem);❷
    void Notify();❸
}
```

❶ Specifies if item has expired or not.

❷ Called to give the instance the opportunity to initialize itself from information contained in the `CacheItem`.

❸ Called to tell the expiration that the `CacheItem` to which this expiration belongs has been touched by the user.

There are two Add overloaded methods in `CacheManager` to add data into cache (Refer to the following code). The second Add method could accept an array of `ICacheItemExpiration` objects. Once a particular cache item, along with its `ICacheItemExpiration` object, is added by `CacheManager`, Leveraging cache expiration notification mechanism provided by `ICacheItemExpiration`, CAB can make the corresponding cache item expires in time.

```
public interface ICacheManager
{
    // Methods
    void Add(string key, object value);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    void Add(string key, object value,
             CacheItemPriority scavengingPriority,
             ICacheItemRefreshAction refreshAction,
             params ICacheItemExpiration[] expirations);
    ...
}
```

As for writing code, see the following sample via the second overload:

```
string settingHHMMSS = "*:*:30";
AbsoluteTimeExpiration absoluteTimeExpiration = new AbsoluteTimeExpiration(settingHHMMSS);
ICacheManager cacheManager = CacheFactory.GetCacheManager();
cacheManager.Add(CACHE_KEY_A, CACHE_VALUE, CacheItemPriority.Normal, null, absoluteTimeExpiration);
```

In this version of iConnect.NET, two special custom `CacheItemExpiration` will be defined to resolve cache synchronization, and they are `SqlDependencyExpiration` and `AbsoluteTimeExpiration`.

24.2.1. SqlDependencyExpiration

As its name suggests, `SqlDependencyExpiration` is a custom `CacheItemExpiration`, which leverages `SqlDependency` to keep synchronized between the cached item and the data stored in SQL Server table.In essence, `SqlDependencyExpiration` depends on `SqlDependency` object. About the constructor of `SqlDependency` object there are are three constructors defined as below:

```
public SqlDependency();
public SqlDependency(SqlCommand command);
public SqlDependency(SqlCommand command❶, string options❷, int timeout❸);
```

❶ The `SqlCommand` object to associate with this `SqlDependency` object. The constructor sets up a `SqlNotificationRequest` object and bind it to the command.

❷ The notification request options to be used by this dependency. Null to use the default service.

❸ The time-out for this notification in seconds. The default is 0, indicating that the server's time-out should be used.

The mechanism of `SqlDependencyExpiration` is simply said that actually there always is a specific SQL query being executed in database at regular intervals under background. Once the result of the SQL query changed, the cached data would be cleared and the client would receive the notification from server. So, after the simple introduction, the following constructors of `SqlDependencyExpiration` are understood easier:

```
SqlDependencyExpiration(string commandText);
SqlDependencyExpiration(string commandText, IDictionary<string, object> parameters);
SqlDependencyExpiration(string commandText, string connectionStringName,
    IDictionary<string, object> parameters);
SqlDependencyExpiration(string commandText, CommandType commandType,
    IDictionary<string, object> parameters);
SqlDependencyExpiration(string commandText, CommandType commandType,
   string connectionStringName, IDictionary<string, object> parameters);
SqlDependencyExpiration(string commandText, CommandType commandType,
    string connectionStringName, IDictionary<string, object> parameters,
    string options, int timeout);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Obviously, the last one is the core of the constructors above. According to above description, it's simply explained about the parameters of the core constructor. "*commandText*", "*commandType*", "*connectionStringName*" and "parameters" are all used to build a SQL command. The `SqlDependency` object is built through the built command, "options" and "timeout". Thus, `SqlDependency` performs the SQL query to detect changes at regular intervals.

On the another hand, there is an event Expired in `SqlDependencyExpiration` so that developers can extend the business logic after cache expiration. The use of `SqlDependencyExpiration` is just as following code:

```
SqlDependencyExpiration sqlDependencyExpiration = new SqlDependencyExpiration(
  "SQL Query Clause",
  CommandType.Text,
  CONNECTION_NAME,
  null);
sqlDependencyExpiration.Expired += delegate
{
   // Extension logic
};

ICacheManager cacheManager = CacheFactory.GetCacheManager();
cacheManager.Add(CACHE_KEY, CACHE_VALUE, CacheItemPriority.Normal, null, sqlDependencyExpiration);
```

Above sample only used one of the constructors, developers need to take the different constructor according to different scenario.

Caution

Because the notification mechanism of `SqlDependency` depends on Service Broker in SQL Server 2005. So, anyway SQL query clause or stored procedure have to follow the restriction based on Service Broker in SQL Server 2005.For the restriction, refer to: http://msdn.microsoft.com/en-us/library/ms181122(SQL.90).aspx.

24.2.2. AbsoluteTimeExpiration

Another solution to achieve across-machine cache synchronization is making the caches item at the same time, so except for `SqlDependency` based custom `CacheItemExpiration`, we will create another absolute time based one: `AbsoluteTimeExpiration`. `AbsoluteTimeExpiration` can provide the following three expiration mode:

- `PerDay`: The cache item will expire at the time specified every. E.G. The cache item will expire at 00:00:00 every day.
- `PerHour`: The cache item will expire at the time specify every hour. E.G. The cache item will expire at XX: 00:00 every hour.
- `PerMinute`: The cache item will expire at the time specify every minute. E.G. The cache item will expire at XX: XX: 00 every minute.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

We can use a literal express to define the expiration time and mode, and three expiration mode can be defined as below:

- 00:00:00
- *:00:00
- *:*:00

To utilize `AbsoluteTimeExpiration`, the key is how to construct the instance of `AbsoluteTimeExpiration`. For this, `AbsoluteTimeExpiration` has only two constructors:

```
AbsoluteTimeExpiration(string expirationTime);❶
AbsoluteTimeExpiration(string expirationTime, SynchronizedTimeProvider timeProvider);❷
```

❶ The constructor needs only expiration time as unique parameter. In essence, by default iConnect.NET provides `SynchronizedTimeProvider` to specify the standardized time as the synchronized time between application servers.

❷ Contrast to above one, developers can have choice to customize `SynchronizedTimeProvider` to generate the synchronized time on own standard.

As for applying `AbsoluteTimeExpiration` for the application, the sample code as following:

```
// perDay mode
string settingHHMMSS = "12:00:00";
// perHour mode
string settingHHMMSS = "*:00:00";
// perMinute
string settingHHMMSS = "*:*:00";

AbsoluteTimeExpiration absoluteTimeExpiration = new AbsoluteTimeExpiration(settingHHMMSS);
ICacheManager cacheManager = CacheFactory.GetCacheManager();
cacheManager.Add(CACHE_KEY_A, CACHE_VALUE, CacheItemPriority.Normal, null, absoluteTimeExpiration);
```

Of course, if developers had made an override for `SynchronizedTimeProvider`, the constructor of `AbsoluteTimeExpiration` object could utilize the second overloaded constructor against customized `SynchronizedTimeProvider`.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 25. Letter Management System (LMS)

25.1. Introduction

LMS is a document processing engine, which provides an extensible pipeline to process an original document and generate a new document we want. The original raw document is usually a template file with a series of placeholder or fields, the mail merge document handler registered in LMS can replace these placeholders or fields with the provided data sources to generate a complete document. Besides the basic functionality mail merge which binds the data source to the template, other related document processing tasks can be also conducted by LMS. E.G. Convert a MS Word document to PDF format; add a watermark to the document, etc. In addition, LMS document processing pipeline contains the document delivery part which enables us to deliver the document in different manners.

25.2. Terminologies

The following list LMS related Terminologies:

- *Document processing job:* It represents an orchestrated processing flow to process one or more documents. A job (the job represents a document processing job from now) is the basic executable unit in LMS.
- *Document process job item:* A job can process multiple documents in a batch, and document to process corresponds to one document processing job items. In other words, one job is a collection of job items.
- *Document processing context:* During running a particular job, each of its job items is executed in a context, which is named document processing context.
- *Document handler:* Concrete document processing work is performed by the document handlers. E.G. Mail merge is achieved by mail merge handler, and the file convert handler can be used to convert a document from one format to another one.
- *Document store:* It is used to store the generated documents before delivering. At other hand, loading the generated document before really delivering it is also completed by it.
- *Document deliver:* At the end of the document processing life cycle, the document delivery notification is sent by document deliver to the document delivery service.
- *Document delivery service:* It is a WCF service with `NetMsmqBinding` based endpoint. Once receiving the document delivery notification sent from document deliver, it drives the registered document delivery listeners to deliver the documents.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- *Document delivery listener:* Document delivery listener is responsible for deliver the document. E.G. There are listeners used to upload the documents to the target server or send a mail in which the delivered document becomes a attachment.
- *Document processing job tokens:* A data dictionary attached to a job to store job based properties.
- *Document processing job item tokens:* A data dictionary attached to a job item to store job item based properties.
- *Document processing context tokens:* A data dictionary attached to a document processing context to store context based properties.
- *Document processing job initializer:* It is used to initialize the created document processing job before running it.
- *Document processing job item initializer:* It is used to initialize the job's items before running the it.
- *Document processing context initializer:* It is used to initialize the created document processing context before processing the document in it.
- *Document processing job extension:* A job is an extensible object, and we can attach any extension to it before running it. In general, we control the job is executed in the manner we expect by attaching some extension we need. Document processing job extension greatly embodies LMS's extensibility.

25.3. Getting Started

This section provides a sample which demonstrates how the templates based document is processed and delivered. In this sample, there are two template documents to process. After there two are mail merged, one of which is used the body of a mail to send, and the other one becomes the attachment of the mail. In addition, the mail attachment document will be converted from MS Word format to a PDF file. Such a mail will be finally sent to the specified receipt. Because we mail merge is achieved by Asppose.Word, please make sure Aspsose.Word is installed before you program this demonstrative application.

Step I: Create a solution containing the projects to process the documents deliver the documents

Create an empty solution using VS 2010 or VS 2012. We then add two console applications, which are separately named *LmsDemo* and *DeliveryService*. Add the following assembly references to the two projects:

- `System.ServiceModel`
- `System.Messaging`

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- `NCS.IConnect.Lms.Core`
- `NCS.IConnect.Lms.Extensions`
- `NCS.IConnect.Lms.Extensions.Aspose`

We add an *App.config* to the *DeliveryService* project, and copy the following configure snippet to override the existing content. Do not forget the appropriate connection string accordingly the database you use.

```
<configuration>
  <configSections>
    <section name="iConnect.lms"
             type="NCS.IConnect.Lms.Configuration.LmsSettings,
                   NCS.IConnect.Lms.Core"/>
  </configSections>
  <system.serviceModel>
    <bindings>
      <netMsmqBinding>
        <binding exactlyOnce="true">
          <security mode="None"/>
        </binding>
      </netMsmqBinding>
    </bindings>
    <services>
      <service name="NCS.IConnect.Lms.DocumentDeliveryService">
        <endpoint address="net.msmq://localhost/private/lmsdemo"
                  binding="netMsmqBinding"
                  contract="NCS.IConnect.Lms.IDocumentDeliveryService"/>❶
      </service>
    </services>
  </system.serviceModel>
  <connectionStrings>
    <add name="iConnectDb"
         providerName="System.Data.SqlClient"
         connectionString="..."/>
  </connectionStrings>
  <system.net>
    <mailSettings>
      <smtp from="...">
        <network host="..."
                 password="..."
                 port="..."
                 userName="..."/>
      </smtp>❷
    </mailSettings>
  </system.net>
  <iConnect.lms defaultProvider="dbProvider" ❸
                defaultDocumentStore="ShareFolderDocumentStore"❹>
    <providers>
      <add name="dbProvider"
           type="NCS.IConnect.Lms.DbDocumentProcessingJobProvider,
                 NCS.IConnect.Lms.Extensions"
           applicationName="LmsDemo"
           database="iConnectDb"/>❺
    </providers>
    <deliveryListeners>
      <add name="FileSaverDeliveryListener"
           type="NCS.IConnect.Lms.FileSaverDeliveryListener,
                 NCS.IConnect.Lms.Extensions"
           directory="C:\Lms\Documents"/>❻
      <add name="SmtpDeliveryListener"
           type="NCS.IConnect.Lms.SmtpDocumentDeliveryListener,
                 NCS.IConnect.Lms.Extensions"/>❼
    </deliveryListeners>
    <documentStores>
      <add name="ShareFolderDocumentStore"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            type="NCS.IConnect.Lms.ShareFolderDocumentStore, NCS.IConnect.Lms.Core"
            directoryToLoadDocuments="C:\Lms\Workspace\Templates"
            directoryToSaveDocuments="C:\Lms\Workspace\Documents"/>❽
    </documentStores>
  </iConnect.lms>
</configuration>
```

❶ The setting of the document delivery service, which is a queued service with a NetMsmqBinding based endpoint.

❷ The SMTP setting. Please configure this according to concrete network environment.

❸ The configuration name of the default document processing job provider.

❹ The configuration name of the default document store.

❺ The configuration of DbDocumentProcessingJobProvider, which provides job related information by accessing to the specified data base.

❻ The `FileSaverDeliveryListener` which saves the document to deliver to the specified directory, C:\Lms\Documents.

❼ The `SmtpDocumentDeliveryListener` which deliver the documents by sending a mail to the expected receipt.

❽ The `ShareFolderDocumentStore` which save the generated documents into the specified share folder, and loads the original template files from the specified share folder. According to the configuration, the templates should be stored in the directory *C:\Lms\Workspace\Templates*, and the generated documents will be stored in the directory *C:\Lms\Workspace\Documents* before they are successfully delivered.

In the Main method of the Program.cs of the *DeliveryService* project, the following code is programed to host the document delivery service. In addition, the message queue used by such a queued service is created if it does not exist. What is created is a transactional message queue, and must be a transactional queue for delivery auto retry and error logging..

```
class Program
{
    static void Main(string[] args)
    {
        string path = @".\private$\lmsdemo";
        if (!MessageQueue.Exists(path))
        {
            MessageQueue.Create(path, true);
        }
        using (ServiceHost host = new ServiceHost(typeof(DocumentDeliveryService)))
        {
            host.Opened += delegate { Console.WriteLine("LMS Delivery Service is opened..."); };
            host.Open();
            Console.Read();
        }
    }
}
```

As with console application *LmsDemo* which is used to process the document, we add the following configuration snippet in the App.config. In the LMS configuration section, the same document store and document processing job provider are configured. There are two kinds of document handlers registered. The 1st `MailMerge` handler is used to perform mail merge, and the 2nd one `ConvertFormatDocumentHandler` is used to perform file format conversion.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<configuration>
  <configSections>
    <section name="iConnect.lms"
             type="NCS.IConnect.Lms.Configuration.LmsSettings,
                   NCS.IConnect.Lms.Core"/>
  </configSections>
  <system.serviceModel>
    <bindings>
      <netMsmqBinding>
        <binding exactlyOnce="true">
          <security mode="None"/>
        </binding>
      </netMsmqBinding>
    </bindings>
    <client>
      <endpoint name="deliveryservice"
                address="net.msmq://localhost/private/lmsdemo"
                binding="netMsmqBinding"
                contract="NCS.IConnect.Lms.IDocumentDeliveryService"/>
    </client>
  </system.serviceModel>
  <connectionStrings>
    <add name="iConnectDb"
         providerName="System.Data.SqlClient"
        connectionString="..."/>
  </connectionStrings>
  <iConnect.lms defaultProvider="dbProvider"
                defaultDocumentStore="ShareFolderDocumentStore"
               deliveryEndpoint="deliveryservice"❶>
    <providers>
      <add name="dbProvider"
           type="NCS.IConnect.Lms.DbDocumentProcessingJobProvider,
                 NCS.IConnect.Lms.Extensions"
           applicationName="LmsDemo"
           database="iConnectDb"/>
    </providers>
    <documentHandlers>
      <add name="mailMerge"
           type="NCS.IConnect.Lms.Aspose.MailMerge,
                 NCS.IConnect.Lms.Extensions.Aspose"
           order="1"/>❷
      <add name="fileConverter"
           type="NCS.IConnect.Lms.Aspose.ConvertFormatDocumentHandler,
                 NCS.IConnect.Lms.Extensions.Aspose"
           order="2"/>❸
    </documentHandlers>
    <documentStores>
      <add name="ShareFolderDocumentStore"
           type="NCS.IConnect.Lms.ShareFolderDocumentStore, NCS.IConnect.Lms.Core"
           directoryToLoadDocuments="C:\Lms\Workspace\Templates"
           directoryToSaveDocuments="C:\Lms\Workspace\Documents"/>
    </documentStores>
  </iConnect.lms>
</configuration>
```

❶ The configuration name of the endpoint to invoke document delivery service.

???

❷ The `MailMerge` document handler to fill the fields in the template using the provided data sources.

❸ The `ConvertFormatDocumentHandler` to perform file format conversion, The mail attachment PDF file is converted by this document handler from MS Word document.

Step II Create Application Record

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by iConnect.NET to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You firstly need to execute the following SQL script in this database to insert a new record, which stands for our demonstration application into the T_IC_APP table. Our demonstration application is named "*LmsDemo*".

```
INSERT INTO [dbo].[T_IC_APP] (
    [APP_ID]
    ,[APP_NAME]
    ,[LOWERED_APP_NAME]
    ,[APP_DESC]
    ,[FLAG]
    ,[VERSION_NO]
    ,[TRANSACTION_ID]
    ,[CREATED_BY]
    ,[CREATED_TIME]
    ,[LAST_UPDATED_BY]
    ,[LAST_UPDATED_TIME])
VALUES (
   'A089D23B-7C1F-4AB0-9E9F-1396BEB73BA1'
    ,'CodeTableDemo'
    ,'codetabledemo'
    ,''
    ,''
    ,1
    ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7'
    ,'Sys'
    ,GETDATE()
    ,'Sys'
    ,GETDATE())
GO
```

Step III Create the templates

As mentioned in the above section, two templates are needed, which are separately used as the cover letter and the attachment of the mail to send. These two templates are two MS Word documents with mail merge fields. The following figure illustrates cover letter based MS Word document based template. In this document, there are only two mail merge fields, `FirstName` and `LastName`. As with the rest free text, you can replace them with any words. We name this template "CoverLetter.docx" and save it into the directory "C:\Lms\Workspace\Templates" where the configured default document store loads the templates.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

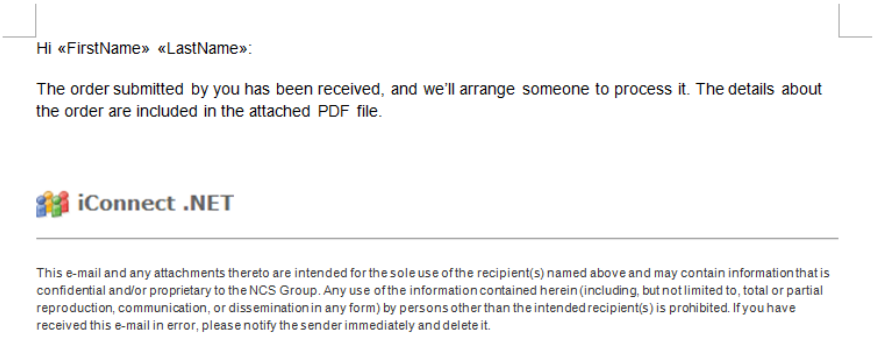


Figure 25.1. Cover letter template

Note

If we need to insert a mail merge field in an MS Word document, we can open the *Field* dialog by selecting *Insert->Quick Parts->Field*. After selecting the *MergeField* item from the Field names drop down list and specifying the file name (like FirstName and LastName), the field will be inserted into the document after clicking OK button.

According to the content of the cover letter template we know this is a notification sent to the person whose submit an order. The details about the order information is included in a separate document, which is used as the attachment of the mail. When this template is merged by the mail merge document handler, the fields will be filled by extracting the corresponding data from data sources. What is used as the data source is a strongly typed DataSet, which includes two data tables of parent-child relationship. The template document and data source schema are both illustrated in the following figure.

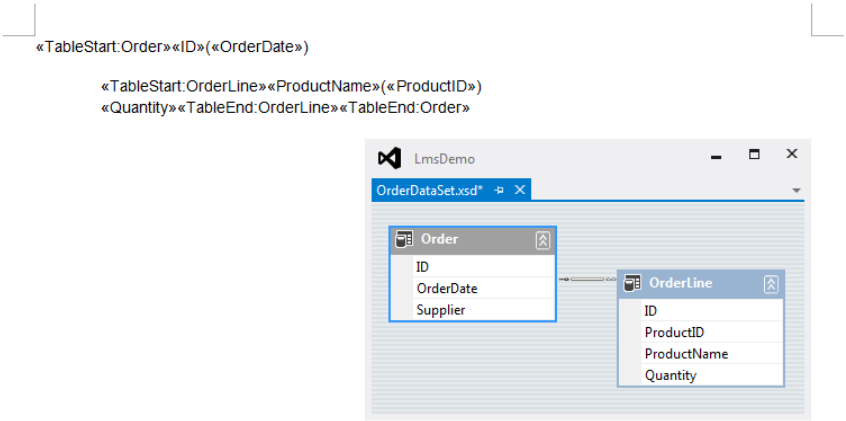


Figure 25.2. Order List Template and DataSet Schema

Unlike the two fields in the cover letter template, FirstName and LastName, which are bound to scalar data fields, the fields in the order list template are regional fields. The region between the filed whose name prefixes with TableStart: (E.G. "TableStart:Order") and the one whose name prefixes with the matched TableEnd: (E.G, :"TableEnd: Order") corresponds to a table's

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

row. It will automatically grow when the source table has multiple rule. For the above template, if there are multiple rows in the data Order, all of the order records will be listed. If an order record owns more than one order lines, the order line records are also listed in each order region.

We name this template document "OrderList.docx" and save it into the directory "C:\Lms\Workspace\Templates" where is cover letter template is also stored.

Step IV Process and Deliver Documents

We program the following code in the *LmsDemo* project to process the document and finally send notification to the delivery service. An empty job is created by calling the current LmsManager's CreateEmptyJob method. Two data tokens are then added to the job. The token "MailMessage" holds an MailMessage object carrying the email addresses and mail subject information. The other one "DeliverDocumentsViaEmail" to involve the registered SmtpDocumentDeliveryListener into the current delivery.

Two job items, which separately correspond to the cover letter template and attachment template, are created and then added to the job. The 1st job item has two tokens. The token TargetDocumentFormat determine the format of the finally generated document. In this sample, the format is the mail body's format (HTML), Another token "IsCoverLetterDocument" indicating the generated document will be used as the cover letter.

As with the attachment template based job item, there are also two data tokens attached. The token "TargetDocumentFormat" determine the attachment is a PDF file. The other one "TargetFileName" specifies the attachment name.

Finally, we run the job by calling its Run method. The two arguments passed into this method are the data sources. The 1st one created by calling the method CreateScalarDataSources() is the data source to fill the two fields in the cover letter template, and the 2nd one created by calling the method CreateRegionalDataSources() corresponds to attachment template.

```
class Program
{
    static void Main(string[] args)
    {
        //Create an empty job.
        DocumentProcessingJob job = LmsManager.Current.CreateEmptyJob();
        //Create an MailMessage and save as job's data token.
        MailMessage mailMessage = new MailMessage("<<from maill address>>",
            "Submited order is received");
        mailMessage.To.Add("<<to mail address>>");
        job.DataTokens.Set(DataTokenNames.MailMessage, mailMessage);
        //The DeliverDocumentsViaEmail token is added to indicate to deliver
        //documents by sending an mail.
        job.DataTokens.Set(DataTokenNames.DeliverDocumentsViaEmail, true);

        //Add cover letter specific job item.
        using (Stream stream1 = LmsManager.Current.DocumentStore
            .GetInputStreamToReadTemplateDocument("CoverLetter.docx"))
        {
            DocumentProcessingJobItem jobItem1 = DocumentProcessingJobItem
                .CreateByStream(stream1, DocumentFormat.Docx);
            //Add two tokens:
            //TargetDocumentFormat: Target document format (Html)
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            //IsCoverLetterDocument: Is cover letter or not
            jobItem1.DataTokens.Set(DataTokenNames.TargetDocumentFormat, DocumentFormat.Html);
            jobItem1.DataTokens.Set(DataTokenNames.IsCoverLetterDocument, true);
            job.AddJobItem(jobItem1);
        }

        //Add attachment specific job item.
        using (Stream stream2 = LmsManager.Current.DocumentStore
            .GetInputStreamToReadTemplateDocument("OrderList.docx"))
        {
            DocumentProcessingJobItem jobItem2 = DocumentProcessingJobItem
                .CreateByStream(stream2, DocumentFormat.Docx);
            //Add two tokens:
            //TargetDocumentFormat: Target document format (PDF)
            //TargetFileName: Attachment file name (Order(<<Timestamp>>)
            jobItem2.DataTokens.Set(DataTokenNames.TargetDocumentFormat, DocumentFormat.Pdf);
            jobItem2.DataTokens.Set(DataTokenNames.TargetFileName,
                string.Format("Order({0})", DateTime.Now.ToString("yyMMddhhmmss")));
            job.AddJobItem(jobItem2);
        }
        job.Run(CreateScalarDataSources(), CreateRegionalDataSources());
    }

    private static IEnumerable<object> CreateScalarDataSources()
    {
        Dictionary<string, object> dictionary = new Dictionary<string, object>();
        dictionary.Add("FirstName", "Henry");
        dictionary.Add("LastName", "Wang");
        return new object[] { dictionary };
    }

    private static IEnumerable<object> CreateRegionalDataSources()
    {
        OrderDataSet ds = new OrderDataSet();

        ds.Order.AddOrderRow("O001", DateTime.Today, "Dell");
        ds.Order.AddOrderRow("O002", DateTime.Today, "HP");

        ds.OrderLine.AddOrderLineRow(ds.Order[0], "P001", "Printer", 2);
        ds.OrderLine.AddOrderLineRow(ds.Order[0], "P002", "PC", 10);
        ds.OrderLine.AddOrderLineRow(ds.Order[1], "P003", "Projector", 5);
        ds.OrderLine.AddOrderLineRow(ds.Order[1], "P004", "Phone", 30);
        return new object[] { ds };
    }
}
```

Step V: Check mail and uploaded file

After running the LmsDemo application after running the DeliveryService application, a mail will send to the receipt whose email address is specified in the above program. As the following figure illustrates, the mail body's content comes from the cover letter template and two fields are successfully replaced accordingly.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

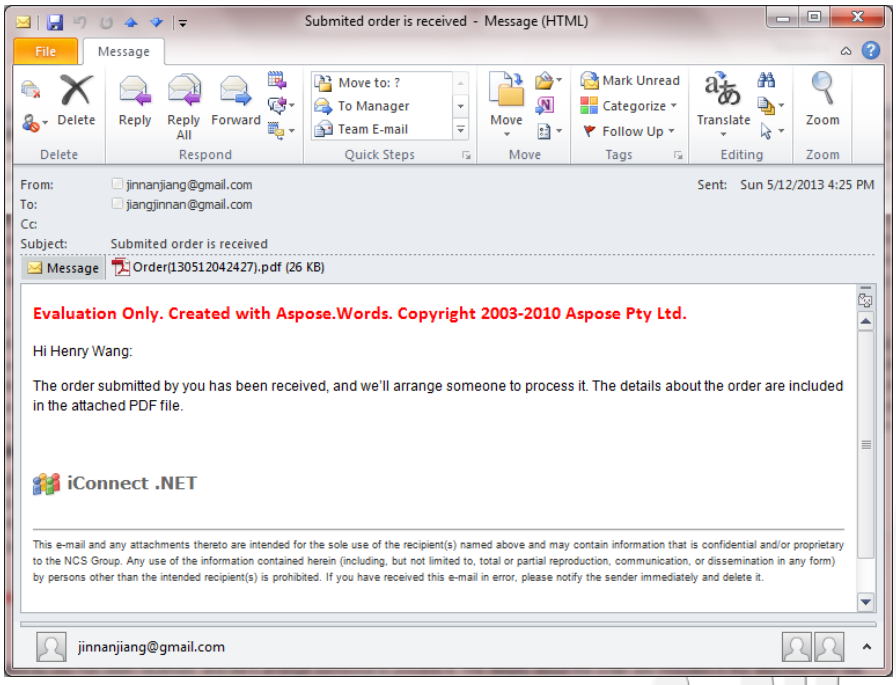


Figure 25.3. Received mail

As illustrated by the above figure, the mail has a PDF file based attachment. The order detailed information are all included in this PDF file. The following figure illustrates the content of this PDF file.

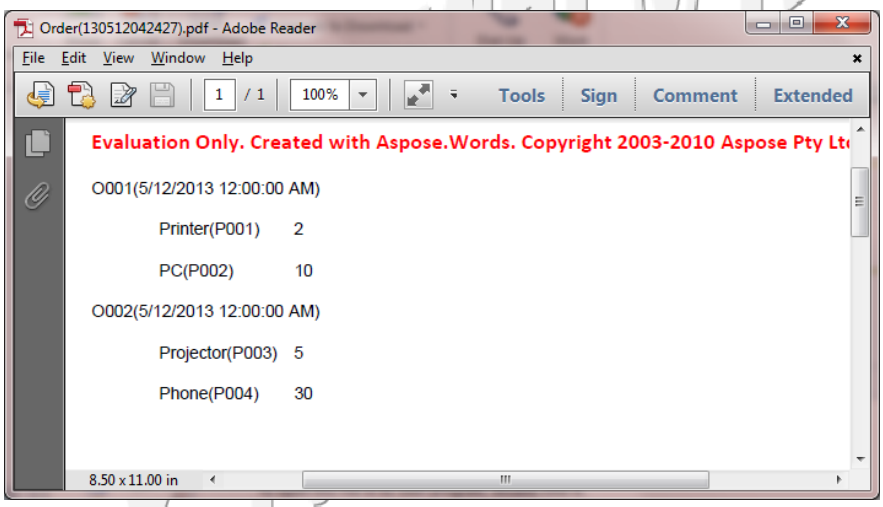


Figure 25.4. PDF file as mail attachment

Note

There is extra line of words in red: "*Evaluation Only. ...*" in both the mail body and attachment PDF. The reason is the Aspose installed is evaluation edition.

In the App.config in the DeliveryService project, besides the SmtpDocumentDeliveryListener registered, there is a FileSaverDocumentDeliveryListener. Such a delivery listener saves all documents to deliver into the specified directory, "C:\Lms\Documents". As illustrated in the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

following figure, there are totally three files in this directory. The ,html and .pdf files are separately the cover letter file and the attachment file. The .png file is the image which is referenced in the HTML file. In the corresponding MS Word document, it is the embedded iConnect.NET logo.

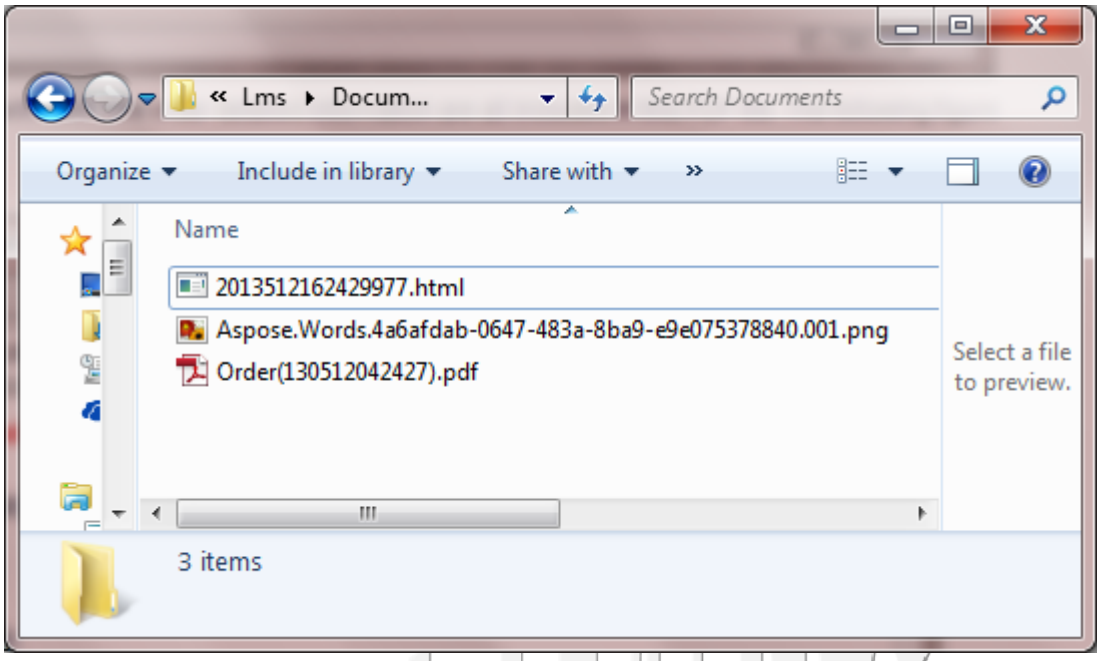


Figure 25.5. Uploaded files

25.4. Document Processing Pipeline

The following figure illustrates LMS document processing pipeline, and the basic document processing & delivery flow is like this:

- *Initialize document processing job:* When we run a document processing job, the job initializers and job item initializers will be used to initialize the job itself.
- *Execute document handler pipeline:* Once the job initialization completes, each job item will be executed one after another. Before starting executing each job item, a separate document processing context is created and it can be considered as the execution context of the job item. After the context is created, the document processing context initializers which are pre-registered to the job are used to initialize this context.

  Each job item owns one raw document, which will be used as the context's document. Such a document will be processed by the registered document handler pipeline. In detailed, each document handler will load the document directly from context and release it to the context after processing it. The final document processed by the document handler pipeline will be temporarily stored by the document store. When the document is temporarily stored and unique document ID will be generated, and the document store will use it to get the corresponding document when performing delivery.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- *Deliver documents:* If the finally generated documents need to be delivered, the document deliver will send a notification to the document delivery service. Once receiving the document delivery notification, the delivery service will select the matched delivery listeners to perform concrete delivery.

LMS's document processing pipeline and the general document processing flow is illustrated in the following figure.

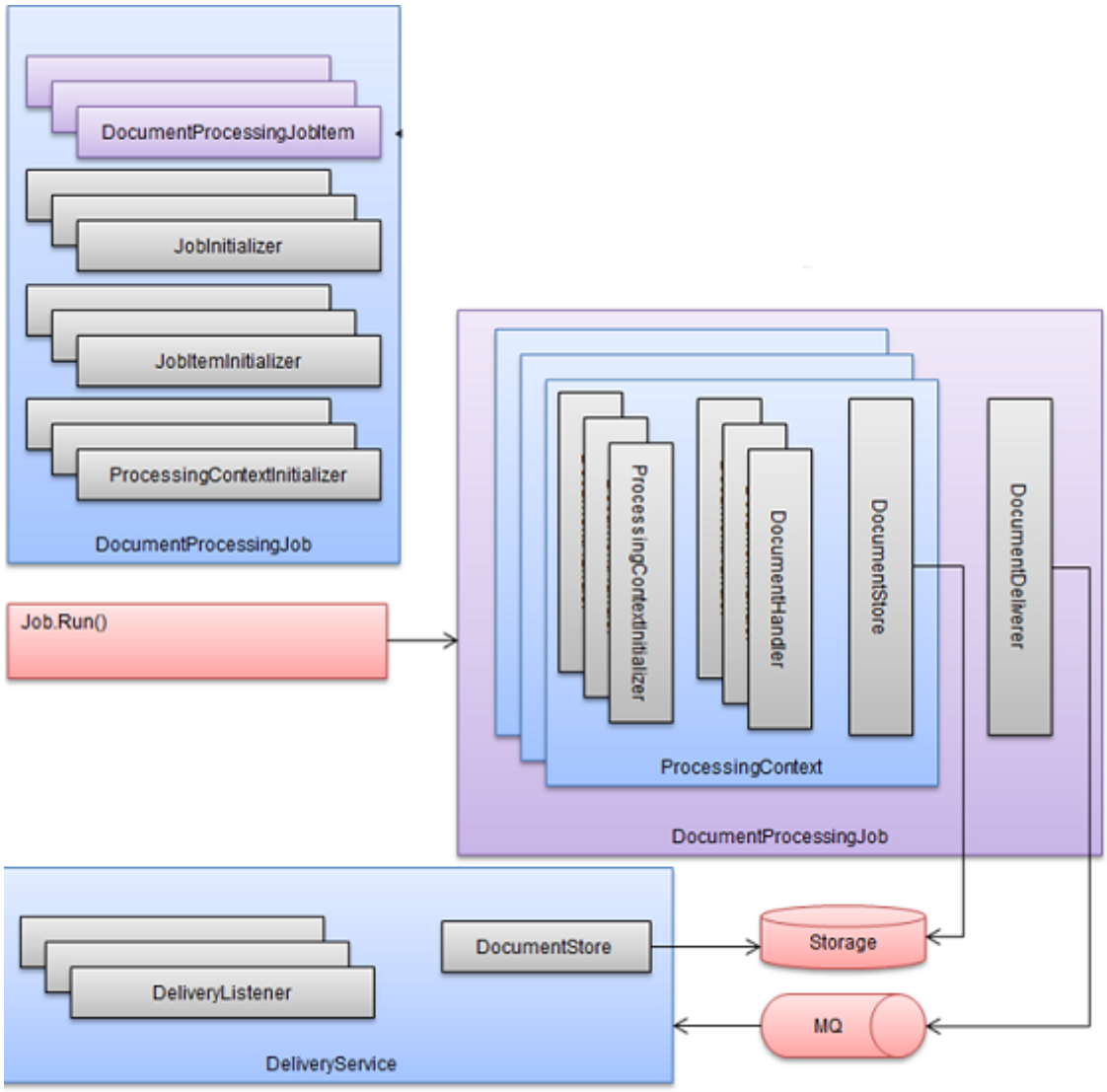


Figure 25.6. Document Processing Pipeline

25.5. Document Processing Job

In LMS, a document processing job is the basic executable unit. It represents an orchestrated work flow to process the given document(s) and finally generate and deliver the new document(s). All of the document processing work is driven by a job, so it is the core of LMS. In LMS application programming interface, a job is represented by the class `DocumentProcessingJob`.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

25.5.1. Data Token

Data token in LMS can be considered as a data dictionary item. It is attached to a particular object to provide the owner with a data container. In LMS, the job, job item and document processing context all own a data token dictionary which controls how the document should be processed.

In LMS application programming interface, a data token dictionary is represented by the class `DataTokenDictionary`. As the following code snippet illustrates, we can invoke the two `Attach` methods to directly attach the items from a given dictionary object. Two `Get` methods is used get the token value based on the specified token name. The `Set` method will add a new token into the dictionary or override the existing one.

```
public class DataTokenDictionary
{
   //...
   public DataTokenDictionary();
   public void Attach(IDictionary<string,object> dictionary);
   public void Attach(IDictionary<string,string> dictionary);
   public object Get(string tokenName);
   public TGet<T>(string tokenName, T DefaultValue = default(T));
   public void Set(string tokenName, object value); }
```

A `DataTokenDictionary` can be attached to another `DataTokenDictionary` object, and becomes its one child. `DataTokenDictionary`'s read-only property Parent represents its parent dictionary, and we can call the `ChangeParent` method to set or change a particular `DataTokenDictionary`'s parent.

```
public class DataTokenDictionary
{
    //...
    public void ChangeParent(DataTokenDictionary parent);
    public DataTokenDictionary Parent { get; }
}
```

If a `DataTokenDictionary` has parent, when we call the `Get` method to get the token value based on the given token name, if such a token does not exists in the current dictionary, the parent dictionary's `Get` method will be recursively called. However, what the `Set` method operates is always the current dictionary itself.

`DataTokenDictionary` can be frozen. Once frozen, a `DataTokenDictionary` will become read-only, and exception will be thrown for any modification attempt. As the following code snippet illustrates, the read-only property IsFrozen indicates if the `DataTokenDictionary` is frozen. We can call the `Freeze` method to freeze the `DataTokenDictionary`.

```
public class DataTokenDictionary
{
    //...
    public voidFreeze();
    public bool IsFrozen { get; }
}
```

The data tokens attached to a job are stored in the `DocumentProcessingJob`'s DataTokens property.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public class DocumentProcessingJob :
      IExtensibleObject<DocumentProcessingJob>
{
    //...
    public DataTokenDictionary DataTokens { get; }
}
```

25.5.2. Document

The documents in LMS are not limited to any file format. The document processed in LMS pipeline is represented by the class `Document`. The Content property represents the content of the document in the form of byte array, and Encoding represents the text encoding. We can create a `Document` by the specified file name (path) or input stream to read the content.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public sealed class DocumentFormat
{
    public static DocumentFormatCreate(string extension);
    [DataMember] public string Extension { get;}
}
```

The 4 built-in DocumentFormat (Doc, Docx, Html and Pdf) are directly defined in the class `DocumentFormat` in the form of static property.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public sealed class DocumentFormat
{
    //...
    public static DocumentFormat Doc { get; }
    public static DocumentFormat Docx { get; }
    public static DocumentFormat Html { get; }
    public static DocumentFormat Pdf { get; }
}
```

25.5.3. Job and Job Item

A job is not limited to process only one document. A job is a collection of job items, each of which corresponds to a single document processing. As the following code snippet illustrates, there is a JobItems property which represents its job item collection. We can call the method `AddJobItem` to add the specified job item to the collection.

```
public class DocumentProcessingJob :
      IExtensibleObject<DocumentProcessingJob>
{
    //...
    public IEnumerable<DocumentProcessingJobItem> JobItems{get;}
    public void AddJobItem(DocumentProcessingJobItem jobItem);
}
```

The class `DocumentProcessingJobItem` represents a job item. Like the `DocumentProcessingJob`, `DocumentProcessingJobItem` also has a DataTokens property to stored job item based data tokens. Its property Job refers to the job to which it is belong. It is worth emphasizing that the job item's data token dictionary's parent is its job based one.

```
public class DocumentProcessingJobItem
{
    //...
    public static DocumentProcessingJobItem CreateByFileName(string fileName,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        DocumentFormat format = null, Encoding encoding = null);
    public static DocumentProcessingJobItem CreateByStream(Stream stream,
        DocumentFormat format = null, Encoding encoding = null);
    public DataTokenDictionary DataTokens { get; }
    public DocumentProcessingJob Job { get; set; }
    public Document Document { get; set; }
}
```

The property Document holds a `Document` instance, which represents the original raw document to process. We can create a new `DocumentProcessingJobItem` based on the specified document file name (path) or the input stream to read the document content by calling the static factory methods `CreateByFileName` and `CreateByStream`.

A job with its job items can be persisted into database. The persistently stored job item has a unique name. We can directly create a job item based on its name by calling the `CreateByJobItemName` method. In addition, template based documents can be separately maintained in LMS, we can also create a job item based on the specified template name by calling the `CrateByTemplateName`. As the following code snippet illustrates, there are two extra properties, JobItemName and TemplateName, which represent the job item's name and template name.

```
public class DocumentProcessingJobItem
{
    //...
    public static DocumentProcessingJobItem CreateByJobItemName(string jobItemName);
    public static DocumentProcessingJobItem CreateByTemplateName(string temlateName);
    public string JobItemName { get; set; }
    public string TemplateName { get; set; }
}
```

There are no public constructor defined in `DocumentProcessingJob`, we have to use the current `LmsManager`, which is LMS's facade class, to create `DocumentProcessingJob`. As the following code snippet illustrates, there are totally 5 factory extension methods defined to create a `DocumentProcessingJob`.

```
public static class LmsManagerExtensions
{
    public static DocumentProcessingJob CreateEmptyJob(this LmsManager manager);
    public static DocumentProcessingJob CreateJobByName(this LmsManager manager,
        string jobName);
    public static DocumentProcessingJob CreateJobByFileName( this LmsManager manager,
        string fileName, DocumentFormat format = null, Encoding encoding = null);
    public static DocumentProcessingJob CreateJobByStream(this LmsManager manager,
        Stream stream, DocumentFormat format, Encoding encoding = null);
    public static DocumentProcessingJob CreateJobByTemplateName( this LmsManager manager,
        string templateName);
}
```

We can call the `CreateJobByName` method to create a job based on its name, which uniquely identifies a persistent job. As with the rest three ones, what is created is a job with only one job item.

25.5.4. Job Initializer & Job Item Initializer

As their names suggest, the job initializer and job item initializer are separately used for job/job item initialization. The both initializer types separately implement two interfaces,

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`IDocumentProcessingJobInitializer` and `IDocumentProcessingJobItemInitializer`. Only one method Initialize is used to perform job/job item initiation.

```
public interface IDocumentProcessingJobInitializer :
      ISequentialEntry<IDocumentProcessingJobInitializer>
{
    void Initialize(DocumentProcessingJob job);
}
public interface IDocumentProcessingJobItemInitializer :
      ISequentialEntry<IDocumentProcessingJobItemInitializer>
{
    void Initialize(DocumentProcessingJobItem jobItem);
}
```

The both initializer interfaces have the same base interface, `ISequentialEntry<T>`. Once a class implements such an interface, we can streamline a set of objects as a pipeline. The position in the pipeline is determined by its Order property, and its Next property refers to the next one in the pipeline. We define the following extension method `BuildPipeline<T>` for the `IEnumerable<T>` type to build such a pipeline.

```
public interface ISequentialEntry<T> where T:ISequentialEntry<T>
{
    T Next { get; set; } int Order { get; }
}
public static class SequentialEntryCollectionExtensions
{
    public static T BuildPipeline<T>(this IEnumerable<T> collection)
        where T: ISequentialEntry<T>;
}
```

If more than one initializer objects are used to initialize the job or job item, the above `BuildPipeline<T>` method will be called to create an initializer pipeline, and each initializer will be executed one by one. The lower the Order property, the higher execution priority it owns.

Besides the following two kinds of initializers, there is the 3rd one, document processing context initializer, which corresponds to the following interface `IDocumentProcessingContextInitializer`.

```
public interface IDocumentProcessingContextInitializer :
      ISequentialEntry<IDocumentProcessingContextInitializer>
{
    void Initialize(DocumentProcessingContext context);
}
```

By separately implementing the above three initializer interface, the following three abstract base classes are defined. In the three base initializer classes, the implemented `Initialize` method will directly call the abstract `InitializeCore` method to perform real initialization, and then call the next initializer if it exists. All concrete initializer only needs to inherit these base classes and implement the initialization in the `InitializeCore` method.

```
public abstract class DocumentProcessingJobInitializerBase :
    IDocumentProcessingJobInitializer
{
    public DocumentProcessingJobInitializerBase(int order);
    public void Initialize(DocumentProcessingJob job);
    protected abstract void InitializeCore(DocumentProcessingJob job);
    public IDocumentProcessingJobInitializer Next { get; set; }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public int Order {get; }
}

public abstract class DocumentProcessingJobItemInitializerBase
    : IDocumentProcessingJobItemInitializer
{
    public DocumentProcessingJobItemInitializerBase(int order);
    public void Initialize(DocumentProcessingJobItem jobItem);
    protected abstract void InitializeCore(DocumentProcessingJobItem jobItem);
    public IDocumentProcessingJobItemInitializer Next { get; set; }
    public int Order { get; }
}

public abstract class DocumentProcessingContextInitializerBase :
    IDocumentProcessingContextInitializer
{
    public DocumentProcessingContextInitializerBase(int order);
    public void Initialize(DocumentProcessingContext context);
    protected abstract void InitializeCore(DocumentProcessingContext context);
    public IDocumentProcessingContextInitializer Next { get; set; }
    public int Order { get; }
}
```

The three kinds of initializers need to be registered to a concrete job. When a job begins to run, the registered initializers will be used to initialize the job, job item and document processing context before really processing the document. A `DocumentProcessingJob` uses the following three properties, JobInitializers, JobItemInitializers and ContextInitializers to store the registered initializers.

```
public class DocumentProcessingJob : IExtensibleObject<DocumentProcessingJob>
{
    //...
    public ICollection<IDocumentProcessingContextInitializer>
        ContextInitializers { get; }
    public ICollection<IDocumentProcessingJobInitializer> JobInitializers
        {get; }
    public ICollection<IDocumentProcessingJobItemInitializer>
      JobItemInitializers { get; }
}
```

As the child initializers of the above 3 base initializers, the following three delegate based initializers are defined. As illustrated in the following code snippet, the three concrete initializera leverage the delegate (`Action<T>` instance) provided when constructing to perform initialization.

```
public class DocumentProcessingJobInitializer :DocumentProcessingJobInitializerBase
{
    public DocumentProcessingJobInitializer( Action<DocumentProcessingJob>
       initializeJob, int order);
    protected override void InitializeCore(DocumentProcessingJob job);
    public Action<DocumentProcessingJob> InitializeJob { get; } }

public class DocumentProcessingJobItemInitializer
    :DocumentProcessingJobItemInitializerBase
{
    public
      DocumentProcessingJobItemInitializer(
      Action<DocumentProcessingJobItem> initializeJob, int order);
      protected override void InitializeCore(DocumentProcessingJobItem
      jobItem);
    public Action<DocumentProcessingJobItem>
      InitializeJobItem { get; }
}

public class DocumentProcessingContextInitializer
    :DocumentProcessingContextInitializerBase
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public DocumentProcessingContextInitializer(
        Action<DocumentProcessingContext> initializeContext, int order);
    protected override void InitializeCore(DocumentProcessingContext
        context);
    public Action<DocumentProcessingContext> InitializeContext { get; }
}
```

25.5.5. Extensions

LMS's extension mainly focuses on extending document processing job. As with the job itself, it is an extensible object. As the following code snippet illustrates, `DocumentProcessingJob` implements the interface `IExtensibleObject<DocumentProcessingJob>`, so we can define any extensions by implementing the `IExtension<DocumentProcessingJob>` interface to extend a job. In order make it easy to attach extensions to a given job, the following two `AddExtension<T>` methods are defined in the `DocumentProcessingJob`.

```
public class DocumentProcessingJob :IExtensibleObject<DocumentProcessingJob>
{
    public DocumentProcessingJob AddExtension<T>()
        where T: IExtension<DocumentProcessingJob>, new();
    public DocumentProcessingJob AddExtension(
        IExtension<DocumentProcessingJob> extension);
    IExtensionCollection<DocumentProcessingJob>
      IExtensibleObject<DocumentProcessingJob>.Extensions { get; }
}
```

In LMS, we define the following abstract class, `DocumentProcessingJobExtensionBase`, as the base class of all concrete extension. There are three abstract methods, `GetJobInitializers`, `GetJobItemInitializers`, `GetContextInitializers`, which separately return three collections of the different initializers. In the implemented `Attach` method, these initializers returned by the three abstract methods will be registered to the job.

```
public abstract class DocumentProcessingJobExtensionBase
    :IExtension<DocumentProcessingJob>
{
    protected DocumentProcessingJobExtensionBase();
    public void Attach(DocumentProcessingJob owner);
    public void Detach(DocumentProcessingJob owner);
    protected abstract IEnumerable<IDocumentProcessingContextInitializer>
        GetContextInitializers();
    protected abstract IEnumerable<IDocumentProcessingJobInitializer>
        GetJobInitializers();
    protected abstract IEnumerable<IDocumentProcessingJobItemInitializer>
        GetJobItemInitializers();
}
```

Only one concrete extension class, `DocumentProcessingJobExtension`, is defined in LMS. It is sub class of `DocumentProcessingJobExtensionBase`. The initializers returned from the three implemented abstract method come from the `Func<T>` and `Action<T>`. If the `Func<T>` is specified in the constructor, its return initializes will be directly used. If `Action<T>` is passed to the constructor, a `DocumentProcessingJobInitializer`, or `DocumentProcessingJobItemInitializer` or `DocumentProcessingContextInitializer` will be created.

```
public class DocumentProcessingJobExtension : DocumentProcessingJobExtensionBase
{
    public DocumentProcessingJobExtension(
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        Func<IDocumentProcessingContextInitializer> contextInitializerAccessor);
    public DocumentProcessingJobExtension(
        Func<IDocumentProcessingJobInitializer> jobInitializerAccessor);
    public DocumentProcessingJobExtension(
        Func<IDocumentProcessingJobItemInitializer> jobItemInitializerAccessor);
    public DocumentProcessingJobExtension(
        Action<DocumentProcessingContext> initializeContext,
        int contextInitializerOrder);
    public DocumentProcessingJobExtension(
        Action<DocumentProcessingJob> initializeJob,
        int jobInitializerOrder);
    public DocumentProcessingJobExtension(
        Action<DocumentProcessingJobItem> initializeJobItem,
        int jobItemInitializerOrder);
    public DocumentProcessingJobExtension(
        Func<IDocumentProcessingJobInitializer> jobInitializerAccessor,
        Func<IDocumentProcessingJobItemInitializer> jobItemInitializerAccessor,
        Func<IDocumentProcessingContextInitializer> contextInitializerAccessor);
    public DocumentProcessingJobExtension(
        Action<DocumentProcessingJob> initializeJob,
        int jobInitializerOrder,
        Action<DocumentProcessingJobItem> initializeJobItem,
        int jobItemInitializerOrder,
        Action<DocumentProcessingContext> initializeContext,
        int contextInitializerOrder);
    protected override IEnumerable<IDocumentProcessingContextInitializer>
        GetContextInitializers();
    protected override IEnumerable<IDocumentProcessingJobInitializer>
        GetJobInitializers();
    protected override
      IEnumerable<IDocumentProcessingJobItemInitializer>
      GetJobItemInitializers();
}
```

In order to make it easier to apply a `DocumentProcessingJobExtension` to a job, the following three extension methods, `InitializeJob`, `InitializeJobItem` and `InitializeContext`, are defined to `DocumentProcessingJob`.

```
public static class DocumentProcessingJobExtensions
{
    public static DocumentProcessingJob InitializeContext( this
        DocumentProcessingJob job, Action<DocumentProcessingContext>
        initializeContext, int order = -1);
    public static DocumentProcessingJob
        InitializeJob( this DocumentProcessingJob job,
        Action<DocumentProcessingJob> initializeJob, int order = -1);
    public static DocumentProcessingJob InitializeJobItem( this
        DocumentProcessingJob job, Action<DocumentProcessingJobItem>
       initializeJobItem, int order = -1);
}
```

25.6. Processing Documents

For a `DocumentProcessingJob`, we can attach or apply any extensions to it before running. Generally speaking, these extension provides a series of job initializers, job item initializers and document processing context initializers, to perform some initialization before we run it.

25.6.1. Run the Job

The job's state before running determines how the raw document attached to job item is processed. As with how to run a job, we can directly call the `DocumentProcessingJob`'s `Run` method.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

If what we process is a template and needs to perform mail merge to generate a complete document, we have to provide its data source. Data sources in LMS are categorized two types, scalar and regional. Scalar data sources are bound to the standalone fields, and regional ones are bound to a dynamically growing region (a loop). Regional data source is usually a collection or list.

There is a special stored procedure based data source, which runs the predefined stored procedure to get the data to fill the mail merge fields. If a template uses stored procedure data source, we need to provide the input arguments of the stored procedure when we run the job. As the following code snippet illustrates, there are totally three `Run` methods, and the parameters separately represents the two kinds of data source and the input arguments passed to stored procedure, which is a dictionary and the key indicates the parameter name without prefix line "@". The return value of the `Run` method represents the document delivery ID.

```
public class DocumentProcessingJob :IExtensibleObject<DocumentProcessingJob>
{
    //...
    public string Run(IDictionary<string, object> inputArguments = null);
    public string Run(IEnumerable<object> dataSources,
        IEnumerable<object> regionalDataSources);
    public string Run(IDictionary<string, object> inputArguments,
        IEnumerable<object> dataSources, IEnumerable<object>
        regionalDataSources);
}
```

25.6.2. Document Processing Context

Once we run the job, LMS will iterate its job item and execute them one by one. Each of these job item is executed in a separate document processing context. Document processing context is the execution context of job item. In LMS application programming interface, the document processing context is represented by the class `DocumentProcessingContext`. As illustrated by the following code snippet, we create a `DocumentProcessingContext` based on a job item, which will be referenced by its JobItem property.

```
public class DocumentProcessingContext
{
    public DocumentProcessingContext(DocumentProcessingJobItem jobItem);
    public DocumentProcessingJobItem JobItem { get; }
    public Document Document {get; set; }
    public DataTokenDictionary DataTokens { get; }
    public ICollection<IDataSource> DataSources { get; }
    public IDictionary<string, object> InputArguments { get; }
}
```

The `Document` object referenced by `DocumentProcessingContext`'s Document property represents the document to process. It originally comes from job item's raw document. `DocumentProcessingContext` also has its own data token dictionary which is referenced by its DataTokens property. It is worth emphasizing that job item's data token dictionary will become the parent of context's based one, and job item specific data token dictionary uses job specific one as its parent.

The data sources and input arguments passed when calling the job's `Run` method, and the corresponding properties are DataSources and InputArguments.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

25.6.3. Data Source

Inside LMS document processing pipeline, the data source is not represented by a raw object but by an instances whose class implements the `IDataSource` interface. As illustrated by the following code snippet, there are only two properties defined in this interface, the Data property represents the data bound to mail merge fields in the template. The IsRegional property indicates whether the data source is a scalar or regional.

```
public interface IDataSource
{
    object Data { get; }
    boolIsRegional { get; }
}
```

We also define the following abstract class, `DataSourceBase`, as the base class of all concrete data source classes. The abstract method `GetDataCore` is used to retrieve the data to fill the mail merge fields in the template. The `DataSourceBase` has one concrete sub class, `DataSource`, which use the specific delegate (`Func<object>`) to get the data.

```
public abstract class DataSourceBase : IDataSource
{
    public DataSourceBase(bool isRegional);
    protected abstract object GetDataCore();
    public object Data { get; }
    public bool IsRegional { get; }
}

public class DataSource : DataSourceBase
{
    public DataSource(Func<object> dataAccessor, bool isRegional);
    public DataSource(object data, bool isRegional);
    protected override object GetDataCore();
    public Func<object> DataAccessor { get; }
}
```

The above-mentioned stored procedure data source, which gets the data by executing the predefined stored procedure, is represented by the class `ProcedureDataSource`. The result to execute the stored procedure is a `DataSet` containing one or more data tables. The property TableNames represents names of these data tables. The Relations property returns a collection of `DataSourceRelation` object, which represents one relationship between two data table in the `DataSet`.

```
public class ProcedureDataSource : DataSourceBase
{
    public ProcedureDataSource(string procedureName,
        DocumentProcessingContext context,
        Func<string,IDictionary<string, object>, string[], DataSet> loadDataSet,
        bool isRegional, params string[] tableNames);
    public ProcedureDataSource(string procedureName,
        DocumentProcessingContext context,
        string databaseName, bool isRegional, params string[] tableNames);
    public DocumentProcessingContext Context { get; }
    public string ProcedureName { get; }
    public ICollection<DataSourceRelation> Relations { get; }
    public ICollection<string> TableNames { get; }
}

[DataContract(Namespace="http://www.ncs.com.sg/")]
public class DataSourceRelation
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public DataSourceRelation(string parentTable, string childTable,
        string parentColumn, string childColumn);
    [DataMember]
    public string ChildColumn { get; set; }
    [DataMember]
    public string ChildTable { get; set; }
    [DataMember]
    public string ParentColumn { get; set; }
    [DataMember]
    public string ParentTable { get; set; }
}
```

25.6.4. Document Handler

Once the document processing context is created based on the job item, all of the document processing context initializers are streamlined as a pipeline to initialize it. After completing context initialization, the registered document handlers will really process the document, which is referenced by the Document property of the current document processing context. It is worth emphasizing that the context will be frozen before executing the document handler pipeline.

In LMS application programming interface, all document handlers implement the interface `IDocumentHandler`. As the following code snippet illustrates, document processing is implemented in the only one `Process` method. Two input parameters separately represent the document to process, and the current document processing context. The returned `Document` object represents the document generated after processing the original document.

```
public interface IDocumentHandler : ISequentialEntry<IDocumentHandler>
{
    Document Process(Document document, DocumentProcessingContext context);
}
```

As well as the three above-mentioned initializer interfaces, `IDocumentHandler` also inherits the interface `ISequentialEntry<IDocumentHandler>`, so all of registered document handlers can be streamlined as a pipeline. We can directly register the document handler to a particular `DocumentProcessingJob`. As the following code snippet illustrates, `DocumentProcessingJob` has a DocumentHandlers property used to store all of the registered document handlers.

```
public class DocumentProcessingJob : IExtensibleObject<DocumentProcessingJob>
{
    //...
    publicICollection<IDocumentHandler> DocumentHandlers { get;}
}
```

We can also globally register document handlers in a configurable manner. The related configuration is as below. When we create a `DocumentProcessingJob` using current `LmsManager`, the configured document handlers will be automatically registered to the job.

```
<configSections>
  <section name="iConnect.lms"
           type="NCS.IConnect.Lms.Configuration.LmsSettings, NCS.IConnect.Lms.Core"/>
</configSections>
<iConnect.lms ...>
  <documentHandlers>
    <add name = "handler1"
         type = "handler type 1"
         order = "1" .../>
    <add name = "handler2"
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        type = "handler type 2"
        order = "2" .../>
        ...
  </documentHandlers>
</iConnect.lms>
```

When the document handler pipeline is built, the document from the current document processing context will be processed by the 1st handler. The new document generated by the document handler after processing the original one will be used to override the context's existing document. For the 2nd iteration, the document to process is the new one. In order words, the document processed by a particular handler (not the 1st handler) in the pipeline is the one generated by its previous handler.

We define the following abstract class, `DocumentHandlerBase`, as the base class of all concrete document handlers. There are two abstract methods, `WillProcessDocument` and `ProcessCore`. The 1st one determines whether the current handler will really process the document, and handler really processes the document in the 2nd method.

```
public abstract class DocumentHandlerBase : IDocumentHandler,
{
    public DocumentHandlerBase(int order);
    public Document Process(Document document, DocumentProcessingContext context);
    protected abstract Document ProcessCore(Document document,
        DocumentProcessingContext context);
    protected abstract bool WillProcessDocument(DocumentProcessingContext context);
    public IDocumentHandler Next { get; set; }
    public int Order { get; }
}
```

In the implemented `Process` method, the `WillProcessDocument` will be called first. If this method returns false, the original document as the input parameter will be passed to the next handler's `Process` method if it exists. Otherwise, the `ProcessCore` method will be invoked and the returned document will be passed to the next handler's `Process` method if it exists. The finally returned document is the one generated by the last one of the document handler pipeline.

25.6.5. Convert File Format

LMS provides a concrete document handler, `ConvertFormatDocumentHandler`, to perform file format conversion. Because the underlying file format conversion is implemented by *Aspose.Word*, this document handler is available only we installed *Aspose.Word*. `ConvertFormatDocumentHandler` supports conversion between 4 formats, *Html/Htm*, *Doc*, *Docx* and *Pdf*.

The following code snippet represents the member definition of `ConvertFormatDocumentHandler`. Whether to perform file format conversion and the destination file format are determined by the data tokens of current document processing context.

```
[ConfigurationElementType(typeof(ConvertFormatDocumentHandlerData))]
public class ConvertFormatDocumentHandler : DocumentHandlerBase
{
    public ConvertFormatDocumentHandler(int order);
    protected override Document ProcessCore(Document document,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        DocumentProcessingContext context);
    protected override bool WillProcessDocument(DocumentProcessingContext context);
}
```

In detailed, `ConvertFormatDocumentHandler` depends on one built-in data token, whose name is "*TargetFileFormat*". It is worth emphasizing that the commonly used data token names are all defined as constants of the static class `DataTokenNames`. As its name suggests, this token indicates the format of the target file, and the value is an above mentioned `DocumentFormat`.

```
public static class DataTokenNames
{
    //...
    public const string TargetDocumentFormat = "TargetFileFormat";
}
```

For the `WillProcessDocument` method which determines whether to really process the document, if the original document's format is different from the target format, it returns true, otherwise return false. As with the `ProcessCore` method, it converts the original document to a new one of target format.

Because `ConvertFormatDocumentHandler` depends on the data token of document processing context, and job item's data token dictionary is the parent of context based one, we usually specify job item's data token in the way to extended job. The following is a demonstrative code snippet.

```
LmsManager.Current.CreateJobByFileName("c:\\template.docx")
    .InitializeJobItem(jobItem=>
    {
      jobItem.DataTokens.Set(DataTokenNames.TargetDocumentFormat,DocumentFormat.Html);
    });
```

25.6.6. Mail Merge

Mail merge, which fills the fields or placeholders in template using the provided data source, is achieved by a special category of document handlers. We define a separate interface `IMailMerge` for all document handlers which perform mail merge. As the following code snippet illustrates, `IDocumentHandler` is the base interface of `IMailMerge`, the method `Execute` binds the given data sources to the given template, and finally generate a complete document.

```
public interface IMailMerge : IDocumentHandler {
    Document Execute(Document template, IEnumerable<IDataSource>dataSources);
}
```

In LMS application programing interface, there is an abstract `MailMergeBase` class, which is used as the base class of all concrete mail merge document handlers. `MailMergeBase` directly inherits `DocumentHandlerBase`, and the abstract method `Execute` is called in the overridden `ProcessCore` method. The template document and data sources directly come from the current document processing context. As with the concrete mail merge document handler inheriting `MailMergeBase`, the mail merge functionality just needs to be implemented in the `Execute` method.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public abstract class MailMergeBase : DocumentHandlerBase
{
    public MailMergeBase(order);
    public abstract Document Execute(Document template,
        IEnumerable<IDataSource> dataSources);
    protected override Document ProcessCore(Document document,
        DocumentProcessingContext context);
}
```

Two kinds of built-in mail merge are supported in LMS. One adopts MS Word document based template, and the other supports HTML based template. We leverage *Aspose.Word* to perform MS Word document template based mail merge. In LMS application programing interface, the following class `MailMerge`, which is defined in the namespace `NCS.IConnect.Lms.Aspose`, represents the *Aspose.Word* based mail merge document handler.

```
[ConfigurationElementType(typeof(MailMergeData))]
public class MailMerge : MailMergeBase
{
    public MailMerge(int order);
    public override Document Execute(Document template,
        IEnumerable<IDataSource> dataSources);
    protected virtual void ExecuteCore(Document document, object dataSource);
    protected virtual void ExecuteWithRegionsCore(Document document,
        object dataSource);
    protected override bool WillProcessDocument(DocumentProcessingContext
        context); }
```

The content about how to add mail merge fields and how the fields are parsed by *Aspose.Word* is not covered in the developer's guide. The reader can directly refer to the *Aspose.Word* document.

Because *Aspose.Word* is a commercial product, LMS provides an alternative to enable to create HTML based template. The HTML parser used in this kind of mail merge comes from "*Html Agility Pack*", which is an open source hosted in www.codeplex.com. You can get complete information about this "*Html Agility Pack*" from http://htmlagilitypack.codeplex.com/. In LMS application programming interface, the class `MailMerge`, which is defined in the namespace `NCS.IConnect.Lms.Html`,

25.7. Delivering Documents

LMS hosts a service to deliver the processed documents. For a job with one or more job items, it will send the document delivery service a notification after its job items are all executed. The job will not directly send the documents to the service, but first store it and sent the document ID to the service. The service then uses the unique ID to load the generated document. LMS leverages a component named document store to store and load the documents.

25.7.1. Document Store

Document delivery service is a queued service of a `NetMsmqBinding` based endpoint. Considering MSMQ message size limitation, it is not good idea to use the content of the document as a part of the message payload. Instead, LMS adopts such an alternative solution: the document will be stored one place which is reachable for delivery service.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

In LMS application programming interface, the interface `IDocumentStore` represents the document store used to store and load documents. There are three groups of methods defined in this interface, the method `GetOutputStreamToWriteGeneratedDocument` and `GetOutputStreamToWriteTemplateDocument` return the stream to read the specified generated document and original template file. The method `GetInputStreamToReadGeneratedDocument` and `GetInputStreamToReadTemplateDocument` returns the stream to write the generated document and template file. `DeleteGeneratedDocument` method is called to delete the generated documents, and this method will be called after the document is successfully delivered.

```
public interface IDocumentStore
{
    Stream GetOutputStreamToWriteGeneratedDocument(string documentId);
    Stream GetOutputStreamToWriteTemplateDocument(string filePath);
    Stream GetInputStreamToReadGeneratedDocument(string documentId);
    Stream GetInputStreamToReadTemplateDocument(string filePath);
    void DeleteGeneratedDocument(string documentId);
}
```

There is only one concrete document store, `ShareFolderDocumentStore`, in LMS application programming interface. As its name suggests, this document store uses a share folder as the document storage. As the following code snippet illustrates, its two properties, DirectoryToLoadDocuments and DirectoryToSaveDocuments, separately represent the directory from which the original template files are loaded and the directory to which the generated documents are saved.

```
[ConfigurationElementType(typeof(ShareFolderDocumentStoreData))]
public class ShareFolderDocumentStore : IDocumentStore
{
    public ShareFolderDocumentStore(string directoryToLoadDocuments, string
        directoryToSaveDocuments);
    public void DeleteGeneratedDocument(string
        documentId);
    public Stream GetInputStreamToReadGeneratedDocument(string
        documentId);
    public Stream GetInputStreamToReadTemplateDocument(string
        filePath);
    public Stream GetOutputStreamToWriteGeneratedDocument(string
        documentId);
    public Stream GetOutputStreamToWriteTemplateDocument(string
        filePath);
    public string DirectoryToLoadDocuments { get; } public string
        DirectoryToSaveDocuments { get; }
}
```

The document store can be directly registered to the job. As illustrated in the following code snippet, the class `DocumentProcessingJob`'s property DocumentStore references a document store used to load the template files and store the generated documents.

```
public class DocumentProcessingJob : IExtensibleObject<DocumentProcessingJob>
{
    //...
    public IDocumentStore DocumentStore { get; set; }
}
```

Generally speaking, document store is globally registered in a configurable manner. The following configuration snippet represents the `ShareFolderDocumentStore` based configuration.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<configSections>
  <section name = "iConnect.lms"
           type = "NCS.IConnect.Lms.Configuration.LmsSettings,
                   NCS.IConnect.Lms.Core" />
</configSections>
<iConnect.lms defaultDocumentStore="ShareFolderDocumentStore">
  <documentStores>
    <add name = "ShareFolderDocumentStore"
         type = "NCS.IConnect.Lms.ShareFolderDocumentStore, NCS.IConnect.Lms.Core"
         directoryToLoadDocuments = "C:\Lms\Workspace\Templates"
         directoryToSaveDocuments = "C:\Lms\Workspace\Documents"/>
  </documentStores>
</iConnect.lms>
```

25.7.2. Document Delivery Service

Document delivery service is a general WCF service. The following code snippet represents the definition of the service type and its contract interface. There is only one-way operation method `Deliver` defined.

```
[ServiceContract(Namespace="http://www.ncs.com.sg/")]
public interface IDocumentDeliveryService
{
    [OperationContract(IsOneWay=true)]
    void Deliver(DocumentDeliveryInfo deliveryInfo);
}

[ServiceBehavior(AddressFilterMode=AddressFilterMode.Any)]
public class DeliveryFailureLogService : IDocumentDeliveryService
{
    public void Deliver(DocumentDeliveryInfo deliveryInfo);
}
```

The `DocumentDeliveryInfo` object, which is input parameter of the `Deliver` method, carries some delivery related information, including the delivery ID and delivery time. Its data tokens stored in the DataTokens property come from the job's data token dictionary, but the non-serializable tokens will be ignored.

```
[KnownType(typeof(MailMessage))]
[KnownType(typeof(DocumentFormat))]
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class DocumentDeliveryInfo
{
    public DocumentDeliveryInfo(DocumentProcessingJob job);
    public DocumentDeliveryInfo(string deliveryId, DateTime deliveryTime);
    [DataMember]
    public IDictionary<string, object> DataTokens { get; }
    [DataMember]
    public string DeliveryId { get; }
    [DataMember]
    public DateTime DeliveryTime { get; }
    [DataMember]
    public ICollection<DocumentDeliveryItemInfo> Items { get; }
}
```

DocumentDeliveryInfo is a collection of `DocumentDeliveryItemInfo`, which are stored in the property Items. Each `DocumentDeliveryItemInfo` is created based on a document process context. The property DocumentID represents the ID of the finally generated document. With the help of document store, the delivery service can load the corresponding document based on this ID. Another property DataTokens contains the serializable tokens which come from the document processing context.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class DocumentDeliveryItemInfo
{
    public DocumentDeliveryItemInfo(string documentId, DocumentProcessingContext
      context = null);
    [DataMember]
    public IDictionary<string, object> DataTokens { get; }
    [DataMember] public string DocumentId { get; }
}
```

25.7.3. Document Deliverer

Delivery notification is sent by document deliverer, which implements the following interface `IDocumentDeliverer`. There is only one method `Deliver`, which notifies the delivery service that the documents are already prepared. As we know, document delivery service is just a WCF service, so delivery notification just one service invocation. LMS provides the only document deliverer, `WcfDocumentDeliverer`.

```
public interface IDocumentDeliverer
{
    void Deliver(DocumentDeliveryInfo deliveryInfo);
}

public class WcfDocumentDeliverer : IDocumentDeliverer
{
    public WcfDocumentDeliverer(string endpointName);
    public void Deliver(DocumentDeliveryInfo deliveryInfo);
    public string EndpointName { get; }
}
```

The EndpointName represents configuration name of the client endpoint to invoke the document delivery service. Because the `WcfDocumentDeliverer` is by default used, we do not have to manually configure it. Instead we just need to configure the endpoint name like this:

```
<configSections>
  <section name="iConnect.lms"
           type="NCS.IConnect.Lms.Configuration.LmsSettings,
                 NCS.IConnect.Lms.Core"/>
</configSections>
<iConnect.lms deliveryEndpoint="lmsService" ...>
  ...
<iConnect.lms>
<system.serviceModel>
  <bindings>
    <netMsmqBinding>
      <binding exactlyOnce="true"/>
    </netMsmqBinding>
  </bindings>
  <client>
    <endpoint name="lmsService"
              Address = "net.msmq://localhost/private/lmsdemo"
              binding = "netMsmqBinding"
              contract = "NCS.IConnect.Lms.IDocumentDeliveryService"/>
  </client>
</system.serviceModel>
```

25.7.4. Document Delivery Listener

Document delivery service does not really deliver the documents. What are used for document delivery in LMS are document delivery listeners registered. All of the document delivery listeners implement the interface `IDocumentDeliveryListener`. As illustrated in the following

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

code snippet, there is only one method `Deliver` defined in this interface, which is used to deliver documents.

```
public interface IDocumentDeliveryListener
{
    void Deliver(IDocumentStore documentStore, DocumentDeliveryInfo deliveryInfo);
}
```

There are two parameters defined in the `Deliver` method. The 1st one represents the registered document store, which is used to load the document to deliver. The 2nd one *`deliveryInfo`* carries current document delivery related information.

By implementing this interface, the following abstract class `DocumentDeliveryListenerBase` is defined as the base class of all of the concrete delivery listener classes. Except for implementing the abstract `Deliver` method, there are other methods defined in `DocumentDeliveryListenerBase`. The method `WillDeliverDocumetns` determines whether the current listener needs to be used for the current document delivery. The listener will deliver the document only if this method returns true.

```
public abstract class DocumentDeliveryListenerBase : IDocumentDeliveryListener
{
    public void Deliver(IDocumentStore documentStore, DocumentDeliveryInfo deliveryInfo);
    protected abstract void DeliverCore(IDocumentStore documentStore,
        DocumentDeliveryInfo deliveryInfo);
    protected virtual bool HasDelivered(string deliveryId);
    protected abstract bool WillDeliverDocuments( DocumentDeliveryInfo deliveryInfo);
    protected string Identifier { get; } }
```

For a particular document delivery request, maybe more than delivery listeners are used. E.G. The mailing based delivery listener first sent the documents to the predefined receipt, and the upload listener then upload the documents to the pre-configured file server. LMS treats the whole delivery involving in more than one delivery listeners as a “transaction” and the auto-retry mechanism make the whole delivery reoccur once exception happens.

However, some listeners have no rollback capability once completing delivery. In order to prevent the listener to re-deliver the documents when delivery retries, we give a listener a unique ID and save a flag into database after it completing delivery to indicates it has successfully finished the current delivery. `DocumentDeliveryListenerBase`’s property Identifier represents such a unique ID. The method `HasDelivered` returns a `Boolean` value indicating whether it has successfully completed the given delivery. Real document delivery defined in the `DeliverCore` method will happens only the method `WillDeliverDocuments` and `HasDelivered` methods separately return true and false.

At the delivery service side, both document store and document delivery listeners can be directly registered to the current `LmsManager`, which is referenced by the static property Current of LmsManager. `LmsManager`’s properties DocumentStore and Listeners represent the registered document store and document delivery listeners. There is an extra property DeliveryExceptionPolicy, which represents the configuration name of the exception policy to handle the exception thrown during performing document delivery. If this property is not

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

explicitly specified, the error message will be directly written to event log and the log entry source is "*Application*".

```
public class LmsManager
{
    //...
    public static LmsManager Current { get; }
    public string DeliveryExceptionPolicy { get; set; }
    public IDocumentStore DocumentStore { get; set; }
    public DocumentDeliveryListenerCollection Listeners { get; }
}
```

Generally speaking, the exception policy and document delivery listeners are registered in a configurable manner. The following code snippet illustrates the configuration's XML structure.

```
<configSections>
  <section name="iConnect.lms"
           type="NCS.IConnect.Lms.Configuration.LmsSettings,
                 NCS.IConnect.Lms.Core"/>
</configSections>
<iConnect.lms deliveryExceptionPolicy = "policy name">
  <deliveryListeners>
    <add name="listener 1"
         type="Listener Type 1" .../>
    <add name="listener 2"
         type="Listener Type 2" .../>
  </deliveryListeners>
</iConnect.lms>
```

It is worth emphasizing that only one document delivery listener of a particular type can be registered. In other words, the registered listener can be identified by its type. In LMS, the listener's identifier is determined by the `DeliveryListenerTypeIdAttribute` attribute applied to the listener type. The member definition of `DeliveryListenerTypeIdAttribute` is illustrated in the following code snippet. The property ListenerTypeId represents document delivery listener type' identifier, which is also the identifier of the registered listener instance.

```
[AttributeUsage(AttributeTargets.Class)]
public class DeliveryListenerTypeIdAttribute : Attribute
{
  public DeliveryListenerTypeIdAttribute(string listenerTypeId);
  public string ListenerTypeId { get; }
}
```

25.7.5. SmtpDocumentDeliveryListener

As it class name suggests, `SmtpDocumentDeliveryListener` is used deliver the document by sending a mail to the predefined receipts. The documents to deliver can be sent as attachments, or directly used as the mail body. The following code snippet represents the member definition of `SmtpDocumentDeliveryListener`.

```
[DeliveryListenerTypeId("{6D1CFEDF-2CD5-4D4A-9A76-45E806F90C92}")]
[ConfigurationElementType(typeof(SmtpDocumentDeliveryListenerData))]
public class SmtpDocumentDeliveryListener : DocumentDeliveryListenerBase
{
    public SmtpDocumentDeliveryListener( KeyValueConfigurationCollection headers = null);
    protected virtual MailMessage CreateMessage(MailMessage mailMessage);
    protected override void DeliverCore(IDocumentStore documentStore, DocumentDeliveryInfo
deliveryInfo);
    protected override bool WillDeliverDocuments( DocumentDeliveryInfo deliveryInfo);
    public Encoding BodyEncoding { get; set; }
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public DeliveryNotificationOptions DeliveryNotificationOptions { get; set; }
    public bool EnableSsl { get; set; }
    public NameValueCollection Headers { get; }
    public Encoding HeadersEncoding { get; set; }
    public bool IsBodyHtml { get; set; }
    public Encoding SubjectEncoding { get; set; }
}
```

The method `WillDeliverDocuments` determines whether to perform the current delivery based on a predefined data token, *DeliverDocumentsViaEmail*. If such a data token exists in the `DeliveryInfo` and the value is *true* or *1*, this method return *true*.

Besides the predefined data token *DeliverDocumentsViaEmail*, there are other mailing related ones. The following code snippet illustrates the corresponding constant definition in the class `DataTokenNames`.

```
public static class DataTokenNames
{
    public const string DeliverDocumentsViaEmail = "DeliverDocumentsViaEmail";
    public const string IsCoverLetterDocument = "IsCoverLetterDocument";
    public const string MailMessage = "MailMessage";
    public const string TargetDocumentFormat = "TargetFileFormat";
    public const string TargetFileName = "TargetFileName";
    public const string UseDocumentAsMailBody = "UseDocumentAsMailBody";
}
```

Before running a document processing job, the mail message which mainly carries mail address related information must be placed in the job's data token dictionary, and the token name is "*MailMessage*". The value of such a token is a `MailMessage` object.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class MailMessage
{
    public MailMessage(string from, string subject);
    [DataMember]
    public ICollection<Attachment> Attachments  { get; }
    [DataMember]
    public ICollection<string> Bcc { get; }
    [DataMember]
    public string Body { get; set; }
    [DataMember]
    public ICollection<string> CC { get; }
    [DataMember]
    public string CoverLetterTemplateName { get; set; }
    [DataMember]
    public string From { get; }
    [DataMember]
    public ICollection<ImageAttachment> Images { get; }
    [DataMember]
    public string MailMessageName { get; set; }
    [DataMember]
    public MailPriority Priority { get; set; }
    [DataMember]
    public ICollection<string> ReplyTo { get; }
    [DataMember]
    public string Subject { get; set; }
    [DataMember]
    public ICollection<string> To { get; }
}
```

The `CreateMessage` of the `SmtpDocumentDeliveryListener` will convert such a `MailMessage` to a `System.Net.Mail.MailMessage` object, which will be finally used as the mail message to send. There are some mail message related properties, including BodyEncoding,

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

DeliveryNotificationOptions, IsBodyHtml and SubjectEncoding etc, which all correspond to `System.Net.Mail.MailMessage`'s properties of the same name.

The content of the document to deliver can be directly used as the mail body. This is determined by another predefined token, *UseDocumentAsMailBody*. If such a token does not exist or its value is false or 0, the documents will be sent as the attachments of mail.

In the situation with more than one job items for the job to run, there must be one and only one cover letter based job item. Like general documents, there is no special processing mechanism applied to the cover letter based job item. LMS determines the predefined token *IsCoverLetterDocument* to determine if a given job item is a cover letter based one. As with `SmtpDocumentDeliveryListener`, it loads the cover letter document based on this token and use its content as the mail body. Other documents will be used as the attachment of the mail to send. The attachment's name is determined by another two tokens, *TargetFileFormat* and *TargetFileName*, which separately represent the format and name of the target file. If neither of the two tokens does not exist, a string generated based on the current timestamps will become its name.

`SmtpDocumentDeliveryListener` can be registered in a configurable manner as below. From this configuration snippet, we can see all of the properties defined in `SmtpDocumentDeliveryListener` can be specified by the corresponding configuration attributes.

```
<configSections>
  <section name="iConnect.lms"
           type="NCS.IConnect.Lms.Configuration.LmsSettings,
                 NCS.IConnect.Lms.Core"/>
</configSections>
<iConnect.lms ...>
  <deliveryListeners>
    <add name=" SmtpDocumentDeliveryListener "
         type="NCS.IConnect.Lms.SmtpDocumentDeliveryListener,
               NCS.IConnect.Lms.Extensions"
         deliveryNotificationOptions ="OnFailure"
         isBodyHtml = "true"
         bodyEncoding = "utf-8"
         headersEncoding = "utf-8"
         subjectEncoding = "utf-8">
    </add>
  </deliveryListeners>
</iConnect.lms>
```

25.7.6. FileSaverDeliveryListener

The `FileSaverDeliveryListener` is used to save the documents to a predefined directory. The following code snippet illustrates the member definition of `FileSaverDeliveryListener`. The property Directory represents the directory to which the documents will be saved.

```
[DeliveryListenerTypeId("{CEA7D3C1-2126-4F24-8F36-53FC7F6C6290}")]
[ConfigurationElementType(typeof(FileSaverDeliveryListenerData))]
public class FileSaverDeliveryListener : DocumentDeliveryListenerBase
{
  public FileSaverDeliveryListener(string directory);
  protected override void DeliverCore(IDocumentStore documentStore,
      DocumentDeliveryInfo deliveryInfo);
  protected virtual string GetFileName(string targetFileName);
  protected override bool WillDeliverDocuments(
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
      DocumentDeliveryInfo deliveryInfo);
  public DirectoryInfo Directory { get; }
}
```

Similar to attachment naming for `SmtpDocumentDeliveryListener`, the name of the file to save is determined by the two predefined data tokens, *TargetFileFormat* and *TargetFileName*. File name calculation is implemented in the virtual method `GetFileName`. If the file of the calculated file name exists, the file name will be appended a suffix like “(1)”, “(2)”, etc.

The following code snippet represents the configuration of `FileSaverDeliveryListener`. Only the directory which the documents are saved to needs to be configured.

```
<configSections>
  <section name="iConnect.lms"
      type="NCS.IConnect.Lms.Configuration.LmsSettings,
            NCS.IConnect.Lms.Core"/>
</configSections>
<iConnect.lms ...>
  <deliveryListeners>
    <add name=" SmtpDocumentDeliveryListener "
        type = "NCS.IConnect.Lms.FileSaverDeliveryListener,
                NCS.IConnect.Lms.Extensions"
        directory = "c:\lms\documents\"
    </add>
  </deliveryListeners>
</iConnect.lms>
```

25.7.7. Delivery Retry

The mechanism of retry is applied to document delivery. Another document delivery will automatically be started up if the previous one fails. The retry cycle count, retry time per retry cycle, and the delay time can be configured.

As above mentioned, the document delivery service is a queued service with a `NetMsmqBinding` endpoint. The retry mechanism of WCF queued service is directly borrowed. Because queued service's retry depends on transactional message queue, please make sure the queue used by the `NetMsmqBinding` of document delivery service's endpoint is transactional. As with retry based configuration, including the count of retry cycles, retry count per cycle, and retry delay time, can refer to the following demonstrative configuration. For detailed about queued service's retry, please refer to WCF related documentations.

```
<system.serviceModel>
  <bindings>
    <netMsmqBinding>
      <binding exactlyOnce = "true"
               maxRetryCycles = "2"
               receiveRetryCount = "2"
               retryCycleDelay = "00:00:05">
       </binding>
     </netMsmqBinding>
  </bindings>
  <services>
    <service name= "NCS.IConnect.Lms.DocumentDeliveryService">
      <endpoint address = "net.msmq://localhost/private/lmsdemo"
                binding = "netMsmqBinding"
                contract = "NCS.IConnect.Lms.IDocumentDeliveryService"/>
    </service>
  </services>
</system.serviceModel>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

25.7.8. Delivery Failure Handling

Once exceeding the maximum retry cycles, the current delivery will be considered as a failed one. We leverage the error handling strategy "*Move*" of queued service to log the delivery information when delivery fails. For such an error handling policy, the message will be move from the endpoint's queue to the "*poison*" queue.

We define another service, `DeliveryFailureLogService`, which is hosted to target to the poison queue and log the delivery information into database. The following code snippet represents the definition of the `DeliveryFailureLogService` and its interface.

```
[ServiceContract(Namespace="http://www.ncs.com.sg/")]
public interface IDocumentDeliveryService
{
    [OperationContract(IsOneWay=true)]
    void Deliver(DocumentDeliveryInfo deliveryInfo);
}

[ServiceBehavior(AddressFilterMode = AddressFilterMode.Any)]
public class DeliveryFailureLogService :IDocumentDeliveryService
{
    public void Deliver(DocumentDeliveryInfo deliveryInfo);
}
```

The `DeliveryFailureLogService` is hosted to target at the poison queue. The configuration is like this, and please take notes the endpoint's address "*net.msmq://localhost/private/lmsdemo;poison*".

```
<system.serviceModel>
  <services>
    <service name="NCS.IConnect.Lms.DocumentDeliveryService">
      <endpoint address="net.msmq://localhost/private/lmsdemo"
                binding="netMsmqBinding"
                contract="NCS.IConnect.Lms.IDocumentDeliveryService"/>
    </service>
    <service name="NCS.IConnect.Lms.DeliveryFailureLogService">
      <endpoint address="net.msmq://localhost/private/lmsdemo;poison"
                binding="netMsmqBinding"
                contract="NCS.IConnect.Lms.IDocumentDeliveryService"/>
    </service>
  </services>
</system.serviceModel>
```

25.8. Document Processing Job Provider

The document processing job is the basic executable unit, which drive the whole document processing flow. A job can be regarded as an orchestrated work flow to process documents. Its state before it is executed determines how the documents are processed. We can manually create job and configure it by attaching some extensions in the above mentioned manner. We can also persist a job with its state into database.

In LMS, persistently storing the job and loading job from database are implemented by job provider. Besides the two basic functionalities, another application programming interface to maintain template, mail message, and data source is defined in job provider. All of the job providers implement the interface `IDocumentProcessingJobProvider`.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

25.8.1. Job Provider is an Extension

According to the definition, we can know job provider itself is a job initializer, job item initializer and document context initializer. It loads the related information from persistent storage to perform some initialization.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
{
    //...
}
```

We define an extra abstract class, `DocumentProcessingJobProviderBase`, which is used as the base class of all concrete job providers. As illustrated in the following code snippet, it directly inherits `DocumentProcessingJobExtensionBase`, so a job provider is also a job extension. Once we attaching a job provider to a job, the initialization operation will be performed when the job is initialized. As the matter of fact, the configured default job provider will be automatically attached to the job created using current `LmsManager`.

```
public abstract class DocumentProcessingJobProviderBase
    : DocumentProcessingJobExtensionBase,
      IDocumentProcessingJobProvider
{
    //...
}
```

25.8.2. Job Maintenance

We can leverage the application programming interface defined in the job provider to maintain the job. The `CreateJob` and `DeleteJob` method can be used to create a new job or delete an existing job. The `SetMailMessage` method is invoked to set a predefined a mail message if the document needs to be delivered by sending a mail. The cover letter can be specified to a job by calling the `SetCoverLetterTemplate`. The methods `SetJobDataTokens` and `RemoveJobDataTokens` are used to maintain a job's data tokens.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
    {
        //...
        void CreateJob(string jobName, string description, string mailMessageName,
            string coverLetterTemplateName);
        void DeleteJob(string jobName);

        void SetMailMessage(string jobName, string mailMessageName);
        void SetCoverLetterTemplate(string jobName,
            string coverLetterTemplateName);

        void SetJobDataTokens(string jobName,
            IDictionary<string, string> dataTokens);
        void RemoveJobDataTokens(string jobName, string[] tokenNames);
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

We can directly call some methods defined in job provider to get job related information. `FindJobs` method returns a collection of `DocumentProcessingJobInfo`, which carries the basic job information without is job items, based on the specified search criteria. Pagination and sorting are supported. Another method `GetJob` returns a `DocumentProcessingJobInfo` based on the specified job name. It returns null if the job does not exists.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
    {
        IEnumerable<DocumentProcessingJobInfo> FindJobs(string jobNameToMatch,
            int pageSize, int pageIndex, SortDirection sortDirection);
        DocumentProcessingJobInfo GetJob(string jobName);
}

[DataContract(Namespace="http://www.ncs.com.sg/")]
public class DocumentProcessingJobInfo
{
    [DataMember]
    public string CoverLetterTemplateName { get; set; }
    [DataMember]
    public IDictionary<string, string> DataTokens { get; }
    [DataMember]
    public string Description { get; set; }
    [DataMember]
    public string JobName { get; set; }
    [DataMember]
    public string MailMessageName { get; set; }
}
```

25.8.3. Job Item Maintenance

After creating an empty job, we can add a job item to it by calling the `AddJobItem` method. A job item is usually mapped to a predefined a template, the method `ChangeTemplate` can be called to change the given job item's template. The job item can be removed from its job by calling the `RemoveJobItem`. We set or remove job item's data tokens by calling the methods `SetJobItemDataTokens` and `RemoveJobItemDataTokens`. In LMS application programming interface, the persistent information about a job item is represented by the class `DocumentProcessingJobItemInfo`. We can get all job items of a given job by calling the `GetAllJobItems` method.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
    {
        //...
        void AddJobItem(string jobName, string jobItemName, string description,
            string templateName);
        void ChangeTemplate(string jobName, string jobItemName,
            string templateName);
        void RemoveJobItem(string jobName, string jobItemName);
        void SetJobItemDataTokens(string jobName, string jobItemName,
            IDictionary<string, string> dataTokens);

        void RemoveJobItemDataTokens(string jobName, string jobItemName,
            string[] tokenNames);
        IEnumerable<DocumentProcessingJobItemInfo> GetAllJobItems(string jobName);
   }

[DataContract(Namespace="http://www.ncs.com.sg/")]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public class DocumentProcessingJobItemInfo
{
    [DataMember]
    public IDictionary<string, string> DataTokens { get; set; }
    [DataMember]
    public string Description { get; set; }
    [DataMember]
    public string JobItemName { get; set; }
    [DataMember]
    public string TemplateName { get; set; }
}
```

25.8.4. Template Maintenance

Considering template effectiveness, which means different template file should be used in different period, the visioning is applied to LMS. A template can be considered as a collection of template version, which is mapped to a template file and is only effective during the specified period. In LMS application programming interface, the template and template version are separately represented by the class `TemplateInfo` and `TemplateVersionInfo`.

```
[DataContract(Namespace="http://www.ncs.com.sg/")]
public class TemplateInfo
{
    [DataMember]
    public string Description { get; set; }
    [DataMember]
    public string TemplateName { get; set; }
}

[DataContract(Namespace="http://www.ncs.com.sg/")]
public class TemplateVersionInfo
{
    [DataMember]
    public DateTime EffectiveStartDate { get; set; }
    [DataMember]
    public string TemplateFilePath { get; set; }
    [DataMember]
    public string TemplateName { get; set; }
    [DataMember]
    public string TemplateVersionNo { get; set; }
}
```

We can call the method `CreateTemplate` to create a new template. An existing template can be deleted by calling the method `DeleteTemplate`. The method `AddTemplateVersion` is used to add a new version to an existing template. For an existing application version, its template file path and effective start date can be changed by separately calling the method `ChangeFilePath` and `ChangeEffectiveDate`. We can remove a version from the specified template by calling the `RemoveTemplateVersion` method.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
{
        //...
        void CreateTemplate(string templateName, string description);
        void DeleteTemplate(string templateName);

        void AddTemplateVersion(string templateName, string templateVersionNo,
            string filePath, DateTime effectiveStartDate);
        void ChangeFilePath(string templateName, string TemplateVersionNo,
            string filePath);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        void ChangeEffectiveDate(string templateName, string templateVersionNo,
            DateTime effectiveStartDate);
        void RemoveTemplateVersion(string templateName, string templateVersionNo);

        IEnumerable<TemplateInfo> FindTemplates(string templateNameToMatch,
            int pageSize, int pageIndex, SortDirection sortDirection);
        TemplateInfo GetTemplate(string templateName);
        IEnumerable<TemplateVersionInfo> GetAllVersions(string templateName);
        TemplateVersionInfo GetEffectiveVersion(string templateName);
}
```

The `FindTemplates` retrieves the templates which match the specified search criteria. Both pagination and sorting are supported. If we need to get a particular template by its name, we can call the method `GetTemplate`. The `GetAllVersions` method can be chosen if we need to get all versions of the given template. The method `GetEffectiveVersion` get the effective template version by the specified template name. Effectiveness calculation is based on the current date time.

25.8.5. Data Source Maintenance

Stored procedure based data source is a special mail merge data source, which gets the data to fill the mail merge fields by executing the specified stored procedure. As above mentioned, a stored procedure based data source is represented by the class `ProcedureDataSource`. We can use related application programming interface defined in job provider to maintain stored procedure data source.

As illustrated in the following code snippet, the method `CreateDataSource` is used to create a new data source. The stored procedure name and the names of the data tables loaded into `DataSet` can be changed by separately calling the method `ChangeProcedureName` and `ChangeTableNames`. We can call the `SetRegional` to determine an existing data source is regional or scalar. In the `DataSet` loaded by executing the predefined stored procedure, there is usually a relationship between two data tables. Such a relationship can be maintained by calling the method `AddRelationToDataSource` and `RemoveRelationFromDataSource`. An obsoleted data source can be deleted by calling the method `DeleteDataSource`.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
{
    //...
    void CreateDataSource(string dataSourceName, string procedureName,
         bool isRegional, string[] tableNames, DataSourceRelation[] relations);
    void ChangeProcedureName(string dataSourceName, string procedureName);
    void ChangeTableNames(string dataSourceName, string[] tableNames);
    void SetRegional(string dataSourceName, bool isRegional);

    void AddRelationToDataSource(string dataSourceName, string parentTable,
         string childTable, string parentColumn, string childColumn);
    void RemoveRelationsFromDataSource(string dataSourceName);
    void DeleteDataSource(string dataSourceName);
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

The following method `FindDataSources` retrieves the data sources which match the specified search criteria. Pagination and sorting are both supported. Another method `GetDataSource` return the data source by specified data source name.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
{
    //...
    IEnumerable<DataSourceInfo> FindDataSources(string dataSourceNameToMatch,
        int pageSize, int pageIndex, DataSourceSortField orderBy,
        SortDirection sortDirection);
    DataSourceInfo GetDataSource(string dataSourceName);
}
```

A template is usually mapped to one or more data sources. We can use the following methods to build or remove such a mapping between template and data source. We can get all of the data sources which are added to the specified template by calling the method `GetAllDataSources`.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
{
    //...
    void AddDataSourceToTemplate(string dataSourceName, string templateName,
        string templateVersionNo);
    void RemoveDataSourceFromTemplate(string dataSourceName,
        string templateName, string templateVersionNo);
    IEnumerable<DataSourceInfo> GetAllDataSources(string templateName,
        string templateVersionNo);
}
```

25.8.6. Mail Message Maintenance

As above mentioned, mail message is represented by the class `MailMessage`, which mainly contains the mail addresses and subject of a mail to send. We can use the following methods to maintain mail messages.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
{
    //...
    void CreateMailMessage(string mailMessageName, string subject, string from,
        string to, string cc, string bcc, string replyTo);
    void UpdateMailMessage(string mailMessageName, string subject, string from,
        string to, string cc, string bcc, string replyTo);
    void DeleteMailMessage(string mailMessageName);

    IEnumerable<MailMessage> FindMailMessages(string mailMessageNameToMatch,
       int pageSize, int pageIndex, MailMessageSortFiled orderBy,
       SortDirection sortDirection);
    MailMessage GetMailMessage(string mailMessageName);
}
```

In most situations, there are some inline images each of which uniquely identified by a content ID in the mail body. We call these images *image attachment*. The following methods are

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

defined in job provider to make an existing mail message can own one or more image attachments.

```
public interface IDocumentProcessingJobProvider :
    IDocumentProcessingJobInitializer,
    IDocumentProcessingJobItemInitializer,
    IDocumentProcessingContextInitializer
    {
        //…
        void AddImageAttachment(string mailMessageName, string contentId,
            string imagePath);
        void RemoveImageAttachment(string mailMessageName, string contentId);
        void RemoveAllImageAttachments(string mailMessageName);
    }
```

25.8.7. DbDocumentProcessingJobProvider and WcfDocumentProcessingJobProvider

There are two built-in concrete job provider types, `DbDocumentProcessingJobProvider` and `WcfDocumentProcessingJobProvider`. As their names suggest, `DbDocumentProcessingJobProvider` persists the job related information into database, and job persistence for `WcfDocumentProcessingJobProvider` is achieved by remotely call a service. The type of the `WcfDocumentProcessingJobProvider` specific service is `DocumentProcessingJobProviderService`, which implements the service contract interface `IDocumentProcessingJobProviderService`.

The following code snippet demonstrates how to configure the two job providers. The job provider which is by default used is determined the configuration property "*defaultProvider*" of LMS configuration section.

```
<configSections>
  <section name="iConnect.lms"
           type="NCS.IConnect.Lms.Configuration.LmsSettings,
                 NCS.IConnect.Lms.Core" />
</configSections>
<system.serviceModel>
  <client>
    <endpoint name = "jobproviderservice"
              address = "http://localhost/jobproviderservice"
              binding = "ws2007HttpBinding"
              contract = "NCS.IConnect.Lms.IDocumentProcessingJobProviderService"/>
  </client>
</system.serviceModel>

<connectionStrings>
  <add name="iConnectDb"
       providerName="System.Data.SqlClient"
       connectionString="…"/>
</connectionStrings>

<iConnect.lms defaultProvider="dbProvider"
  <providers>
    <add name="dbProvider"
          type="NCS.IConnect.Lms.DbDocumentProcessingJobProvider,
                NCS.IConnect.Lms.Extensions"
          applicationName="LmsDemo"
          database="iConnectDb"/>
    <add name="wcfProvider"
          type="NCS.IConnect.Lms.WcfDocumentProcessingJobProvider,
                NCS.IConnect.Lms.Extensions"
          endpoint="jobproviderservice"/>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
  </providers>
</iConnect.lms>
```

25.8.8. Run a Persisted Document Processing Job

In the getting-started sample, we manually create an empty document processing job and attach related job items to send a order receival notification mail with an attachment. If we persist the job, with its job items and related template into database, we just need to create a job based on the job name and directly run it. The following code snippet demonstrates how to persist the job and template and create and run a named job. This code snippet has the same effect with the one defined in the getting-started sample.

```
string coverLetterTemplate = "CoverLetter";
string attachmentTemplate = "OrderList";
string jobName = "SendOrderNotificationMail";

if (LmsManager.Current.JobProvider.GetTemplate(coverLetterTemplate) == null)
{
    LmsManager.Current.JobProvider.CreateTemplate(coverLetterTemplate, null);
   LmsManager.Current.JobProvider.AddTemplateVersion(coverLetterTemplate, "2013", "CoverLetter.docx",
DateTime.Today);
}
if (LmsManager.Current.JobProvider.GetTemplate(attachmentTemplate) == null)
{
    LmsManager.Current.JobProvider.CreateTemplate(attachmentTemplate, null);
   LmsManager.Current.JobProvider.AddTemplateVersion(attachmentTemplate, "2013", "CoverLetter.docx",
DateTime.Today);
}

if (LmsManager.Current.JobProvider.GetJob(jobName) == null)
{
    LmsManager.Current.JobProvider.CreateJob(jobName, null, null, coverLetterTemplate);
    Dictionary<string, string> tokens = new Dictionary<string, string>();
    tokens.Add(DataTokenNames.DeliverDocumentsViaEmail, true.ToString());
    LmsManager.Current.JobProvider.SetJobDataTokens(jobName, tokens);

   LmsManager.Current.JobProvider.AddJobItem(jobName, "ProcessOrderList", null, attachmentTemplate);
    tokens.Clear();
    tokens.Add(DataTokenNames.TargetDocumentFormat, DocumentFormat.Pdf.Extension);
    tokens.Add(DataTokenNames.TargetFileName, "OrderList");
    LmsManager.Current.JobProvider.SetJobItemDataTokens(jobName, "ProcessOrderList", tokens);
}

DocumentProcessingJob job = LmsManager.Current.CreateJobByName(jobName);
MailMessage mailMessage = new MailMessage("...", "Submited order is received");
mailMessage.To.Add("...");
job.DataTokens.Set(DataTokenNames.MailMessage, mailMessage);
job.Run(CreateScalarDataSources(), CreateRegionalDataSources());
```

In the above program, we just persist the job with it job items and the two templates. The MailMessage is dynamically specified because it is determined by the person whose submit the order. The data source providing order data is a manually created DataSet. If the stored procedure data source is used, it can also be persisted and mapped to the template.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

25.9. Template Editor

25.9.1. Word Template Editor

The word template editor is a MS Word add-in. It is used to define the mail merge template. If the mail merge data sources is from database, we can directly load the data source from the database and select and insert the corresponding mail merge filed into the Word document. In addition, it also allows us to manually add the mail merge fields.

Note

You can create the template without this editor. But this editor can make the process of creating template easier and make less mistake when you enter the mail merge field.

25.9.1.1. Getting Started

You can get the word template editor installer from iConnect package. And install this tool follow the below step:

Step I: Install the certificate under the "Lms Editor" folder.

1. Double click the "LmsRootCertificate.cer"
2. Click the "Install Certificate" button
3. Choose "Place all certificates in the following store" in the wizard window
4. Click "Browse"
5. Choose "Trusted Root Certification Authorities" Item and click "OK"
6. Click "Next"
7. Click "Finish"

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

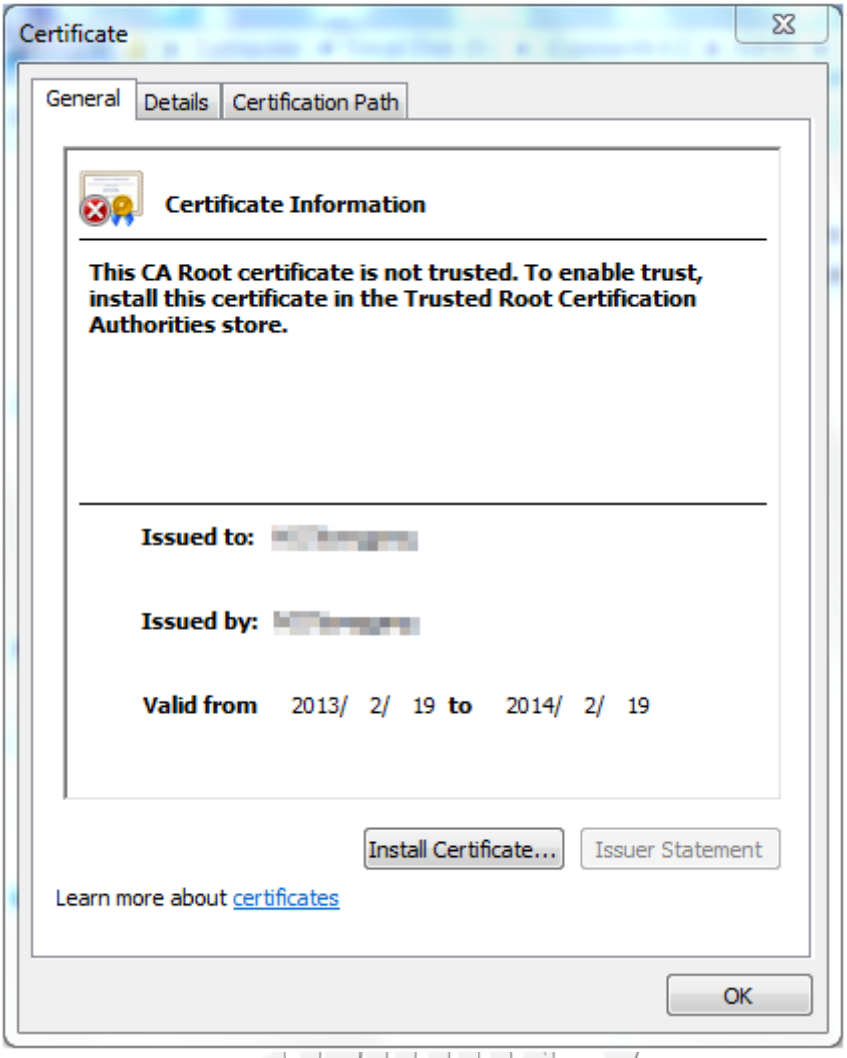


Figure 25.7. Word add-in certification

Repeat the above steps and choose the “Trusted publishers” in the 5th step.

Step II: Un-zip the LmsAddIn.zip under the "Lms Editor" folder.

Step III: Install the “Lms Editor” by click the "setup.exe"

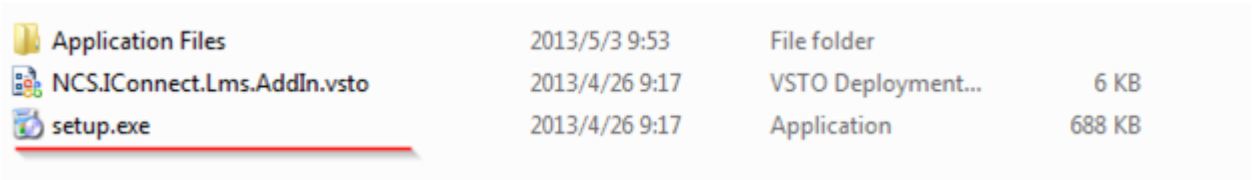


Figure 25.8. Lms word add-in setup

25.9.1.2. Components

The word template editor include two components.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

As illustrated in the following screen shot the first one is a new ribbon group called "Lms". There are two ribbon items "Import From Database" and "Custom Data Source". "Import From Database" is used to import the mail merge data source structure from the database. "Import From Database" is used to create the mail merge data source manually.

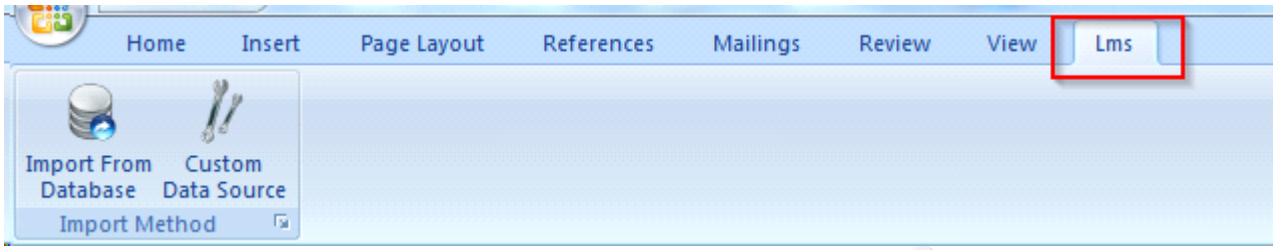


Figure 25.9. Template editor ribon

Note

It is not mandatory to use the "Import From Database" function to create the template when the data source is from database. You can also create the template by the "Custom Data Source" function.

The second component is a task pane as illustrated in the following screen shot which default located at the right side of the word. It is hide before you import or build the data source. It is used to display your imported or manually builded data source.

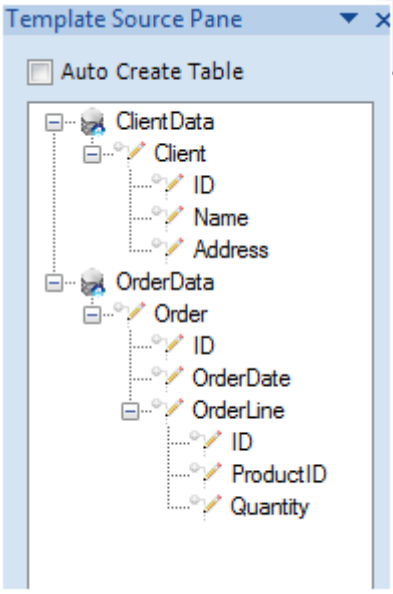


Figure 25.10. Template source pane

25.9.1.3. Create Template

1. How to create template with data source from database

   Step I:Open Word

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step II: Select template Data Source

a. Click ribbon item"Lms"->"Import From database"

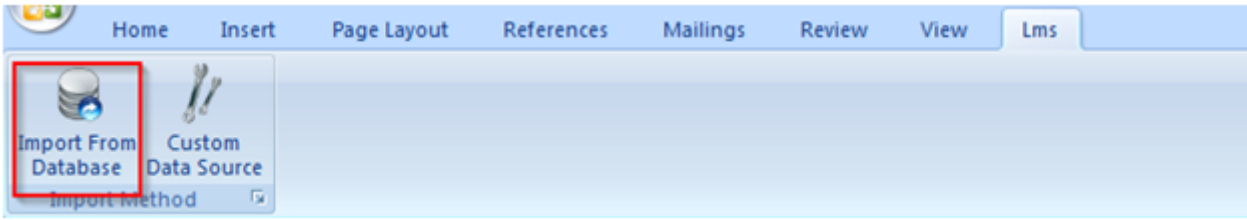


Figure 25.11. Import from database

b. Select the data source and data provider of your database. E.g.The data source exists at a sql database. Choose "Microsoft sql server" in data source list and select ".Net Framework Data Provider for SQL Server"

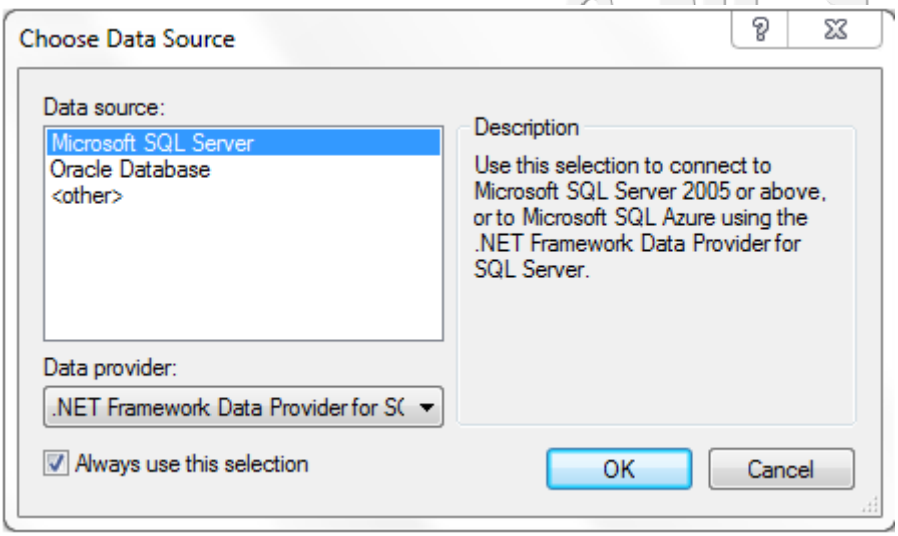


Figure 25.12. Choose data source

c. Connect to database .Enter the database connection string and click "OK" button

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

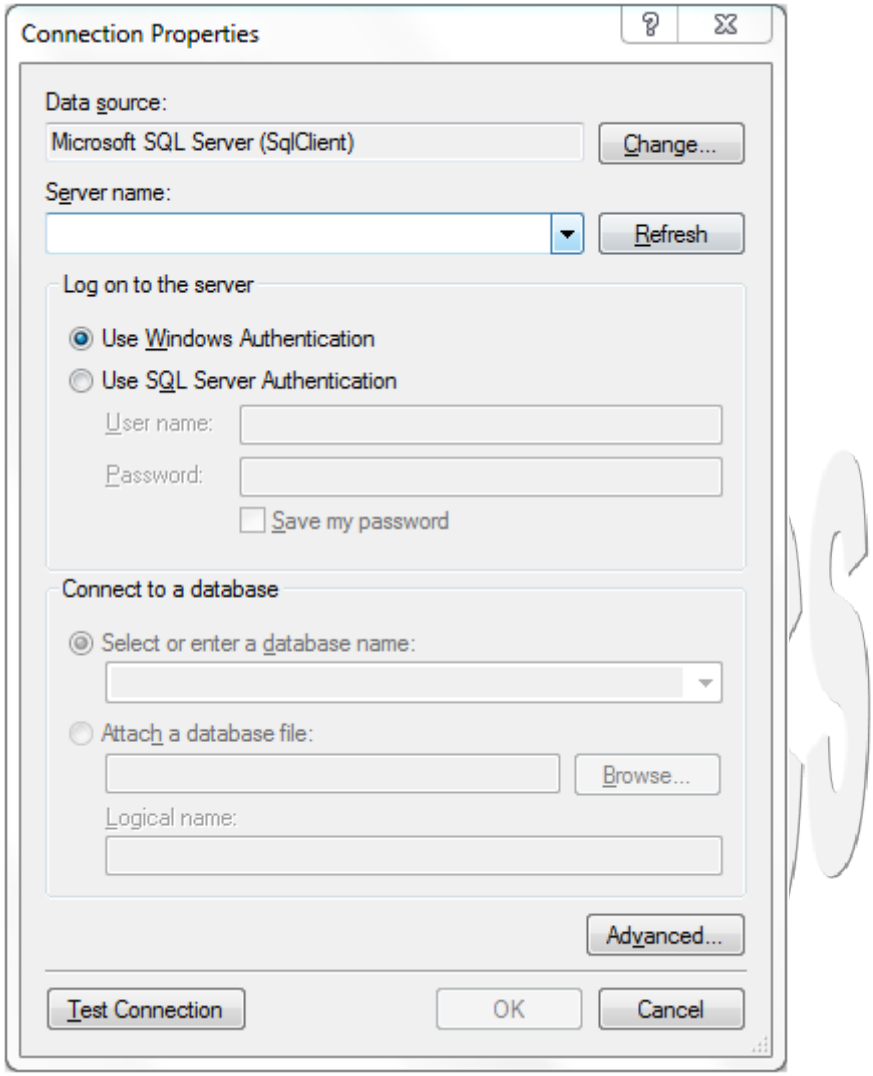


Figure 25.13. Database connection dialog

Note

The database connection is not equal the runtime connection which connected in the application, the connection which connected here is just for creating the template.

d. Select the template data source. Select the application and choose the related data source in the grid. If the structure of your data source will be different based on the passed data source parameter. You can use the "…" button beside the data source name. It is used to modify the default parameter(`DBNull`). Once you select the data source , the task pane will be shown. And the structure of the selected data source will be listed in the task pane.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

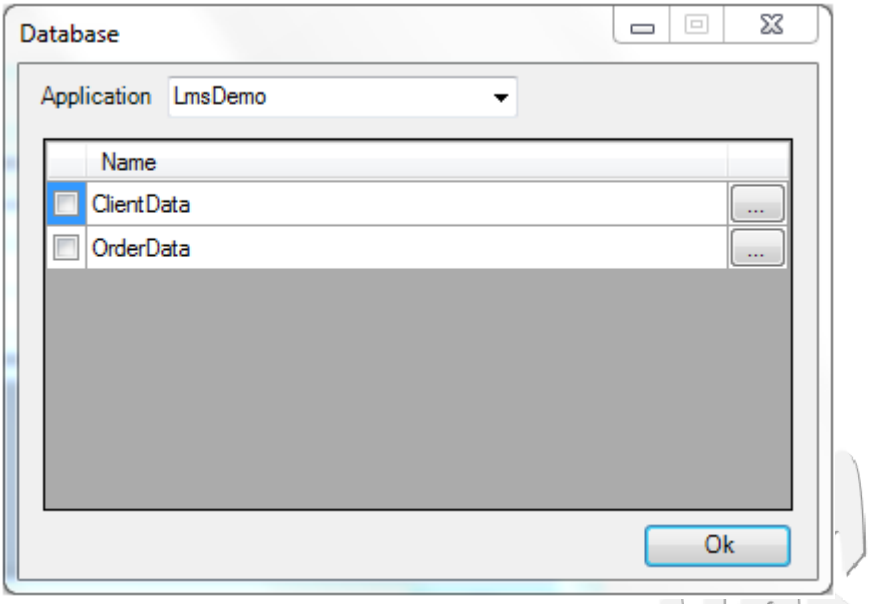


Figure 25.14. Data source list

Step III:Create Template. Write the document and drag drop the mail merge fields to the required place. If you drag an item which don't has child nodes, the item will be treated as a field and display like "<<Filed>>". If you drag an item which has some child nodes it will be treated as a table and display like" <<tablestart:table>> <<field>> <<tableend:table>>". At the top of this task pane there is a check box called "Auto Create Table". If it is checked and you drag drop the table item it will help you to create a grid.

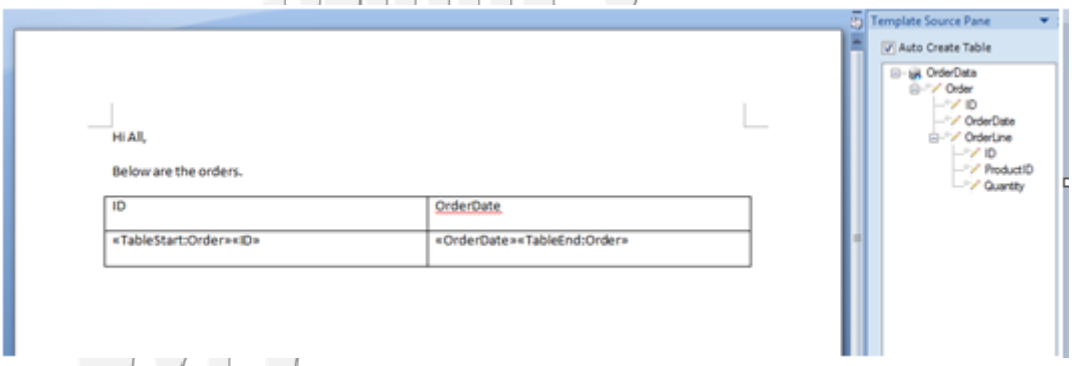


Figure 25.15. Template

Note

Aspose and html mail merge handler support the "DataTable","DataSet","DataView","DataRow","Dictionary" as the data source. But the data source can come from anywhere like database,memory and so on. If the data source is from database you can use this function to create the template otherwise you can use the “Custom data source” function which will be introduced in the below.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step IV:Save Template. Just use the word's "save as" function , save the template to the LmsShareFolder which predefine in your configuration file.

2. How to create template with custom data source

Step I: Open Word

Step II: Enter Custom Data Source. Click ribbon item "Lms"->"Custom data source". Enter the data source name.

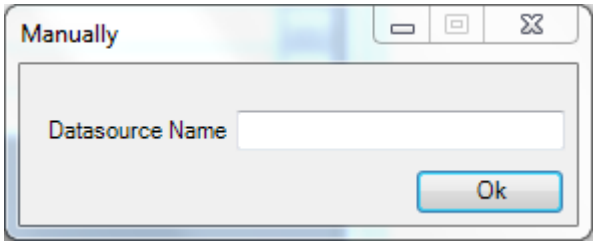


Figure 25.16. Custom data source

Step III: Build the custom data source. Edit the custom data source in the task pane. You can edit(Add,Edit,Delete) the data source via the context menu.

Note

Aspose and html mail merge handler support the "DataTable","DataSet","DataView","DataRow","Dictionary" as the data source. But the data source can come from anywhere like database,memory and so on. If the data source is from the memory you can use this function to create the template otherwise you can use the “Import from database“ function which introduced in the above.

Step IV: Create Template. Write the document and drag the mail merge fields to the required place. If you drag drop an item which don't has child nodes, the item will be treated as a field and display like "<<Filed>>". If you drag drop an item which has some child nodes it will be treated as a table and display like"<<tablestart:table>> <<field>> <<tableend:table>>". At the top of this task pane there is a check box called "Auto Create Table". If it is checked and you drag drop the table item it will help you to create a grid.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

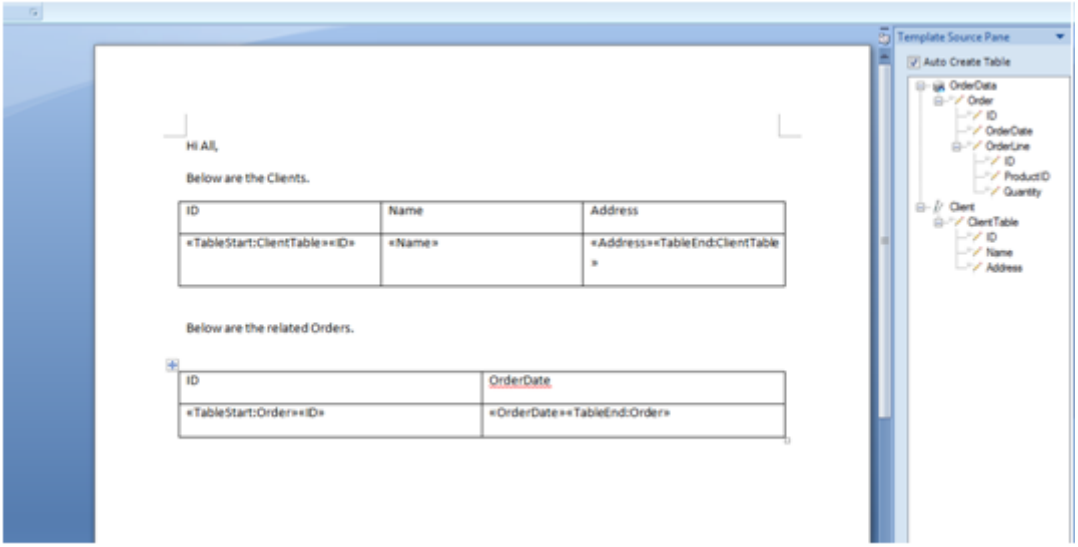


Figure 25.17. Template2

Step V:Save Template. Just use the word's "save as" function , save the template to the LmsShareFolder which predefine in your configuration file.

3. How to update a template

   Step I: Open the template

   Step II:Modify the template. If the template is created by the Word template before, the task pane will be loaded and the data source structure which created in the last time will be listed here too. You can modify the template and drag drop the merge fields. If the data source in the database was changed you can re-connect to the database use the "Import from database" function to reload the new data source structure(You need to remove the originally data source by using the context menu first.) or you can edit the loaded data source structure directly by using the context menu.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

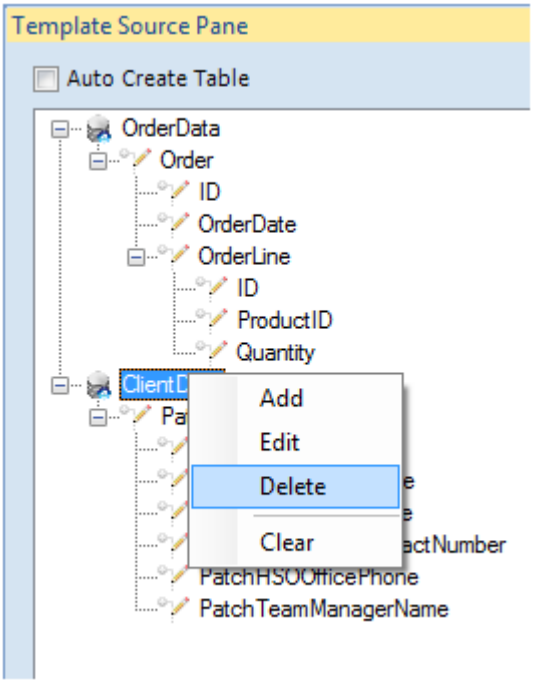


Figure 25.18. Delete data source

Step III:Save Template. Just use the word's "save as" function , save the template to the LmsShareFolder which predefine in your configuration file.

25.9.2. Html mail merge and Html Template Editor

Html mail merge is used to merge the document within the html template.Html template editor is used to maintain the html template.

1. How to use the html templaet editor in your project

   This section gives you a demonstrative sample application to guide you how to import the html editor to your project. About the html editor we recommend the TinyMCE html editor as the toolbar of this html editor can be customized by yourself.

   Step I:Import TinyMCE html editor to your asp.net project.Copy the TinyMCE html editor package to the above project. You can get it from Lms quick start. It is open source and the advantage of this control is the toolbar of this control can be customized.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

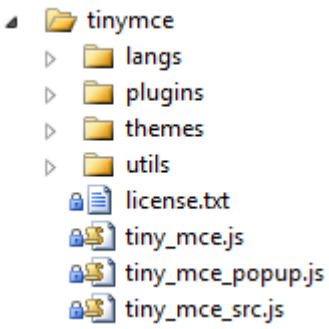


Figure 25.19. TinyMCE package

Step II: Initialize the TinyMCE html editor.Copy the below java scripts to your template edit page. Change this code" elements: "<%=htmlContentTextBox.ClientID %>"" to use your own textbox. It is used to initialize the TinyMce html editor.

```
<script type="text/javascript">
        tinymce.create('tinymce.plugins.ExamplePlugin', {
            createControl: function (n, cm) {
                switch (n) {
                    case 'lmsbutton':
                        var c = cm.createMenuButton('lmsButton', {
                            title: 'Lms Button',
                            image: 'images/mail.png',
                            icons: false
                        });

                        c.onRenderMenu.add(function (c, m) {
                            var sub;

                            m.add({ title: 'field', onclick: function () {
                                tinyMCE.activeEditor.execCommand('mceInsertContent',
                                                                    false, "{{**" +
tinymce.activeEditor.selection.getContent({ format: 'text' }) + "**}}");
                            }
                            });

                            m.add({ title: 'loop', onclick: function () {
                                tinyMCE.activeEditor.execCommand('mceInsertContent', false,
"{{$$TableStart:"
                                                                                                +
tinymce.activeEditor.selection.getContent({ format: 'text' })
                                                                                                +
"$$}}"
                                                                                                +
"{{$$TableEnd:"
                                                                                                +
tinymce.activeEditor.selection.getContent({ format: 'text' })
                                                                                               +
"$$}}");
                            }
                            });

                        });

                        // Return the new menu button instance
                        return c;
                }

                return null;
            }
        });

        // Register plugin with a short name
        tinymce.PluginManager.add('example', tinymce.plugins.ExamplePlugin);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        tinyMCE.init({
            // General options
            mode: "exact",
            elements: "<%=htmlContentTextBox.ClientID %>",
            theme: "advanced",
            plugins:
"example,autolink,lists,spellchecker,pagebreak,style,layer,table,save,advhr,advimage,

advlink,emotions,iespell,inlinepopups,insertdatetime,preview,media,searchreplace,

print,contextmenu,paste,directionality,fullscreen,noneditable,visualchars,nonbreaking,xhtmlxtras,template",

            // Theme options
            theme_advanced_buttons1:
"save,newdocument,|,bold,italic,underline,strikethrough,|,justifyleft,justifycenter

,justifyright,justifyfull,|,styleselect,formatselect,fontselect,fontsizeselect",
            theme_advanced_buttons2:
"cut,copy,paste,pastetext,pasteword,|,search,replace,|,bullist,numlist,|,outdent,indent,blockquote,|,undo

,redo,|,link,unlink,anchor,image,cleanup,help,code,|,insertdate,inserttime,preview,|,forecolor,backcolor",
            theme_advanced_buttons3:
"tablecontrols,|,hr,removeformat,visualaid,|,sub,sup,|,charmap,emotions,iespell,media,advhr,|,print,|,ltr,rtl,|,fullscreen",
            theme_advanced_buttons4:
"insertlayer,moveforward,movebackward,absolute,|,styleprops,spellchecker,|,cite,abbr

,acronym,del,ins,attribs,|,visualchars,nonbreaking,template,blockquote,pagebreak,|,insertfile,insertimage",
            theme_advanced_buttons5: "lmsbutton",
            theme_advanced_toolbar_location: "top",
            theme_advanced_toolbar_align: "left",
            theme_advanced_statusbar_location: "bottom",
            theme_advanced_resizing: true,

            // Skin options
            skin: "o2k7",
            skin_variant: "silver",

            // Drop lists for link/image/media/template dialogs
            template_external_list_url: "js/template_list.js",
            external_link_list_url: "js/link_list.js",
            external_image_list_url: "js/image_list.js",
            media_external_list_url: "js/media_list.js"
        });

    </script>
```

Note

You can customize the behavior of the “lmsButton” by changing the below code snippet. For how to use TinyMce you can refer to : http://www.tinymce.com/wiki.php/TinyMCE3x:TinyMCE_3.x

```
createControl: function (n, cm) {
     switch (n) {
      case 'lmsbutton':
           var c = cm.createMenuButton('lmsButton', {
                   title: 'Lms Button',
                   image: 'images/mail.png',
                   icons: false
                 });

           c.onRenderMenu.add(function (c, m) {
                 var sub;
                 m.add({ title: 'field', onclick: function () {
                       tinyMCE.activeEditor.execCommand('mceInsertContent', false,
"{{**"
                                                                                  +
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
tinymce.activeEditor.selection.getContent({ format: 'text' })
                                                                                +
"**}}");
                        }
                        });

                 m.add({ title: 'loop', onclick: function () {
                       tinyMCE.activeEditor.execCommand('mceInsertContent', false,
"{{$$TableStart:"
                                                                    +
tinymce.activeEditor.selection.getContent({ format: 'text' })
                                                                    + "$$}}"
                                                                 + "{{$$TableEnd:"
                                                                    +
tinymce.activeEditor.selection.getContent({ format: 'text' })
                                                                    + "$$}}");
                      }
                      });

}
```

Step III: Create the html template.Write down your words in this template and use the below highlight button to add the mail merge field. Write down the table name or field name then select the words, use the button loop or field to format the words.

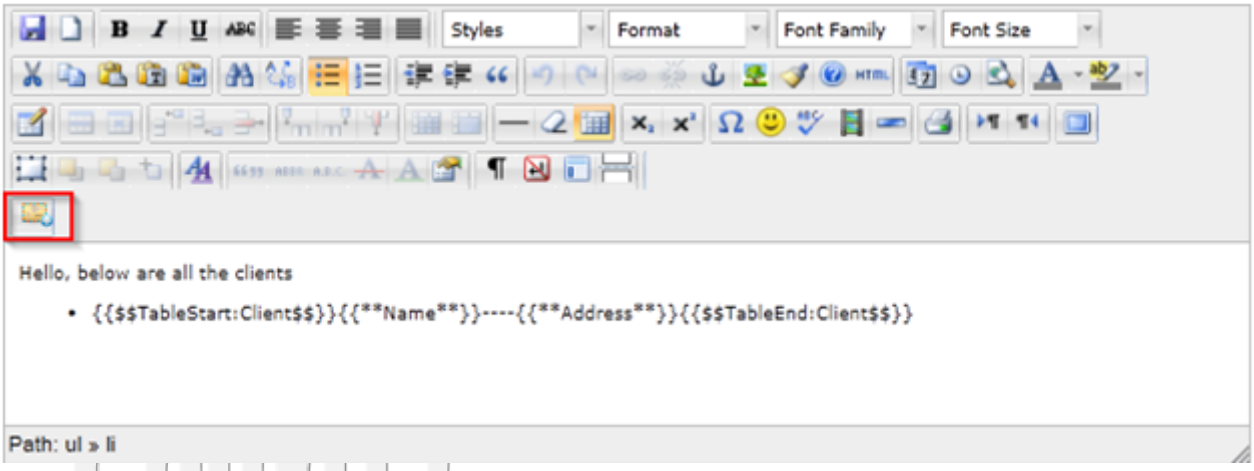


Figure 25.20. Html template

Setp IV: Save the html template. Get the text in this content textbox and save this content.

2. Html mail merge grammar

The principle of html mail merge is replacing our custom tags with the real data. The custom tags are like below:

Tag	Description
{{**FiledName**}}	Filed Tag.Just replace this tag with the real data based on the field name
{{$$TableStart:TableName$$}} {{$$TableEnd:TableName$$}}	Loop tags. Loop the content between this tags.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

For the loop tags normally we loop the content between the tags , but for the loop tags which in the table row or in the bullet, we will consider you want to loop the table row or bullet. For the below snapshoot , it will just loop the content between the loop tags.

```
{{$$TableStart:Client$$}}{{**Name**}}----{{**Address**}}{{$$TableEnd:Client$$}}
```

Figure 25.21. Loop content

The next snapshoot will loop the bullet (Note: The table start tag should at the beginning of the bullet and the table end tag should at the end of the bullet. Please ensure there is no more words before the table start tag and no words follow the t able end tag, otherwise it consider you want to just loop the content between the loop tags)

- {{$$TableStart:Client$$}}{{**Name**}}----{{**Address**}}{{$$TableEnd:Client$$}}

Figure 25.22. Loop bullet

The next snapshoot will loop the table row(Note: The table start tag should at the first cell of the row and the table end tag should at the last cell of the table. Please ensure there is no more words before the table start tag and no more words follow the table end tag, otherwise it consider you want to just loop the content between the loop tags).

Name	Address
{{$$TableStart:Client$$}} {{**Name**}}	{{**Address**}}{{$$TableEnd:Client$$}}

Figure 25.23. Loop table

25.10. Mailing Service

LMS is an engine to process and delivery documents. Pipeline based design gives LMS great extensibility. If we need to generate a document and finally deliver it by sending a mail, we have to create a job and manually attaching mailing based extensions to it.

In order to make it easier to program mailing, some application programming interface is separately define a new facade class `MailManager`. As illustrated in the following code snippet, there are many methods defined in `MailManager` to send a mail we want. The content of the mail body can originally come from a template file, a stream or a template maintained in LMS. The mail message mainly carries mail address, subject and image attachments can be provided by a manually created MailMessage, or a mail message maintained in LMS.

```
public static class MailManager
{
    public static string SendMailWithTemplateFile(string templateFilePath, MailMessage mailMessage);
    public static string SendMailWithTemplateFile(string templateFilePath, string mailMessageName);
    public static string SendMailWithTemplateFile(string templateFilePath, MailMessage mailMessage,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        IEnumerable<object> dataSources, IEnumerable<object> regionalDataSources);
    //...

    public static string SendMailWithTemplateName(string templateName, MailMessage mailMessage);
    public static string SendMailWithTemplateName(string templateName, string mailMessageName);
    //...

    public static string SendMailWithTemplateStream(Stream templateStream, DocumentFormat
templateFormat,
        MailMessage mailMessage);
    public static string SendMailWithTemplateStream(Stream templateStream, DocumentFormat
templateFormat,
        string mailMessageName);
    //...
}
```

25.10.1. How to Send Mail with HTML Template

The above getting started application demonstrates send a mail with an MS Word document based template. In this section, we will demonstrate to use HTML template instead. In addition, this sample will demonstrates how to add attachment to the mail message and embed an inline image into the mail body. We directly use the existing getting-started application. In order to use HTML template based mail merge and embedded image, we need to add the following two assembly references for the LmsDemo project:

- NCS.IConnect.Lms.Extensions.Html
- System.Drawing

The configured Aspose.Word based mail merge document handler will be replaced using the following configuration. As this configuration illustrates, such a mail merge document handler type is `PagingSortNCS.IConnect.Html.MailMerge`, which is defined in the assembly *NCS.IConnect.Lms.Extensions.Html.dll*.

```
<iConnect.lms ...>
  <documentHandlers>
    <add name="mailMerge"
         type="PagingSortNCS.IConnect.Html.MailMerge, NCS.IConnect.Lms.Extensions.Html"
         order="1"/>
    <add name="fileConverter"
          type="NCS.IConnect.Lms.Aspose.ConvertFormatDocumentHandler,
NCS.IConnect.Lms.Extensions.Aspose"
          order="2"/>
  </documentHandlers>
</iConnect.lms>
```

We create an HTML file as the template. The following code snippet represents the HTML of the template file. This template has the same content as the cover letter template in the getting-started demonstrative application. There are only two fields, FirstName and LastName. In addition, there is an image embedded, whose content id is "logo" (*cid:logo*").

```
Hi {{**FirstName**}} {{**LastName**}}:
<p>The order submitted by you has been received, and we will arrange someone to process it. The
details about the
order are included in the attached PDF file.</p>
<hr />
<p>
    <img src="cid:logo" alt="logo" />
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
</p>
<p>This e-mail and any attachments thereto are intended for the sole use of the recipient(s) named
above and may
contain information that is confidential and/or proprietary to the NCS Group. Any use of the
information contained
herein (including, but not limited to, total or partial reproduction, communication, or dissemination
in any form)
by persons other than the intended recipient(s) is prohibited. If you have received this e-mail in
error, please
notify the sender immediately and delete it.</p>
```

After performing mail merge with data sources provided, the content of this template will be used as the body of the mail to send. In addition, we have to provide an image for the *<img>* element. We will also provide a PDF file, which includes the order detail information, as the attachment of the mail. Such an mail can be sent by the following program.

```
//Create an MailMessage
MailMessage mailMessage = new MailMessage("...", "Submited order is received");
mailMessage.To.Add("...");

//Add an image matching the content ID of the <img> in the template
ImageAttachment image = new ImageAttachment("logo", @"C:\Lms\Workspace\Templates\logo.png");
mailMessage.Images.Add(image);

//Load the PDF as the attachment
Attachment attachment = new Attachment("OrderList.pdf", @"C:\Lms\Workspace\Templates\OrderList.pdf");
mailMessage.Attachments.Add(attachment);

//Create data datasource.
Dictionary<string, object> dictionary = new Dictionary<string, object>();
dictionary.Add("FirstName", "Henry");
dictionary.Add("LastName", "Wang");

//Send mail
MailManager.SendMailWithTemplateFile(@"C:\Lms\Workspace\Templates\CoverLetter.html", mailMessage,
    new object[]{dictionary},null);
```

If we run the LmsDemo application after starting the DeliveryService, an mail will be sent to the receipts specified in the MailMessage. As illustrated by the following figure, this mail is the same with the one generated in the getting-started application.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

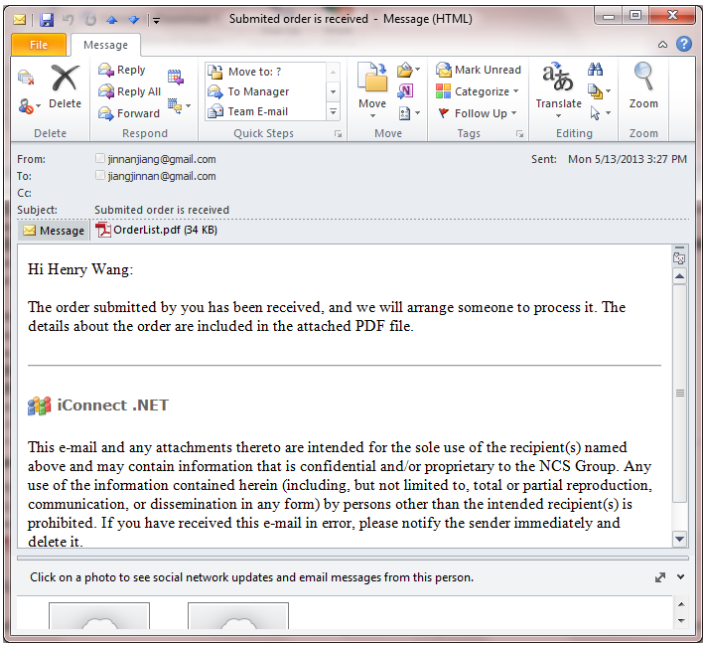


Figure 25.24. Mail generated with HTML template

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 26. Message Compression

26.1. Introduction

Message Compression is a `WCF` extension which leverages the custom message encoder to compress/decompress message. For message exchange with a large scale of payload, compressing the outgoing message can greatly save bandwidth. Two compression algorithms, *GZip* and *Deflate* are supported. It does not make sense to compress small-scale message, so this extension enable you to set the minimum size of the message to compress. The outgoing message whose size is lower than it will be send as it is.

26.2. Programming

The custom message encoder used for message compression is applied to WCF channel layer by a custom message encoding element, `CompressedTextMessageEncodingBindingElement` which is defined in `NCS.IConnect.Core` assembly. Message compression can be achieved by using a custom binding with such a message encoding binding element. The following code snippet represents the basic configuration for CompressedTextMessageEncodingBindingElement.

```
<system.serviceModel>
  <extensions>
    <bindingElementExtensions>
      <add name = "textCompression"
           type =
"NCS.IConnect.Core.WCF.Compression.Configuration.CompressedTextMessageEncodingConfigurationElement,
NCS.IConnect.Core" />
    </bindingElementExtensions>
  </extensions>
  <bindings>
    <customBinding>
      <binding name="compressedBinding">
        <textCompression compressAlgorithm="GZip"❶
          <textEncoding maxReadPoolSize  = "..."
                        maxWritePoolSize = "..."
                        messageVersion   = "..."
                        writeEncoding    = "...">
            <readerQuotas  maxArrayLength         = "..."
                           maxBytesPerRead        = "..."
                           maxDepth               = "..."
                           maxStringContentLength = "..."
                           maxNameTableCharCount  = "..."/>
          </textEncoding>❷
        </textCompression>
        ...
      </binding>
    </customBinding>
  </bindings>
</system.serviceModel>
```

❶ The compression algorithm. The available options include GZip, Deflate and Custom. Gzip is default value;

❷ The text encoding based configuration, and please refer to MSDN online document about WCF for details.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

26.3. Extensions

As illustrated by the above configuration snippet, only two compression algorithms are supported, GZip and Deflate. The other compression method can be adopted by creating custom Compressor. An abstract Compressor class is defined in iConnect.NET, and the following code snippet represents its definition.

```
public abstract class Compressor
{
    public abstract Func<Stream, CompressionMode, bool, Stream> CompressionStreamCreator { get; }❶
    public abstract string ContentType { get; }❷
    public abstract string MediaType { get; }❸
   public virtual ArraySegment<byte> Compress(ArraySegment<byte> buffer, BufferManager bufferManager,
int messageOffset);
    public virtual ArraySegment<byte> Decompress(ArraySegment<byte> buffer, BufferManager
bufferManager);
}
```

❶ The delegate to create the stream to read/write decompressed/compressed content;

❷ The content type of the message body.

❸ The media type of the message body.

The custom Compressor can be directly configured in the configuration element of CompressedTextMessageEncodingBindingElement. As the following configuration snippet illustrates, the customCompressorType represents the assembly qualified name of custom Compressor. In this situation, the compressAlgorithm must be Custom.

```
<system.serviceModel>
  <bindings>
    <customBinding>
      <binding name="compressedBinding">
        <textCompression compressAlgorithm="Custom"
                         customCompressorType = "..."
        </textCompression>
        ...
      </binding>
    </customBinding>
  </bindings>
</system.serviceModel>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 27. ASP.NET MVC Extensions

27.1. Introduction

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and that gives you full control over markup for enjoyable, agile development. It includes many features that enable fast, TDD-friendly development for creating sophisticated applications that use the latest web standards. ASP.NET MVC is an web application development framework of high extensibility, and a series of extensions are made to integrate with existing *iConnect.NET* components, such as IoC, exception handling, code table, messaging, etc. In addition, the ASP.NET MVC built-in validation is greatly enhanced.

27.2. IoC

The *Service Location* based IoC is the core and foundation of the whole *iConnect.NET* framework. We apply it to ASP.NET MVC by customizing `ControllerFactory` to make it possible to achieve IoC based `Controller` activation.

27.2.1. Getting Started

This demonstration guides you how to register IoC based `ControllerFactory` to activate the target `Controller` instance. We first create an empty web project using the ASP.NET MVC project template and name it `MvcApp`. We then add the assembly reference of `NCS.IConnect.Web.Mvc.dll` to this project. In the *Models* directory, we add a class of `Contact`, which will be used as the *Model*. The following code snippet illustrates the definition of `Contact` type.

```
public class Contact
{
    public string Name { get; set; }
    public string PhoneNo { get; set; }
    public string EmailAddress { get; set; }
}
```

This application demonstrates how to retrieve a list of contact and render them in the web page. We define the following `IContactRepository` interface, which can be considered as the abstract to the contact domain model. Only one `GetAllContacts` method is defined to get all contacts from data storage. The class `ContactRepository` implements this interface. Both the two are defined in the *Models* directory.

```
public interface IContactRepository
{
    IEnumerable<Contact> GetAllContacts();
}
public class ContactRepository : IContactRepository
{
    public IEnumerable<Contact> GetAllContacts()
    {
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        return new Contact[]{
            new Contact{ Name = "Foo", PhoneNo="123", EmailAddress="foo@ncsi.com.cn"},
            new Contact{ Name = "Bar", PhoneNo="456", EmailAddress="bar@ncsi.com.cn"},
            new Contact{ Name = "Baz", PhoneNo="789", EmailAddress="baz@ncsi.com.cn"}
        };
    }
}
```

We create a default `HomeController` in the *Controllers* directory. As the following code snippet illustrates, there is a Repository property of `IContactRepository` type, which is initialized in constructor. In terms of IoC, this kind of injection is called constructor injection. In the default Index action method, the contact list is retrieved and rendered in the default view.

```
public class HomeController : Controller
{
    public IContactRepository Repository { get; private set; }
    public HomeController(IContactRepository repository)
    {
        this.Repository = repository;
    }
    public ActionResult Index()
    {
        return View(this.Repository.GetAllContacts());
    }
}
```

The following code snippet represents the definition of the `Index` view, which is strongly typed view with the `IEnumerable<Contact>` as its model type. In this view, each contact related information is rendered in an unordered list (*<ul>*).

```
@model IEnumerable<MvcApp.Models.Contact>
@foreach (var contact in Model)
{
    <ul>
        <li>Name: @contact.Name</li>
        <li>PhoneNo: @contact.PhoneNo</li>
        <li>EmailAddress: @contact.EmailAddress</li>
    </ul>
}
```

We then need to register the type mapping between `IContactRepository` and `ContactRepository`. This can be achieved by the following *Unity* related configuration.

```
<configuration>
  <configSections>
    <section name="unity"
             type="Microsoft.Practices.Unity.Configuration.UnityConfigurationSection,
             Microsoft.Practices.Unity.Configuration"/>
  </configSections>
  <unity>
    <container>
      <register type="MvcApp.Models.IContactRepository, MvcApp"
                mapTo="MvcApp.Models.ContactRepository, MvcApp"/>
    </container>
  </unity>
</configuration>
```

IoC based controller activation is implemented by the `NCS.IConnect.Web.Mvc.IoC.ServiceLocatableControllerFactory`, which is a custom `ControllerFactory`, so we need to register it into the ASP.NET MVC controller activation

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

system before the application starts. The `ServiceLocatableControllerFactory` registration can be achieved by the following code defined in the `Global.asax`.

```
public class MvcApplication : System.Web.HttpApplication
{
    //Other Members
    protected void Application_Start()
    {
        //Other Operations
        ControllerBuilder.Current.SetControllerFactory(
            new ServiceLocatableControllerFactory());
    }
}
```

When we run the demonstration application, the contact list will be rendered in the browser as below.

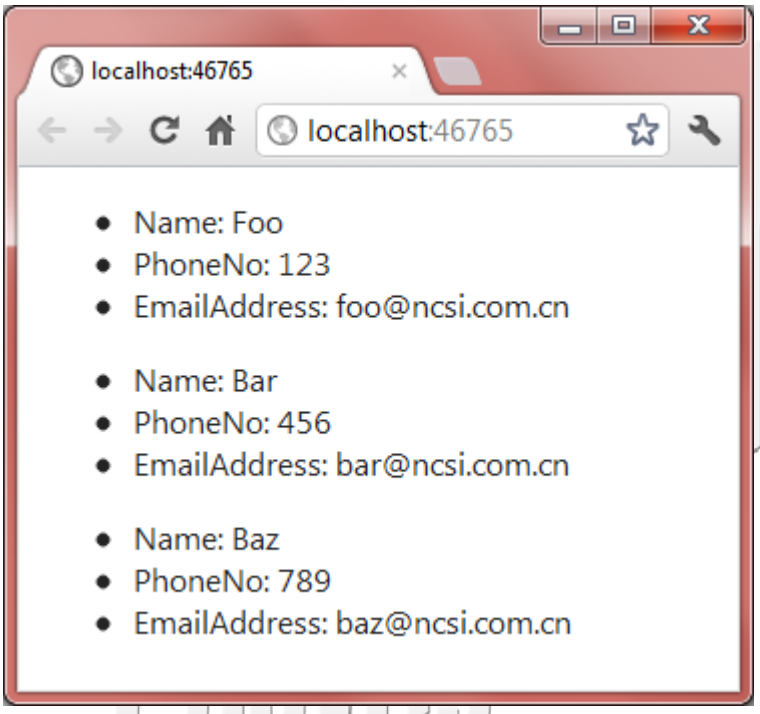


Figure 27.1. Contact List rendered in Browser

27.2.2. `ServiceLocatableControllerFactory`

Applying IoC to ASP.NET MVC mainly embodies to activate the target controller in an IoC manner. Controller activation in ASP.NET MVC is achieved by `ControllerFactory`, so a custom `ControllerFactory`, named `ServiceLocatablControllerFactory` is created to make it possible to activate the target controller instance using `Service Location` based IoC mechanism. The following code snippet represents the definition of `ServiceLocatablControllerFactory`.

```
public class ServiceLocatableControllerFactory : DefaultControllerFactory
{
    public ServiceLocatableControllerFactory(string serviceLocatorName = ""❶);
    public NCS.IConnect.Core.IoC.IServiceLocator ServiceLocator { get; }❷
    protected override IController GetControllerInstance(
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        RequestContext requestContext, Type controllerType);
}
```

❶ The configuration name of the `ServiceLocator`. The default `ServiceLocator` will be used if not explicitly specified.

❷ The `ServiceLocator` used to activate the target controller instance.

27.3. Code Table

There are two basic functionalities for *iConnect.NET* Code Table component, code maintenance and list controller binding. However, there is never control concept in terms of ASP.NET MVC, and all of the HTML are manually customized. In order to make it easy to generate code related HTML, a series of extension methods against `HtmlHelper`/`HtmlHelper<TModel>` are defined. These extensions methods help you automatically generated HTML specific to the traditional `ListControl`, such as `DropDownList`, `ListBox`, `RadioButtonList` and `CheckBoxList`.

27.3.1. Getting Started

In this section, we will provide you with a demonstration application to explore the integration between ASP.NET MVC and iConnect.NET Code Table. In this ASP.NET MVC application,with the help of predefined templates, the `ListControl` based HTML will be automatically generated by calling the template view helper methods.

Step I. Create Application Record and Codes

It is supposed you have a SQL Server database with all iConnect.NET objects, including tables, stored procedures, user defined functions, etc. If such a database is not available, you can use the script file provided by *iConnect.NET* to create it. The path of such a script file is *%iConnect.NET installation directory%Database Scripts\MsSqlCreateObjects.sql*. You firstly need to execute the following SQL script in this database to insert a new record, which stands for our demonstration application, into the `T_IC_APP` table. Our demonstration application is named "*CodeTableIntegrationDemo*".

```
INSERT INTO [dbo].[T_IC_APP]([APP_ID] ,[APP_NAME] ,[LOWERED_APP_NAME] ,[APP_DESC] ,[FLAG] ,[VERSION_NO]
,
[TRANSACTION_ID],[CREATED_BY] ,[CREATED_TIME] ,[LAST_UPDATED_BY] ,[LAST_UPDATED_TIME])
VALUES('3B9413F6-3644-4733-8CD6-DA08B203D751' ,'CodeTableIntegrationDemo' ,'codetableintegrationdemo' ,
'' ,'' ,1 ,'5107213E-8C03-49E4-A32A-C482C2B8DDA7','Sys' ,GETDATE() ,'Sys' ,GETDATE())
```

We then execute the following SQL script to create four categories of codes in the `T_IC_CODE` table. The four categories are `Gender`, `MaritalStatus`, `Department` and `Skill`.

```
INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Gender','gender','M','m','Male',
'','1900-1-1','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722','Sys',
GETDATE(),'Sys',GETDATE())
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Gender','gender','F','f','Female',
'','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722','Sys',GETDATE(),
'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'MaritalStatus','maritalstatus','S',
's','Single','','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722',
'Sys',GETDATE(),'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'MaritalStatus','maritalstatus','M',
'm','Married','','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722',
'Sys',GETDATE(),'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Department','department','Dev1',
'dev1','DEV I','','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722',
'Sys',GETDATE(),'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Department','department','Dev2',
'dev2','DEV II','','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722',
'Sys',GETDATE(),'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Department','department','Dev3',
'dev3','DEV III','','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722',
'Sys',GETDATE(),'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Skill','skill','','C','C#','',
'1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722','Sys',GETDATE(),
'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Skill','skill','','ASP','ASP.NET',
'','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722','Sys',GETDATE(),
'Sys',GETDATE())

INSERT INTO [dbo].[T_IC_CODE]
([CODE_ID],[APP_ID],[CODE_CATEGORY],[LOWERED_CODE_CATEGORY],[CODE],[LOWERED_CODE],[CODE_DESC],
[CODE_REMARKS],[EFFECTIVE_START_DATE],[EFFECTIVE_END_DATE],[CODE_SEQ],[IS_DELETED],[VERSION_NO],
[TRANSACTION_ID],[CREATED_BY],[CREATED_TIME],[LAST_UPDATED_BY],[LAST_UPDATED_TIME])
VALUES(newid() ,'3B9413F6-3644-4733-8CD6-DA08B203D751' ,'Skill','skill','','ADO','ADO.NET',
'','1900-01-01','9999-12-31',NULL,'0',1,'AA0D7AFE-E615-4328-94C4-8CCDFB4F8722','Sys',
GETDATE(),'Sys',GETDATE())
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Step II Create ASP.NET MVC Application and Configure Code Table

Create an empty web application using the ASP.NET MVC project template, and then add the assembly reference to `NCS.IConnect.Web.Mvc.Html.dll`, which is located in the \bin sub folder of *iConnect.NET* installation directory. In the *web.config* file located in the root directory, copy the following XML snippet to configure the `Code Table`. Please do not forget revise the database connection string accordingly.

```
<configuration>
  <configSections>
    <section name="iConnect.codeTable"
             type="NCS.IConnect.CodeTable.Configuration.CodeTableSettings,
                   NCS.IConnect.CodeTable"/>
  </configSections>
  <connectionStrings>
    <add name="iConnectDb"
         providerName="System.Data.SqlClient"
         connectionString="..."/>
  </connectionStrings>
  <iConnect.codeTable defaultProvider="dbProvider">
    <codeTableProviders>
      <add name="dbProvider"
           applicationName="CodeTableIntegrationDemo"
           database="iConnectDb"
           type="NCS.IConnect.CodeTable.DbCodeTableProvider,
                 NCS.IConnect.CodeTable"/>
    </codeTableProviders>
  </iConnect.codeTable>
  ...
</configuration>
```

Step III Create ListControl Specific Templates

Create a sub folder "*EditorTemplates*" in the directory */Views/Shared*. In the newly created directory, create four partial views whose name are separately *DropDownList.cshtml*, *ListBox.cshtml*, *RadioButtonList.cshtml* and *CheckBoxList.cshtml*. As their names suggest, they are separately specific to four kinds of `ListControl` based templates. The following code snippet represents the definition of the four template views.

```
DropDownList.cshtml:
@model string
@{
    string codeCategory = (string)ViewData.ModelMetadata.AdditionalValues["CodeCategory"];
    @Html.DropDownList2("",codeCategory,Model)
}

ListBox.cshtml:
@model IEnumerable<string>
@{
    string codeCategory = (string)ViewData.ModelMetadata.AdditionalValues["CodeCategory"];
    @Html.ListBox2("", codeCategory, Model)
}

RadioButtonList.cshtml:
@model string
@{
    string codeCategory = (string)ViewData.ModelMetadata.AdditionalValues["CodeCategory"];
    @Html.RadioButtonList("", codeCategory,Model)
}

CheckBoxList.cshtml:
@model IEnumerable<string>
@{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    string codeCategory = (string)ViewData.ModelMetadata.AdditionalValues["CodeCategory"];
    @Html.CheckBoxList("", codeCategory,Model)
 }
```

Step IV Program Model, View and Controller

In the *Models* directory, we create the following `Developer` class to used as the *Model*. As the following code snippet illustrates, except the Name property, the rest four ones are specific to four code categories. In the these properties, different `ListControl` based attributes are applied which will determine the style rendered in web page. The string parameters in these attributes represent the corresponding code category.

```
public class Developer
{
    public string Name { get; set; }

    [RadioButtonList("Gender")]
    public string Gender { get; set; }

    [DropdownList("MaritalStatus")]
    public string MaritalStatus { get; set; }

    [ListBox("Department")]
    public IEnumerable<string> Departments { get; set; }

    [CheckBoxList("Skill")]
    public IEnumerable<string> Skills { get; set; }
}
```

We then create the default `HomeController` in the *Controllers* directory. As the following code snippet illustrates, a new `Developer` instance is created and is rendered in the default view in the Index action method.

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        Developer dev = new Developer
        {
            Name          = "Foo",
            Gender        = "M",
            Departments   = new string[] { "Dev1", "Dev2" },
            MaritalStatus = "S",
            Skills        = new string[] { "C", "ASP" }
        };
        return View(dev);
    }
}
```

The following code snippet represents the definition of the `Index` action specific view. This is a strongly typed view whose *Model* type is `Developer`. In this view, all of the properties of the *Model* instance (the `Developer` object) are rendered in edit mode.

```
@model MvcApp.Models.Developer
<div>Name</div>
<div>@Html.EditorFor(d=>d.Name)</div>
<div>Gender</div>
<div>@Html.EditorFor(d => d.Gender)</div>
<div>Marital Status</div>
<div>@Html.EditorFor(d => d.MaritalStatus)</div>
<div>Departments</div>
<div>@Html.EditorFor(d => d.Departments)</div>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
<div>Skills</div>
<div>@Html.EditorFor(d => d.Skills)</div>
```

After running the web application, the result illustrated in the following figure will appear in the browser. We can see the four properties are separately corresponding to different `ListControl` type, and they are consistent with the attributes decorated.

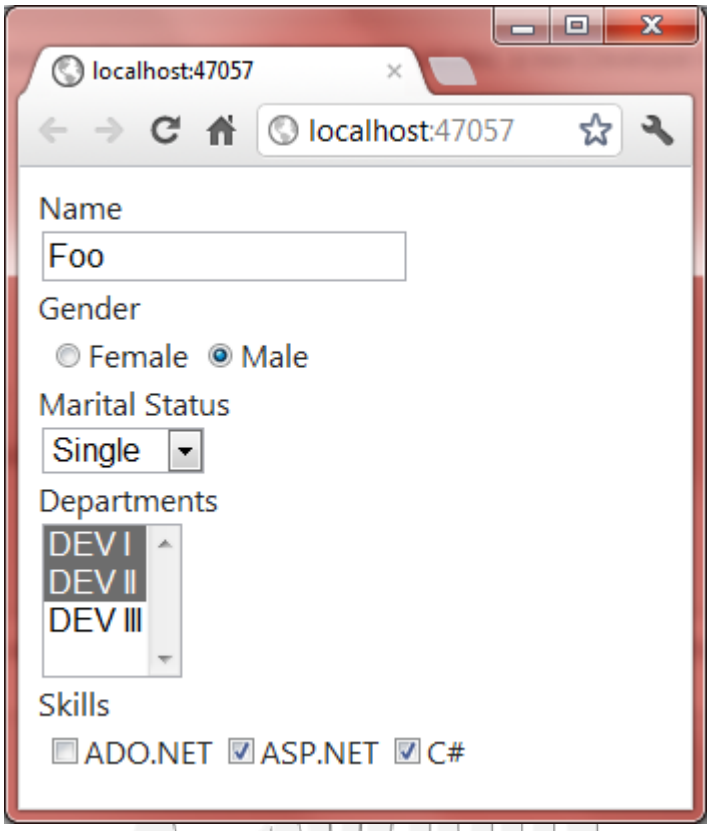


Figure 27.2. Four Kinds of ListControl

27.3.2. `ListControlAttribute`

Four kinds of `ListControl` based attributes are defined, which are mainly used to determine which kind of `ListControl` will be rendered for the target property. The four attributes derived from the same base class, `ListControlAttribute`. As the following code snippet illustrates, `ListControlAttribute` implements `IMetadataAware` interface and perform customization against the `ModelMetadata` in the `OnMetadataCreated` method.

```
[AttributeUsage(AttributeTargets.Property)]
public abstract class ListControlAttribute : Attribute, IMetadataAware
{
    public string CodeCategory { get; private set; }
    public ListControlAttribute(string codeCategory);
    protected abstract string TemplateHint { get; }
    public virtual void OnMetadataCreated(ModelMetadata metadata);
}
```

The CodeCategory property represents the corresponding code category name, and it is initialized in constructor. The abstract property TemplateHint returns a string which stands for the template name. In the `OnMetadataCreated` method, the given `ModelMedata` is customized

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

accordingly. The value of the CodeCategory property is added in the AdditionalValues of the `ModelMedata`, and the key is "*CodeCategory*". The TemplateHint property corresponds to the TemplateHint property of `ModelMedata`.

The following four concrete attributes directly inherit from the abstract `ListControlAttribute`. As with the implemented abstract TemplateHint properties, they separately return "*DropdownList*", "*ListBox*", "*RadioButtonList*" and "*CheckBoxList*".

```
[AttributeUsage(AttributeTargets.Property)]
public class DropdownListAttribute : ListControlAttribute
{
    public DropdownListAttribute(string codeCategory)
        : base(codeCategory)
    { }
    //Return "DropdownList"
    protected override string TemplateHint;
}

[AttributeUsage(AttributeTargets.Property)]
public class ListBoxAttribute : ListControlAttribute
{
    public ListBoxAttribute(string codeCategory)
        : base(codeCategory)
    { }
    //Return "ListBox"
    protected override string TemplateHint;
}

[AttributeUsage(AttributeTargets.Property)]
public class RadioButtonListAttribute : ListControlAttribute
{
    public RadioButtonListAttribute(string codeCategory)
        : base(codeCategory)
    { }
    //Return "RadioButtonList"
    protected override string TemplateHint;
}

[AttributeUsage(AttributeTargets.Property)]
public class CheckBoxListAttribute : ListControlAttribute
{
    public CheckBoxListAttribute(string codeCategory)
        : base(codeCategory)
    { }
    //Return "CheckBoxList"
    protected override string TemplateHint;
}
```

Once the a particular `ListControlAttribute` is applied to a given property, its template name is determined. We can create corresponding template in the form of partial view to control the HTML to render. In the getting started demonstration, the `ListControl` based HTML are all achieved by template.

27.3.3. `HtmlHelper` Extensions

The four kinds of `ListControl` based HTML rendering are implemented by calling the related extension method of `HtmlHelper`/`HtmlHelper<TModel>`.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

27.3.3.1. DropDownList

Two `DropDownList2` method overloads and one `DropDownListFor2<TModel, TProperty>` methods are defined in the static `DropDownListExtensions` class. They will get the code list by the given code category and generate <*select*> element based HTML. The following code snippet represents the definition of the three extension methods.

```
public static class DropDownListExtensions
{
    public static MvcHtmlString DropDownList2(this HtmlHelper htmlHelper, string name,
        string codeCategory, CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null, object htmlAttributes = null);

    public static MvcHtmlString DropDownList2(this HtmlHelper htmlHelper, string name❶,
        string codeCategory❷, string value❸, CodeStatus status = CodeStatus.Effective❹,
        DateTime? date = null❺, CodeBindingOptions bindingOptions = null❻,
        object htmlAttributes = null❼);

   public static MvcHtmlString DropDownListFor2<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,
        Expression<Func<TModel, TProperty>> expression, string codeCategory,
        CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null, object htmlAttributes = null);
}
```

❶ The name of the HTML <select> element;

❷ The code category name;

❸ The status of the code to bind. Only the effective codes will be bound by default;

❹ The current value to bind;

❺ The baseline date time used for code effectiveness calculation. The current date time will be used by default;

❻ The code binding option to control the binding behavior. If this parameter is not explicitly specified, the configured one will be used;

❼ The other HTML attributes append to the <select> element.

27.3.3.2. ListBox

The `ListBox` based extension methods' definition is similar to the above `DropDownList` based one. In fact, the two kinds of `ListControl` are both corresponding to the <*select*> element, however the `ListBox` based one has an extra "*multiple*" attribute. As the following code snippet illustrates, the type of *`values`* parameter is `IEnumerable<string>` as `ListBox` supports multiple selection.

```
public static class ListBoxExtensions
{
    public static MvcHtmlString ListBox2(this HtmlHelper htmlHelper, string name,
        string codeCategory, CodeStatus status = CodeStatus.Effective,
        DateTime? date = null, CodeBindingOptions bindingOptions = null,
        object htmlAttributes = null);

    public static MvcHtmlString ListBox2(this HtmlHelper htmlHelper, string name,
        string codeCategory, IEnumerable<string> values,
        CodeStatus status = CodeStatus.Effective,
        DateTime? date = null, CodeBindingOptions bindingOptions = null,
        object htmlAttributes = null);

    public static MvcHtmlString ListBoxFor2<TModel, TProperty>(this HtmlHelper<TModel> htmlHelper,
        Expression<Func<TModel, TProperty>> expression, string codeCategory,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null, object htmlAttributes = null);
}
```

27.3.3.3. RadioButtonList

Similar extension methods are defined in the `RadioButtonListExtensions`, which will generate the radio button list related HTML based on the given category of codes.

```
public static class RadioButtonListExtensions
{
    public static MvcHtmlString RadioButtonList(this HtmlHelper helper, string name,
        string codeCategory, CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null,
        RepeatDirection repeatDirection = RepeatDirection.Horizontal,
        TextAlign textAlign = TextAlign.Right, object htmlAttributes = null);

    public static MvcHtmlString RadioButtonList(this HtmlHelper helper, string name,
        string codeCategory, string value, CodeStatus status = CodeStatus.Effective,
        DateTime? date = null, CodeBindingOptions bindingOptions = null,
        RepeatDirection repeatDirection = RepeatDirection.Horizontal❶,
        TextAlign textAlign = TextAlign.Right❷, object htmlAttributes = null);

    public static MvcHtmlString RadioButtonListFor<TModel, TProperty>(this HtmlHelper<TModel> helper,
        Expression<Func<TModel, TProperty>> expression, string codeCategory,
        CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null,
        RepeatDirection repeatDirection = RepeatDirection.Horizontal,
        TextAlign textAlign = TextAlign.Right, object htmlAttributes = null);
}
```

❶ Gets or sets the direction in which the radio buttons within the group are displayed.

❷ Gets or sets the text alignment for the radio buttons within the group.

27.3.3.4. CheckBox & CheckBoxList

In the following static class `CheckBoxExtensions`, the `CheckBox` extension method against `HtmlHelper` is defined to generate the `CheckBox` related HTML with the value specified.

```
public static class CheckBoxExtensions
{
    public static MvcHtmlString CheckBox(this HtmlHelper htmlHelper, string name❶,
       bool isChecked❷, string value❸, object htmlAttributes = null);
}
```

❶ The name of the check box specific <input> element;

❷ The boolean value which indicates whether the check box is checked;

❸ The value of the check box.

`CheckBoxList` specific extension method against `HtmlHelper`/`HtmlHelper<TModel>` are defined in the following static `CheckBoxListExtensions` type. They are similar to the `RadioButtonList` based ones except the `IEnumerable<string>`, instead of string is used as the value type.

```
public static class CheckBoxListExtensions
{
    public static MvcHtmlString CheckBoxList(this HtmlHelper helper, string name,
        string codeCategory, CodeStatus status = CodeStatus.Effective,
        DateTime? date = null, CodeBindingOptions bindingOptions = null,
        RepeatDirection repeatDirection = RepeatDirection.Horizontal,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
        TextAlign textAlign = TextAlign.Right, object htmlAttributes = null);

    public static MvcHtmlString CheckBoxList(this HtmlHelper helper, string name,
        string codeCategory, IEnumerable<string> values,
        CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null,
        RepeatDirection repeatDirection = RepeatDirection.Horizontal,
        TextAlign textAlign = TextAlign.Right, object htmlAttributes = null);

    public static MvcHtmlString CheckBoxListFor<TModel, TProperty>(this HtmlHelper<TModel> helper,
        Expression<Func<TModel, TProperty>> expression, string codeCategory,
        CodeStatus status = CodeStatus.Effective, DateTime? date = null,
        CodeBindingOptions bindingOptions = null,
        RepeatDirection repeatDirection = RepeatDirection.Horizontal,
        TextAlign textAlign = TextAlign.Right, object htmlAttributes = null);
}
```

27.4. Exception Handling

ASP.NET MVC specific exception handling leverages *EntLib* Exception Handling Application Block to automatically handle the exception thrown in *Action* methods of *Controller*. The exception thrown during invoking action methods is not only handled based on the given exception policy, the error message can also be displayed accordingly as well. As with error reporting, there are two kinds of manners: showing the error message by invoking the matched *handle-error-action* method, or render the error message in appropriate error view.

27.4.1. Getting Started

In this getting-started demonstration application, we will guide you how to leverage EntLib Handling Application Block to automatically handle the exception which thrown in a particular action method, and display the customized error message in the current view.

Step I Create ASP.NET MVC Application and Program Controller and View

Create an empty web application using the ASP.NET MVC project template of Visual Studio and add the assembly reference to the `NCS.IConnect.Web.Mvc.dll` and `NCS.IConnect.ExceptionHandling.dll`, which is located in the sub-folder *\bin* of the installation directory. This application demonstrates the login scenario. We create the following `LoginInfo` class in the *Models* directory.

```
public class LoginInfo
{
    [DisplayName("User Name")]
    public string UserName { get; set; }

    [DisplayName("Password")]
    [DataType(DataType.Password)]
    public string Password { get; set; }
}
```

In the *Models* directory, the following two exception classes are also created. The `InvalidUserNameException` will be thrown when entering an invalid user to sign in, and when the password does not match the user name, the `InvalidPasswordException` is thrown.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
[Serializable]
public class InvalidUserNameException : Exception
{
    public InvalidUserNameException() { }
    public InvalidUserNameException(string message) : base(message) { }
    public InvalidUserNameException(string message, Exception inner)
        : base(message, inner) { }
    protected InvalidUserNameException(
        System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context)
        : base(info, context) { }
}

[Serializable]
public class InvalidPasswordException : Exception
{
    public InvalidPasswordException() { }
    public InvalidPasswordException(string message) : base(message) { }
    public InvalidPasswordException(string message, Exception inner)
        : base(message, inner) { }
    protected InvalidPasswordException(
        System.Runtime.Serialization.SerializationInfo info,
        System.Runtime.Serialization.StreamingContext context)
        : base(info, context) { }
}
```

We then create the default `HomeController` in the *Controllers* directory. As the following code snippet illustrates, the `HomeController` inherits the `BaseController`, which is defined in `NCS.IConnect.Web.Mvc.dll`. The `ExceptionPolicyAttribute` is applied to the `HomeController` and specify the exception policy name ("*myPolicy*").

```
[ExceptionPolicy("myPolicy")]
public class HomeController : BaseController
{
    public ActionResult Index()
    {
        return View(new LoginInfo());
    }

    [HttpPost]
    [HandleErrorAction("HandleError")]
    public ActionResult Index(LoginInfo loginInfo)
    {
        if (string.Compare("foo", loginInfo.UserName, true) != 0)
        {
            throw new InvalidUserNameException();
        }
        if (loginInfo.Password != "password")
        {
            throw new InvalidPasswordException();
        }
        return View(loginInfo);
    }

    public ActionResult HandleError(LoginInfo loginInfo)
    {
        return View(loginInfo);
    }
}
```

The default view, with an empty `LoginInfo` object as the *Model*, is rendered in the `Index` action. In the HTTP-POST Index action, the `InvalidUserNameException` or `InvalidPasswordException` is thrown accordingly when user name is invalid (not "foo") or password is invalid(not "password"). The action of `HandleError` will render a view with model errors when throwing exception, and the `HandleErrorActionAttribute` applied to the `Index`

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

action method is used to match the two action. This means the `HandleError` action method will be invoked to report the errors once the exception is thrown in the `Index` action. The following code snippet represents the definition of the Index action specific view.

```
@using NCS.IConnect.Web.Mvc
@model MvcApp.Models.LoginInfo
<html>
    <head>
        <title>Login</title>
        <style type="text/css">
            .validation-summary-errors{ color: Red; font-weight:bold;}
        </style>
    </head>
    <body>
        @using(Html.BeginForm())
        {
            if (ViewData.ContainExceptionInfo())
            {
                ViewData.ModelState.AddModelError("", ViewData.GetHandleErrorInfo().ErrorMessage);
            }
            <div>@Html.ValidationSummary()</div>
            @Html.EditorForModel()
            <input type="submit" value = "Sign In" />
        }
    </body>
</html>
```

Step II Configure EntLib Exception Handling

In the *web.config*, we define an exception policy named "myPolicy", which is applied to the `HomeController` using `ExceptionPolicyAttribute`. As illustrated in the following code snippet, both the `InvalidUserNameException` and `InvalidPasswordException` are added in the exception policy. There are only one exception handler of `ErrorMessageExceptionHandler` type in the two exception types specific handler chain. As its name suggests, the `ErrorMessageExceptionHandler` is used to customize the error message. In the following configuration, a friendly error message is specified for the two exception types.

```
<configuration>
  <configSections>
    <section name="exceptionHandling"
             type="Microsoft.Practices.EnterpriseLibrary.ExceptionHandling
                   .Configuration.ExceptionHandlingSettings,
                   Microsoft.Practices.EnterpriseLibrary.ExceptionHandling"/>
     </configSections>
  <exceptionHandling>
    <exceptionPolicies>
      <add name="myPolicy">
        <exceptionTypes>
          <add name="InvalidUserNameException"
               type="MvcApp.Models.InvalidUserNameException, MvcApp"
               postHandlingAction="None">
            <exceptionHandlers>
              <add name="ErrorMessageSettingHandler"
                   type="NCS.IConnect.Web.Mvc.ExceptionHandling.ErrorMessageExceptionHandler,
                         NCS.IConnect.Web.Mvc"
                   template="This user name is invalid."/>
            </exceptionHandlers>
          </add>
          <add name="UserNamePasswordNotMatchException"
               type="MvcApp.Models.InvalidPasswordException, MvcApp"
               postHandlingAction="None">
            <exceptionHandlers>
              <add name="ErrorMessageSettingHandler"
                   type="NCS.IConnect.Web.Mvc.ExceptionHandling.ErrorMessageExceptionHandler,
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                        NCS.IConnect.Web.Mvc"
                    template="This password is invalid."/>
            </exceptionHandlers>
          </add>
        </exceptionTypes>
      </add>
    </exceptionPolicies>
  </exceptionHandling>
  ...
</configuration>
```

Step III Run the Application

When we run the web application and enter the invalid user name and password, the error messages set in the above configuration will be displayed accordingly as the following figured.

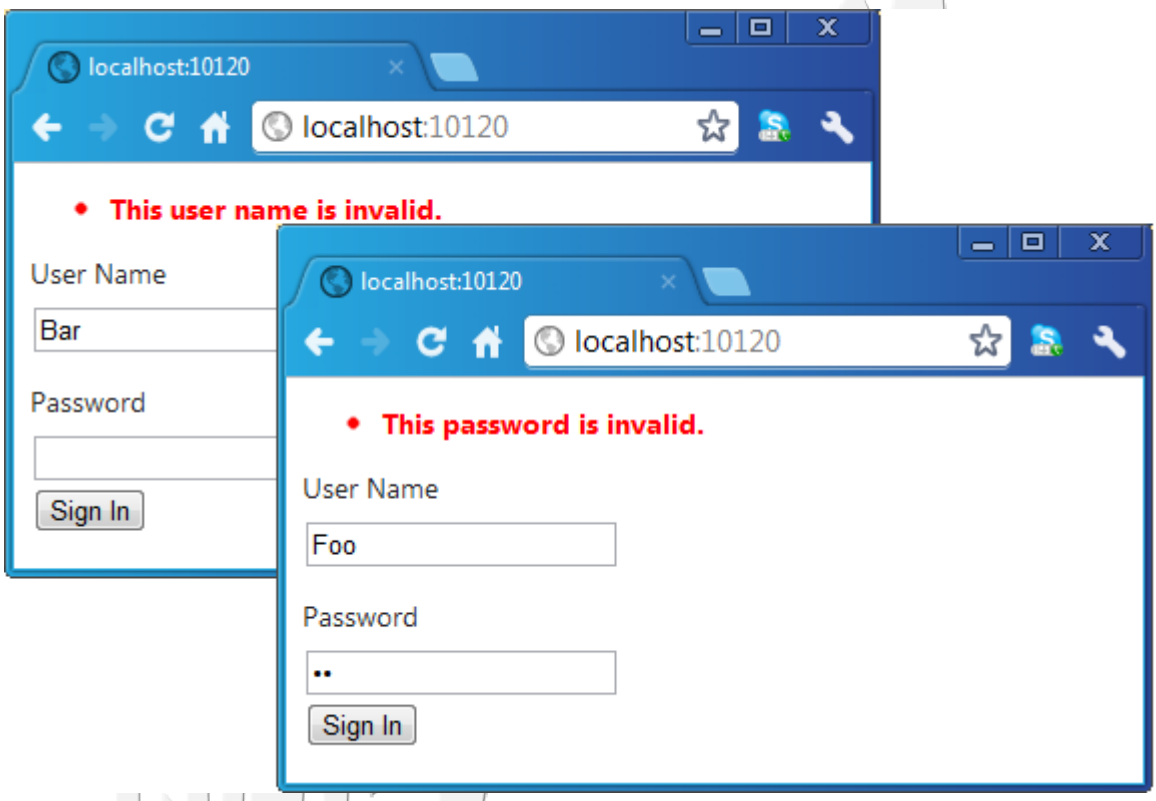


Figure 27.3. Current View Specific Error Reporting

27.4.2. `BaseController`

The integration between ASP.NET MVC and EntLib Exception Handling Application Block is achieved by a custom `Controller`, `NCS.IConnect.Web.Mvc.BaseController`, which directly inherits `System.Web.Mvc.Controller`. As the following code snippet illustrates, there is an EnableExceptionHandlingExtension property of boolean type, which indicates whether the controller supports EntLib based exception handling. This property is initialized in the constructor and the default value is true. When the project specific controller inherits such a `BaseController`, do not forget set the explicitly suppress the automatic exception handling policy if you want.

```
public class BaseController: Controller
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public bool EnableExceptionHandlingExtension { get;}
    public BaseController(bool enableValidationExtension = true,
        bool enableExceptionHandlingExtension = true);
}
```

27.4.3. Set Exception Policy

EntLib approach policy based exception handling, and we must explicitly determine which policy is used to handle the exception thrown. We can use the `NCS.IConnect.Web.Mvc.ExceptionHandling.ExceptionPolicyAttribute` to set the exception policy. As illustrated by the following code snippet, `ExceptionPolicyAttribute`'s constructor accepts a parameter of *`exceptionPolicyName`*, which represents the name of the exception policy.

```
[AttributeUsage( AttributeTargets.Class| AttributeTargets.Method)]
public class ExceptionPolicyAttribute: Attribute
{
    public string ExceptionPolicyName { get; }
    public ExceptionPolicyAttribute(string exceptionPolicyName);
}
```

`ExceptionPolicyAttribute` can be applied to both controller class and action method, and the one decorated with action method takes higher priority. We can also use the static `NCS.IConnect.Web.Mvc.ExceptionHandling.ExceptionPolicyRegister` to register a default exception policy. If neither action method nor controller class is not decorated with a `ExceptionPolicyAttribute`, the registered default exception policy will be used.

```
public static class ExceptionPolicyRegister
{
    public static Func<ActionDescriptor, string> ExceptionPolicyNameAccessor{get;set;};
    public static void SetDefaultExceptionPolicy(string exceptionPolicyName);
}
```

As the above code snippet illustrates, there is a static ExceptionPolicyNameAccessor property, which is a delegate used to get the exception policy name based on the given action descriptor. The static `SetDefaultExceptionPolicy` method can be invoked to register the default exception policy. If we need to apply different exception policy access policy, we can also reevaluate the ExceptionPolicyNameAccessor property.

27.4.4. How Exception Handling Works?

In this section we will introduce the basic process flow to handle the exception thrown in a particular action method. When an exception is thrown during a particular action method, the corresponding exception policy name will be resolved according to the aforementioned exception policy name access logical. The exception is handled by EntLib Exception Handling Application Block according to such an exception policy.

For Ajax specific request, a `NCS.IConnect.Web.Mvc.ExceptionHandling.ExceptionInfo` will be created, which wraps the exception handled. As the following code snippet illustrates, there are similar property definition between `Exception` and `ExceptionInfo`. A `JsonResult`, which is created based on the `ExceptionInfo` instance, is directly responded back to the client.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
[DataContract(Namespace = "http://www.ncs.com.sg")]
public class ExceptionInfo
{
    public ExceptionInfo(Exception exception)
    public string HelpLink { get; private set; }
    [DataMember]
    public ExceptionInfo InnerException { get; private set; }
    [DataMember]
    public string Message { get; set; }
    [DataMember]
    public string StackTrace { get; private set; }
    [DataMember]
    public string ExceptionType { get; private set; }
}
```

For non-Ajax request and if the exception type specific exception policy has a None `PostHandlingAction`, the message of the exception handled will be set to the current model state. And then the current action specific handle-error-action will be invoked if it exists. As demonstrated in the getting-started application, the following `NCS.IConnect.Web.Mvc.ExceptionHandling.HandleErrorActionAttribute` is used to specify the name of the handle-error-action.

```
[AttributeUsage( AttributeTargets.Method)]
public class HandleErrorActionAttribute: Attribute
{
    public string HandleErrorAction { get; }
    public HandleErrorActionAttribute(string handleErrorAction);
}
```

If the handle-error-action is not specified using the `HandleErrorActionAttribute` or the specified handle-error-action does not exist, or the exception `PostHandlingAction` of the exception type specific policy is not None, the appropriate error view will be selected to report the error. We can control the view specific to the exception type by the following configuration.

```
<configuration>
    <section name="iConnect.web.mvc"
             type="NCS.IConnect.Web.Mvc.Configuration.MvcSettings,
                   NCS.IConnect.Web.Mvc"/>
  </configSections>
  <iConnect.web.mvc>
    <errorViews>
      <add name = "exception type 1"
           exception="Exception Type1"
           errorView="Error View Name 1"/>
     <add name = "exception type 2"
           exception="Exception Type2"
           errorView="Error View Name 2"/>
     ...
    </errorViews>
  </iConnect.web.mvc>
</configuration>
```

If the thrown exception whose type is not configured, the view named *Error* will be used. For all of the error views, they can be a weak one without model, or be a strongly typed view with the model type `NCS.IConnect.Web.Mvc.ExceptionHandling.ExtendedHandleErrorInfo`. As the following code snippet illustrates, `ExtendedHandleErrorInfo` is child class of `HandleErrorInfo`, and an extra `ErrorMessage` property is defined.

```
public class ExtendedHandleErrorInfo : HandleErrorInfo
{
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public string ErrorMessage{get;}
    public ExtendedHandleErrorInfo(Exception exception, string controllerName,
        string actionName, string errorMessage = "");
}
```

27.4.5. Get Error Information From ViewData

In the scenario handle-error-action is invoked to generate response, the `ExtendedhandlerErrorInfo` object which holds the final error information will be stored current ViewData. In general, the ExtendedhandlerErrorInfo will be extracted in the view to format a friendly error message displayed. The following related extension methods against `ViewDataDictionary` are defined to make it easy to set error into *ViewData* and to get error from *ViewData*. The method `ContainExceptionInfo` is used to determine whether the error is included in the given *ViewData*.

```
public static class ViewDataExtensions
{
    public static void SetHandleErrorInfo(this ViewDataDictionary viewData, Exception exception,
string message, string controllerName, string actionName);
    public static ExtendedHandleErrorInfo GetHandleErrorInfo(this ViewDataDictionary viewData);
    public static bool ContainExceptionInfo(this ViewDataDictionary viewData);
}
```

27.4.6. `ErrorMessageExceptionHandler`

We create a new custom exception handler named `ErrorMessageExceptionHandler` to customize the error message. The error message is based on literal template, and the placeholder and replacement rule is consistent with `ReplaceHandler` and `WrapHandler`. There are totally three kinds of manner to define the error message template, literal text, resource and messaging. The literal text takes lowest priority, and the priority for messaging based on is highest. The following code snippet illustrates this exception handler's configuration.

```
<configuration>
  <exceptionHandling>
    <exceptionPolicies>
      <add name="myPolicy">
        <exceptionTypes>
          <add ...>
            <exceptionHandlers>
              <add name="ErrorMessageSettingHandler"
                   type="NCS.IConnect.Web.Mvc.ExceptionHandling.ErrorMessageExceptionHandler,
                         NCS.IConnect.Web.Mvc"
                   template="..."❶
                   templateResourceType = "..."❷
                   templateMessageCategory = "..."❸
                   formatterType = "..."❹/>
            </exceptionHandlers>
        </exceptionTypes>
      </add>
    </exceptionPolicies>
  </exceptionHandling>
</configuration>
```

❶ The literal error message template. It is also the name of the error message template resource or id of error message template based message entry.

❷ The type of the error message template resource.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❸ The category of error message template based message entry.

❹ The type of the `ExceptionFormatter` used to format error message. The default type is `TextExceptionFormatter`.

27.5. Validation Enhancement

The model validation in ASP.NET MVC enhanced in *iConnect.NET*. There are two basic features, rule based validation and *Messaging* based error message definition. A new set of validators are provided to support these two features.

27.5.1. Getting Started

In this getting-started demonstration web application, we guide you how to apply *iConnect.NET* specific validator attributes with the error message is provided by Messaging, and how to perform rule based validation.

Step I Program Model, View and Controller

We first create a new empty web application using ASP.NET MVC project template, and add the assembly reference to the *NCS.IConnect.Web.Mvc.dll*, which is located in the sub folder *\bin* of the installation directory of *iConnect.NET*. In the *Models* directory, the following Person class, which is used as the model type, is created. As the following code snippet illustrates, the `DisplayTextAttribute` is applied to the three properties to customize the display name. The MessageId property, which represents the ID of the corresponding message entry, is properly specified.

```
public class Person
{
    [DisplayText(MessageId = "FirstName")]
    [RequiredValidator(ErrorMessageId = "MandatoryField")]
    public string FirstName { get; set; }

    [DisplayText(MessageId = "LastName")]
    [RequiredValidator(ErrorMessageId = "MandatoryField")]
    public string LastName { get; set; }

    [DisplayText(MessageId = "Age")]
    [RangeValidator(12, 18, RuleName = "Rule1", ErrorMessageId = "MustBeInRange")]
    [RangeValidator(18, 25, RuleName = "Rule2", ErrorMessageId = "MustBeInRange")]
    public int Age { get; set; }
}
```

The `RequiredValidatorAttribute` which is similar to `RequiredAttribute`, is used for mandatory field validation is applied to the first two properties. The MessageId of this validator attribute represents the ID of the error message entry. The Age property is decorated with two `RangeValidatorAttribute` with different RuleName (the name of the validation rule). The two validation rule specific `RangeValidatorAttribute` separately ensure the age must be in the range of 12~18 and 18~25.

In the *Controllers* directory, we create the default `HomeController`. As illustrated in the following code snippet, there are two set of action methods defined, Rule1 and Rule2. As the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

names suggest, the two set of actions are specific to the two different validation rule. Which validation rule should be applied is determined by the `ValidationRuleAttribute` applied to the action method. In the four action methods, the view named "Person" is rendered.

```
public class HomeController : BaseController
{
    public HomeController()
        : base(true, false)
    { }

    [ValidationRule("Rule1")]
    public ActionResult Rule1()
    {
        return View("Person",new Person());
    }
    [HttpPost]
    [ValidationRule("Rule1")]
    public ActionResult Rule1(Person person)
    {
        return View("Person", person);
    }

    [ValidationRule("Rule2")]
    public ActionResult Rule2()
    {
        return View("Person", new Person());
    }
    [HttpPost]
    [ValidationRule("Rule2")]
    public ActionResult Rule2(Person person)
    {
        return View("Person", person);
    }
}
```

The following code snippet represents the definition of view of Person, which is a strongly type view using Person as its model type.

```
@model MvcApp.Models.Person
@using (Html.BeginForm())
{
    @Html.EditorForModel()
    <input type="submit" value="Save" />
}
```

Step II Configure Messaging

The model specific display name and validation error message are both provided by *iConnect.NET* Messaging. In the root directory of the web project, we create an XML file used as the storage of message entries. We name it "Messages.xml", and the following message entries are defined.

```
<iConnect.messaging.messages>
  <message category="Validation" id="MandatoryField" severity="error"
      value="The {displayname} is mandatory " />
  <message category="Validation" id="MustBeInRange" severity="error"
      value="The {displayname} must be between {Minimum} and {Maximum}!" />

  <message category="DisplayName" id="FirstName" severity="Information" value="First name"/>
  <message category="DisplayName" id="LastName" severity="Information" value="Last name"/>
  <message category="DisplayName" id="Age" severity="Information" value="Age"/>
</iConnect.messaging.messages>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

The following code snippet is then added in the *web.config* to be used as the *Messaging* configuration.

```
<configuration>
  <configSections>
    <section name="cachingConfiguration"
             type="Microsoft.Practices.EnterpriseLibrary.Caching.Configuration.CacheManagerSettings,
                   Microsoft.Practices.EnterpriseLibrary.Caching" />
    <section name="iConnect.messaging"
             type="NCS.IConnect.Messaging.Configuration.MessageManagerSettings,
             NCS.IConnect.Messaging" />
  </configSections>
  <iConnect.messaging>
    <messageProviders>
      <add name="xmlProvider"
           type="NCS.IConnect.Messaging.XmlMessageProvider,
                 NCS.IConnect.Messaging"
           path="~\Messages.xml"/>
    </messageProviders>
  </iConnect.messaging>
  <cachingConfiguration defaultCacheManager="DefaultCacheManager">
    <backingStores>
      <add name="inMemory"

type="Microsoft.Practices.EnterpriseLibrary.Caching.BackingStoreImplementations.NullBackingStore,
                 Microsoft.Practices.EnterpriseLibrary.Caching" />
    </backingStores>
    <cacheManagers>
      <add name="DefaultCacheManager"
           expirationPollFrequencyInSeconds="60"
           maximumElementsInCacheBeforeScavenging="1000"
           numberToRemoveWhenScavenging="10"
           backingStoreName="inMemory"
           type="Microsoft.Practices.EnterpriseLibrary.Caching.CacheManager,
                 Microsoft.Practices.EnterpriseLibrary.Caching" />
    </cacheManagers>
  </cachingConfiguration>
  ...
</configuration>
```

Step III Register ModelValidatorProvider

iConnect.NET specific model validator is applied to ASP.NET MVC validation system by a custom `ModelValidatorProvider`, so we need to register it before starting the application. In the *Global.asax*, such a custom `FilterableDataAnnotationsModelValidatorProvider` is registered accordingly.

```
public class MvcApplication : System.Web.HttpApplication
{
    //...
    protected void Application_Start()
    {
        //...
        var provider = ModelValidatorProviders.Providers
            .OfType<DataAnnotationsModelValidatorProvider>().FirstOrDefault();
        if (null != provider)
        {
            ModelValidatorProviders.Providers.Remove(provider);
        }
        ModelValidatorProviders.Providers.Add(
            new FilterableDataAnnotationsModelValidatorProvider());
    }
}
```

Step IV Run the demonstration application

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

We run the application and navigate to the address targeting to the Rule1 (*/Home/Rule1*) and Rule2(*Home/Rule2*) action. The empty `Person` object will be rendered in the web page in editor mode. We directly press the "*Save*" button to submit the form, validation error validation will be shown as the following figure. We can see the appropriate error messages, which are defined in *Messaging* component, are displayed. For the Age property decorated with two different validation rule based `RangeValidatorAttribute`, the expected one will be selected based on the current action, and this can be proved by the different error message specific to the Age.

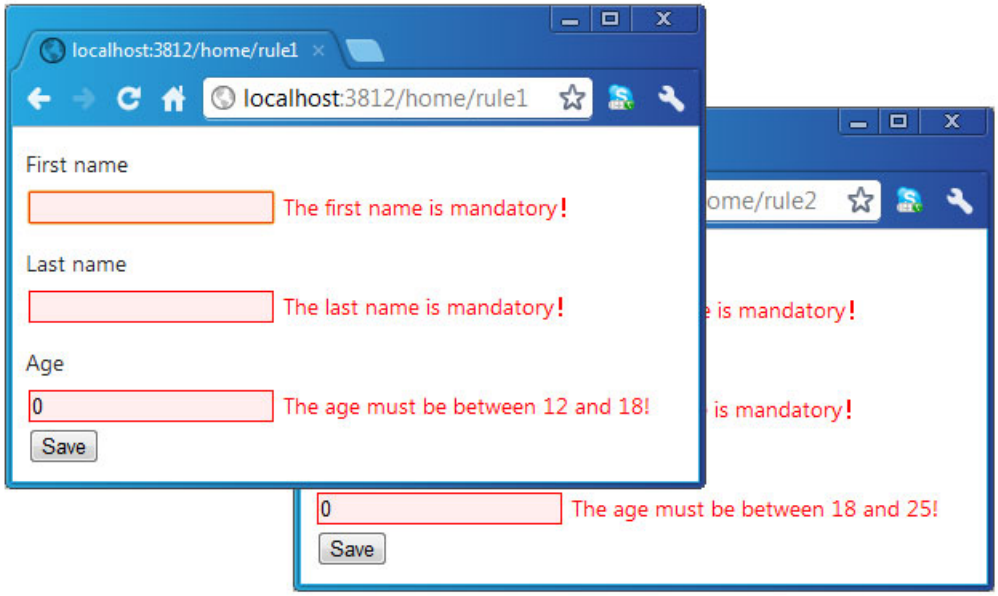


Figure 27.4. Validation Error Message

27.5.2. `BaseController`

We must make our controller class inherit from the aforementioned `BaseController` if we need to gain the enhanced validation support. As illustrated in the following code snippet, there is an EnableValidationExtension property indicating whether to support enhanced validation. This property is initialized by the *`enableExceptionHandlingExtension`* parameter of the constructor. This parameter is by default true.

```
public class BaseController : Controller
{
    //...
    public BaseController(bool enableValidationExtension = true,
        bool enableExceptionHandlingExtension = true);
    public bool EnableValidationExtension {get;}
}
```

27.5.3. ValidatorAttribute

Attribute annotation based validation is also used in the enhanced validation system. All of the *iConnect.NET* specific validator attributes inherits the abstract

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`NCS.IConnect.Web.Mvc.Validation.ValidatorAttribute` class. As illustrated by the following code snippet, `ValidatorAttribute` is a child class of `ValidationAttribute`.

```
[AttributeUsage(AttributeTargets.Field | AttributeTargets.Property
    | AttributeTargets.Class, AllowMultiple=true)]
public abstract class ValidatorAttribute : ValidationAttribute
{
    public string ErrorMessageCategory {get; set;}❶
    public string ErrorMessageId {get; set;}❷

    public string RuleName  {get; set;}❸

    public string Tag { get; set; }❹
    public string TagMessageCategory  {get; set;}❺
    public string TagMessageId {get; set;}❻
    public Type TagResourceType { get; set; }❼
}
```

❶ The category of error message entry. The default value is "*".

❷ The id of the error message entry.

❸ The name the validation policy.

❹ The name of the tag.

❺ The category of tag based message entry. The default value is "*".

❻ The id of the tag based message entry.

❼ The type of the tag based resource.

27.5.3.1. Error Message Template

The template based error message definition is used in the enhanced validation system. However, considering readability, we adopt text based placeholder instead of sequence number based one. For example, the following are two range validation based message entry, the *`{DisplayName}`* and *`{displayname}`* are placeholder for display name of the target model. When perform text replacement, *`{DisplayName}`* and *`{displayname}`* will be replaced with the real display name separately in normal case and lower case.

```
The {DisplayName} must be between {Minimum} and {Maximum}!
The {displayname} must be between {Minimum} and {Maximum}!
```

The Tag property of the `ValidatorAttribute` corresponds to the predefined placeholder "*{Tag}*". You can define such a *{Tag}* in error message template, and it will be replaced with the Tag property of the `ValidatorAttribute`. The following three kinds of template definitions are supported for both error message and tag.

- Literal text represented by ErrorMessage (defined in the `ValidationAttribute`) and Tag. This option takes the lowest priority.

- Resource. The ErrorMessage and ErrorMessageReourceType property will be considered as the name and type of error message resource item. As with resource based Tag, the Tag and TagResourceType are similar two counterparts.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Messaging. The ErrorMessageId/ErrorMessageCategory and TagMessageId/TagMessageCategory represent the id and category of the error message and tag based message entry.

27.5.3.2. `RequiredValidatorAttribute`

This is `RequiredAttribute`'s counterpart in `iConnect.NET` specific validator attributes, and it is used for mandatory field validation. As the following code snippet illustrates, `RequiredValidatorAttribute` has the same property definition as `RequiredAttribute`.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field
    | AttributeTargets.Property, AllowMultiple=true)]
public class RequiredValidatorAttribute : ValidatorAttribute, IClientValidatable
{
    public RequiredValidatorAttribute();
    public virtual IEnumerable<ModelClientValidationRule> GetClientValidationRules(
        ModelMetadata metadata, ControllerContext context);
    public override bool IsValid(object value);

    public bool AllowEmptyStrings { get; set; }
}
```

27.5.3.3. `RegularExpressionValidatorAttribute`

As its name suggests, `RegularExpressionValidatorAttribute` is used to ensure the data field of string type must match a given text pattern defined as regular expression. Its counterpart in data annotation validation attribute set is `RegularExpressionAttribute`. As the following code snippet illustrates, the property Pattern represents such a literal regular expression, which is initialized in the constructor.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field | AttributeTargets.Property,
AllowMultiple=true)]
public class RegularExpressionValidatorAttribute : ValidatorAttribute, IClientValidatable
{
    public RegularExpressionValidatorAttribute(string pattern);
    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
         ModelMetadata metadata, ControllerContext context);
    public override bool IsValid(object value);
    protected override string ReplacePlaceholders(string errorMessageString);

    public string Pattern { get; }
}
```

The `RegularExpressionValidatorAttribute` specific error message template can include a *{Pattern}* placeholder, which will be replaced with the Pattern property value.

27.5.3.4. `StringLengthValidatorAttribute`

`StringLengthValidatorAttribute` is the counterpart of `StringLengthAttribute` in *iConnect.NET* validator attribute set. When we need to ensure the length of the literal data field must be between a given range, we can apply this validator attribute. As the following code snippet illustrates, we can specify the maximum length in the constructor and specify the minimum length by manually evaluate the MinimumLength property.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field
    | AttributeTargets.Property, AllowMultiple=true)]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public class StringLengthValidatorAttribute : ValidatorAttribute, IClientValidatable
{

    public StringLengthValidatorAttribute(int maximumLength);
    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
        ModelMetadata metadata, ControllerContext context);
    public override bool IsValid(object value);
    protected override string ReplacePlaceholders(string errorMessageString);

    public int MaximumLength { get; }
    public int MinimumLength { get; set; }
}
```

When define the error message template, we can use the predefined placeholder *{MaximumLength}* and *{MinimumLength}* to represent the upper and lower bound of the data field's length, and they will be replaced with the MaximumLength and MinimumLength property value accordingly.

27.5.3.5. `RangeValidatorAttribute`

We can use the `RangeValidatorAttribute` to validate numeric data field and ensure their value must be in the give range. The counterpart in data annotation validation attributes is `RangeAttribute`. As the following code snippet illustrates, the upper and lower bound of the data range are represented by the Maximum and Minimum property, which both are initialized in the constructor.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field
    | AttributeTargets.Property, AllowMultiple=true)]
public class RangeValidatorAttribute : ValidatorAttribute, IClientValidatable
{
    public RangeValidatorAttribute(double minimum, double maximum);
    public RangeValidatorAttribute(int minimum, int maximum);
    public RangeValidatorAttribute(Type type, string minimum, string maximum);
    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
       ModelMetadata metadata, ControllerContext context);
    public override bool IsValid(object value);
    protected override string ReplacePlaceholders(string errorMessageString);

    public object Maximum { get; }
    public object Minimum { get; }
    public Type OperandType { get; }
}
```

The `RangeValidatorAttribute` specific error message template can include placeholder *{Maximum}* and *{Minimum}*, which will be replaced with the corresponding Maximum and Minimum property value.

27.5.3.6. `CompareValidatorAttribute`

The `CompareValidatorAttribute` is used to validate the data filed and ensure it must have the same value with another property. The typical usage scenario is password confirmation in user registration scenario. The following code snippet represents the definition of `CompareValidatorAttribute`. The property OtherProperty represents the property to compare with the current property.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field
    | AttributeTargets.Property, AllowMultiple=false)]
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
public class CompareValidatorAttribute : ValidatorAttribute, IClientValidatable
{
    public CompareValidatorAttribute(string otherProperty);
    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
        ModelMetadata metadata, ControllerContext context);
    protected override ValidationResult IsValid(object value, ValidationContext validationContext);
    protected override string ReplacePlaceholders(string errorMessageString);

    public string OtherProperty { get; }
}
```

The `CompareValidatorAttribute` specific error message template can include *`{OtherProperty}`* placeholder, which will be replaced with the value of the OtherProperty property accordingly.

27.5.3.7. `DataTypeValidatorAttribute`

`DataTypeValidatorAttribute` wraps the `DataTypeAttribute`, and provides data type based validation. As the following code snippets illustrates, there are same property definition between `DataTypeValidatorAttribute` and `DataTypeAttribute`.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field
    | AttributeTargets.Property, AllowMultiple=true)]
public class DataTypeValidatorAttribute : ValidatorAttribute
{
    public DataTypeValidatorAttribute(DataType dataType);
    public DataTypeValidatorAttribute(string customDataType);
    public virtual string GetDataTypeName();
    public override bool IsValid(object value);
    protected override string ReplacePlaceholders(string errorMessageString);

    public string CustomDataType { get; }
    public DataType DataType { get; }
    public DisplayFormatAttribute DisplayFormat { get; }
}
```

The three properties have corresponding placeholder in the specific error message template, and they are *`{CustomDataType}`*, *`{DataType}`* and *`{DisplayFormat}`*.

27.5.3.8. `EnumDataTypeValidatorAttribute`

The `EnumDataTypeValidatorAttribute`, which is counterpart of `EnumDataTypeAttribute`, is used to validate enumeration based data field and make sure the input literal value can be normally converted to a particular enumeration value. The definition of this validator attribute is represented in the following code snippet. The `EnumType` property represents the target enumeration type, which is initialized in the constructor.

```
[AttributeUsage(AttributeTargets.Parameter | AttributeTargets.Field
    | AttributeTargets.Property, AllowMultiple=true)]
public sealed class EnumDataTypeValidatorAttribute : DataTypeValidatorAttribute
{
    public EnumDataTypeValidatorAttribute(Type enumType);
    public override bool IsValid(object value);

    public Type EnumType { get; }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

27.5.3.9. `NumericValidatorAttribute`

The `NumericValidatorAttribute` is used to validate the numeric data field to ensure the input literal value has the correct format. As the following code snippet illustrates, no input parameter needs to be specified when applying such a validator attribute.

```
public class NumericValidatorAttribute : ValidatorAttribute, IClientValidatable
{
    public NumericValidatorAttribute();
    public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
        ModelMetadata metadata, ControllerContext context);
    public override bool IsValid(object value);
}
```

27.5.4. `DisplayTextAttribute`

The `DisplayAttribute` enable us to customize the display name of a target data field, and it support two kinds of definitions, literal text and resource. In order to make it possible to leverage Messaging to provide display name, the `DisplayTextAttribute` is defined in iConnect.NET. According to its definition illustrated in the following code snippet, it implements the `IMetadataAware` interface. Using its four properties, we can specify the display name in three different manners, literal text, resource and Messaging.

```
[AttributeUsage(AttributeTargets.Field | AttributeTargets.Property, AllowMultiple=true)]
public class DisplayTextAttribute : Attribute, IMetadataAware
{
    public DisplayTextAttribute();
    public string GetName();
    public void OnMetadataCreated(ModelMetadata metadata);

    public string MessageId { get; set; }
    public string MessageCategory { get; set; }
    public string Name { get; set; }
    public Type ResourceType { get; set; }
}
```

27.5.5. ModelValidatorProvider

All of the above validator attributes are finally applied to the ASP.NET model validation system by a custom ModelValidatorProvider, `MessagingDataAnnotationsModelMetadataProvider`. As the following code snippet, `FilterableDataAnnotationsModelValidatorProvider` is a child class `DataAnnotationsModelValidatorProvider`, and the `GetValidators` method is overridden to apply iConnect.NET validator attribute set.

```
public class FilterableDataAnnotationsModelValidatorProvider
    : DataAnnotationsModelValidatorProvider
{
    public FilterableDataAnnotationsModelValidatorProvider();
    protected override IEnumerable<ModelValidator> GetValidators(ModelMetadata metadata,
        ControllerContext context, IEnumerable<Attribute> attributes);
}
```

Since `DataAnnotationsModelMetadataProvider` is by default used, it must be removed from the ModelValidatorProvider list when we register the

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

`MessagingDataAnnotationsModelMetadataProvider`. The following code snippet demonstrates the programming of `MessagingDataAnnotationsModelMetadataProvider` registration.

```
public class MvcApplication : System.Web.HttpApplication
{
    //...
    protected void Application_Start()
    {
        //...
        var provider = ModelValidatorProviders.Providers
            .OfType<DataAnnotationsModelValidatorProvider>().FirstOrDefault();
        if (null != provider)
        {
            ModelValidatorProviders.Providers.Remove(provider);
        }
        ModelValidatorProviders.Providers.Add(
            new FilterableDataAnnotationsModelValidatorProvider());
    }
}
```

In iConnect.NET validator attribute set, the `NumericValidatorAttribute` is defined to perform date format based validation specific numeric type. However, such kinds of validation will automatically applied for numeric data field, and a static error message is used. In order to make it possible to customize the error message based on the `NumericValidatorAttribute` applied, another custom ModelValidatorProvider, `FilterableClientDataTypeModelValidatorProvider`, is defined. As illustrated by the following code snippet, it is a child class of `ClientDataTypeModelValidatorProvider`, and `GetValidators` method is overridden to suppress the automatically added validation rule once the `NumericValidatorAttribute` is decorated.

```
public class FilterableClientDataTypeModelValidatorProvider
    : ClientDataTypeModelValidatorProvider
{
    public FilterableClientDataTypeModelValidatorProvider();
    public override IEnumerable<ModelValidator> GetValidators(
        ModelMetadata metadata, ControllerContext context);
}
```

Similar to `MessagingDataAnnotationsModelMetadataProvider` registration, the existing `ClientDataTypeModelValidatorProvider` must be removed when we register the `FilterableClientDataTypeModelValidatorProvider`. The following code snippet represents the programming of `FilterableClientDataTypeModelValidatorProvider` registration.

```
public class MvcApplication : System.Web.HttpApplication
{
    //...
    protected void Application_Start()
    {
        //...
        var provider = ModelValidatorProviders.Providers
            .OfType<ClientDataTypeModelValidatorProvider>().FirstOrDefault();
        if (null != provider)
        {
            ModelValidatorProviders.Providers.Remove(provider);
        }
        ModelValidatorProviders.Providers.Add(
            new FilterableClientDataTypeModelValidatorProvider());
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

27.5.6. `ValidationRuleAttribute`

The `ValidationRuleAttribute` is used to determine which validation rule should be used. As the following code snippet illustrates, the Name property represents the validation policy name specified, which is initialized in the constructor. This attribute can be applied to both controller class and action method. If action method and its controller class are both decorated with this attribute, the one applied to action method takes higher priority.

```
[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class)]
public class ValidationRuleAttribute : Attribute
{
    public ValidationRuleAttribute(string name);
    public string Name {get}
}
```

Both client and server validation are supported, and client validation policy is determined by `ValidationRuleAttribute` applied to the action method which renders the view, but server validation policy is determined by the one applied to the action method which handle the posted form or Ajax request. In order to keep consistent between the two kinds of validation, the validation rule must be the same for the two action methods. The following programming should be avoided.

```
public class HomeController: BaseController
{
    [ValidationRuleName("Rule1")]
    pubic ActionResult Index()
    {
        //...
    }
    [ValidationRuleName("Rule2")]
    [HttpPost]
    pubic ActionResult Index(Person person)
    {
        //...
    }
}
```

27.6. Localization

We achieve URL based localization, in which the culture is placed in the request URL, by the `NCS.IConnect.Web.Mvc.Localization.LocalizationModule`, which is a custom `HttpModule`. We configure the `LocalizationModule`, we just need to register a Route with a {culture} variable like this.

```
public class MvcApp : System.Web.HttpApplication
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
        routes.MapRoute(
            "Default", // Route name
            "{culture}/{controller}/{action}/{id}", // URL with parameters
            new { culture = "en-SG", controller = "Home", action = "Index",
            id = UrlParameter.Optional });
    }
    protected void Application_Start()
    {
        RegisterRoutes(RouteTable.Routes);
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    }
}
```

27.7. Authorization

The EntLib authorization is applied to the ASP.NET MVC to make it possible to achieve different granularities (Area, Controller and Action) of authorization. The declarative authorization is adopted and we just need to apply the `ActionAuthorizeAttribute` to the Controller class or Action method to explicitly enable or disable authorization. In addition, both business action and rule based authorization can be used. The `BusinessActionAttribute` and `AuthorizationRuleAttribute` can be applied to the Controller class or Action method so as we can build a mapping between Controller/Action and Business Action/ Authorization Rule. Except for these custom attribute, the application programming interface is also provided to assign/unassign role(s) to Area, Controller and Action.

27.7.1. `BusinessActionAttribute` & `AuthorizationRuleAttribute`

As we know, iConnect.NET provides two basic authorization modes, business action & authorization rule. They both can be applied to ASP.NET MVC based authorization. Because the targets to authorize in ASP.NET MVC are Area, Controller and Action, we must map them to a concrete business action code or authorization rule name. As its name suggests, `BusinessActionAttribute` is applied to Controller class or Action method to explicitly map the target controller or action to the specified business action. Similarly, the `AuthorizationRuleAttribute` is used for authorization rule based mapping. The following code snippet illustrates the definition of the two attributes.

```
[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class, Inherited=true, AllowMultiple=false)]
public class BusinessActionAttribute : Attribute
{
    public BusinessActionAttribute(string actionCode);
    public string ActionCode { get; private set; }
}

[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class, Inherited=true, AllowMultiple=false)]
public class AuthorizationRuleAttribute : Attribute
{
    public AuthorizationRuleAttribute(string ruleName);
    public string RuleName { get; private set; }
}
```

The two kinds of attribute finally determines the authorization mode. Business action authorization will be used if the `BusinessActionAttribute` is applied, and rule based authorization will be used when `AuthorizationRuleAttribute` is decorated. If the two attributes are both applied, the `BusinessActionAttribute` takes higher priority.

27.7.2. `ActionAuthorizeAttribute`

The authorization is finally achieved by the `ActionAuthorizeAttribute`. As illustrated in the following code snippet, `ActionAuthorizeAttribute` is directly derived from `AuthorizeAttribute`, and the business action & rule based authorization are implemented in

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

the three overridden methods. For unauthorized request, `AuthorizeAttribute` will directly send back a response of 403 status code. However, in most situations, we prefer to redirect the user to a predefined error page. The ErrorView and MasterName properties are used to specify the view and layout of the error page. By default, once the `ActionAuthorizeAttribute` is applied, authorization is enabled, and we can disable authorization by specifying the Supress property to True.

```
[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class, Inherited=true, AllowMultiple=true)]
public class ActionAuthorizeAttribute : AuthorizeAttribute
{
    public ActionAuthorizeAttribute();
    protected override bool AuthorizeCore(HttpContextBase httpContext);
    protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext);
    public override void OnAuthorization(AuthorizationContext filterContext);

    public string ErrorView { get; set; }❶
    public string MasterName { get; set; }❷
    public bool   Suppress { get; set; } ❸
    public AuthorizationMode AuthorizationMode { get; set; }❹
}

public enum AuthorizationMode
{
    Declarative,
    DeclarativeAndStore
}
```

❶ The name of error view.

❷ The name of error view's layout page.

❸ Switch on/off authorization. The default value is false.

❹ The authorization mode to indicate whether to perform authorization based on declarative business action or authorization rule (Declarative) only. If DeclarativeAndStore is specified, both declarative and stored business actions or authorization rules will be used. The default value is Declarative.

The ErrorView and MasterName can also be globally registered by using the static class of `ActionAuthorization`. As illustrated in the following code snippet, the ErrorView and MaterName properties are also defined in `ActionAuthorization`. In general, we need to globally register the error page's view name and layout when application starts. If ErrorView/MasterName is not explicitly specified in `ActionAuthorizeAttribute`, the globally registered ones will be used. If the globally registered ones are still missing, the `ActionAuthorizeAttribute` will directly send back response of 403 status code as `AuthorizeAttribute` does.

```
public static class ActionAuthorization
{
    //Others
    public static string ErrorView { get; set; }
    public static string MasterName { get; set; }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

27.7.3. Role Assignment

If `BusinessActionAttribute`/`AuthorizationRuleAttribute` is applied to the current Action method or Controller class, the `ActionAuthorizeAttribute` will perform authorization based on the specified business action code or authorization rule name. In the situation without the two attributes, we can directly assign the role(s) to Area, Controller or Action by using the aforementioned static class of `ActionAuthorization`. As the following code snippet illustrates, there are three set of `AddXxxInRole(s)` methods defined in `ActionAuthorization` to separately perform Area, Controller and Action based role assignment.

```
public static class ActionAuthorization
{
    //Others
    public static void AddAreaInRole(string area, string role);
    public static void AddAreaInRoles(string area, string[] roles);

    public static void AddControllerIntoRole(string controller, string role, string area = "*");
    public static void AddControllerIntoRoles(string controller, string[] roles, string area = "*");

    public static void AddActionInRole(string controller, string action, string role, string area =
"*");
    public static void AddActionInRoles(string controller, string action, string[] roles, string area
= "*");
    public static void AddActionsInRole(string controller, string[] actions, string role, string area
= "*");
    public static void AddActionsInRoles(string controller, string[] actions, string[] roles, string
area = "*");
}
```

As matter of fact, this is a special business action based authorization, in which the Area, Controller and Action will automatically map to a formatted business action code. The following lists the Area, Controller and Action specific business action code's format.

- Area: *[{AreaName}].[*].[*]*
- Controller: *[{AreaName}].[{ControllerName}].[*]*, Or [*].[{ControllerName}].[*]*
- Area: *[{AreaName}].[{ControllerName}].[{ActionName}*] Or [*].[{ControllerName}].[{ActionName}]*

Except for these role assignment based methods, there are also another methods defined in the static class `ActionAuthorization`. The `GetBusinessActionCodeForXxx` methods are used to generate the formatted business action code by the given Area, Controller and Action name. The `IsXxxInRole` method is used to check if the given role is assigned the specified Area, Controller or Action. The `RemoveXxxFromRole(s)` is used to cancel the existing role assignment.

```
public static class ActionAuthorization
{
    // Others
    public static string GetBusinessActionCodeForAction(string controller, string action, string area
= "*");
    public static string GetBusinessActionCodeForArea(string area);
    public static string GetBusinessActionCodeForController(string controller, string area = "*");

    public static bool IsAreaInRole(string area, string role);
    public static bool IsControllerInRole(string controller, string role, string area = "*");
    public static bool IsActionInRole(string controller, string action, string role, string area = "*");
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    public static void RemoveAreaFromRole(string area, string role);
    public static void RemoveAreaFromRoles(string area, string[] roles);
    public static void RemoveControllerFromRole(string controller, string role, string area = "*");
    public static void RemoveControllerFromRoles(string controller, string[] roles, string area =
"*");
    public static void RemoveActionFromRole(string controller, string action, string role, string
area = "*");
    public static void RemoveActionFromRoles(string controller, string action, string[] roles, string
area = "*");
    public static void RemoveActionsFromRole(string controller, string[] actions, string role, string
area = "*");
    public static void RemoveActionsFromRoles(string controller, string[] actions, string[] roles,
string area = "*");
}
```

27.8. FileUpload

In web application, usually we need to upload files to server, and also can see the progress report. iConnect.NET provides the features to do it. Developers don't need to care too much upload detail, just use some easy APIs or attributes to finish the task.

Here FileUpload is focused on ASP.NET MVC project mainly, it provides many useful features and as possible as simple make developers easy to use FileUpload function.

On the other hand, if you want to transfer files based on WCF service, you can look up "Data Transfer" chapter.

As the core of FileUpload, the assembly "NCS.IConnect.Web.FileUpload.dll" needs to be included in web application. The assembly provides the necessary features.

- *Without Progress Report*

  If you just need to upload file without progress report, what you need to do is little. Of course, for your ASP.NET MVC project, you still need to reference the assembly "NCS.IConnect.Web.FileUpload.dll" at first.

  - Regitster custom "ValueProviderFactory" in "Global.asax.cs" so that the file stream to upload can be bound to the specific Model.

```
using NCS.IConnect.Web.FileUpload.Mvc;

namespace FileUploadGetStarted
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            … …

            ValueProviderFactories.Factories.Add(new
UploadedFileCollectionValueProviderFactory());
        }
    }
}
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- Define your Model according to your requirement. Here, just two properties: one is to identify the file name, the other one is the file to upload.

```
public class FileToUpload
{
    public string FileName { get; set; }
    public UploadedFile FileContent { get; set; }

}
```

- Create UI to upload file and write some necessary code in View.

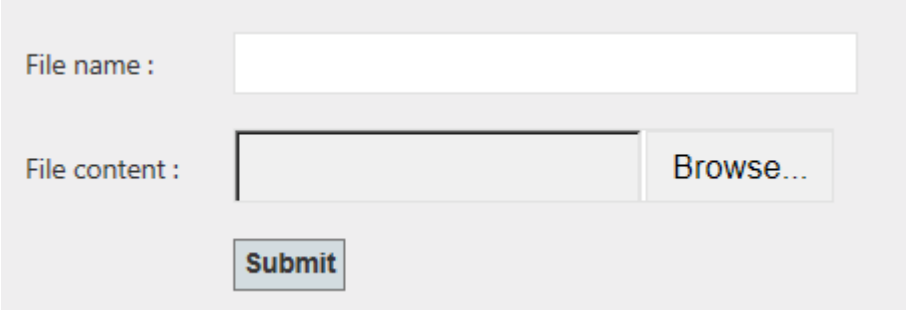


Figure 27.5. Create UI for FileUpload

It's noted that the first code line is mandatory.

```
@using (Html.BeginForm("WithoutProgressPost", "FileUpload", FormMethod.Post, new { enctype =
"multipart/form-data" }))
{
    <table>
        <tr>
            <td>File name : </td>
            <td>@Html.EditorFor(f=>f.FileName)</td>
        </tr>
        <tr>
            <td>File content : </td>
            <td>
                <input type="file" name="FileContent" id="FileContent" />
            </td>
        </tr>
        <tr>
            <td></td>
            <td><input type="submit" value="Submit" /></td>
        </tr>
    </table>
}
```

- Handle Controller to perform file upload action. Just as the following code lines. It's required to mark attribute "FileUpload" on the "Action".

```
[FileUpload(1024)]
public ActionResult WithoutProgressPost(FileToUpload file)
{
    SaveFile(file.FileContent);

    return View();
}

private void SaveFile(UploadedFile uploadedFile)
{
    if (uploadedFile == null) return;
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
    var fileToSave =
        Path.Combine(System.Web.HttpContext.Current.Request.PhysicalApplicationPath,
                        @"App_Data\uploads",
                      uploadedFile.FileName.Substring(uploadedFile.FileName.LastIndexOf(@"\")
+ 1));
    uploadedFile.SaveAs(fileToSave, true);
}
```

- That's all, then invoke your web application, you can upload file through UI.

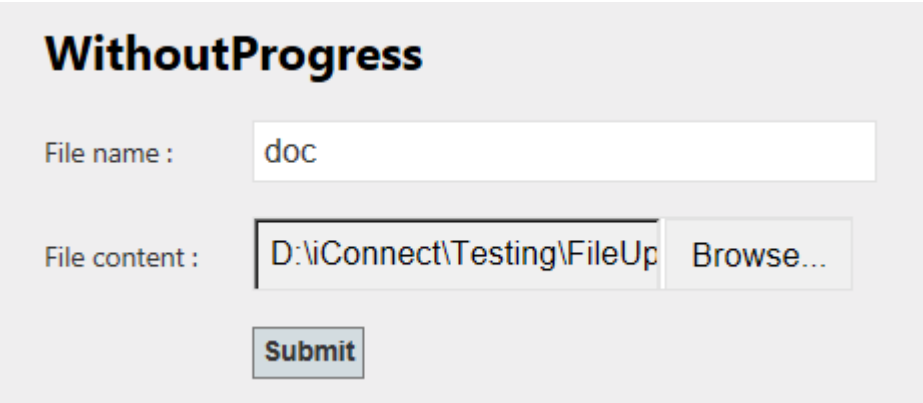


Figure 27.6. Demo without Progress Report (1)

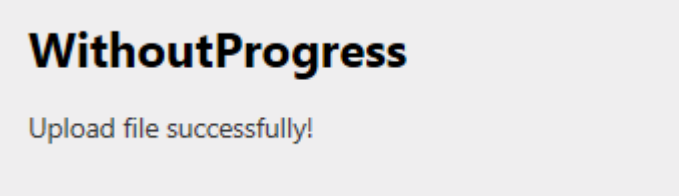


Figure 27.7. Demo without Progress Report (2)

- *With Progress Report*

Usually, if you upload large files, it needs to wait for a long time. Under the case, if there is a progress bar to display the progress, it will become a very friendly. Actually, it's a common behavior as many commercial software.

From above steps, we have implemented the function to upload file, but not any progress notification. So, we just need to find an approach to display the progress, it will be perfect.

- Register JavaScript resources in "Global.asax.cs".

```
using NCS.IConnect.Web.FileUpload.Mvc;
using NCS.IConnect.Web.FileUpload;

namespace FileUploadGetStarted
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            … …
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
            ValueProviderFactories.Factories.Add(new
UploadedFileCollectionValueProviderFactory());
            BundleTable.Bundles.Add(new FileUploadScriptBundle("~/bundle/fileUpload"));
        }
    }
}
```

- Add configuration to support progress report. It's implemented through custom HttpHandler (UploadProgressQueryHandler).

```
<system.web>
  ... ...
  <httpRuntime requireRootedSaveAsPath="true" maxRequestLength="2097151" executionTimeout="7200"/>
  <httpHandlers>
    <add verb="*" path="UploadProgressQuery.axd"
type="NCS.IConnect.Web.FileUpload.UploadProgressQueryHandler, NCS.IConnect.Web.FileUpload"/>
  </httpHandlers>

  ... ...
</system.web>

<system.webServer>
  <validation validateIntegratedModeConfiguration="false" />
  <handlers>
    <add name="UploadProgressQuery" verb="*" path="UploadProgressQuery.axd"
         type="NCS.IConnect.Web.FileUpload.UploadProgressQueryHandler,
NCS.IConnect.Web.FileUpload"/>
  </handlers>
  <modules runAllManagedModulesForAllRequests="true" />
  <security>
    <requestFiltering>
      <requestLimits maxAllowedContentLength="2000000000" ></requestLimits>
    </requestFiltering>
  </security>
</system.webServer>
```

- Modify "Action", just return Json data.

```
[FileUpload(1024)]
public ActionResult WithProgressPost(FileToUpload file)
{
    SaveFile(file.FileContent);

    return new JsonResult { Data = file };
}
```

- On "View", needs to add JavaScript to implement progress notification, as the following html/JavaScript:

```
@model FileUploadGetStarted.Models.FileToUpload

<h2>WithProgress</h2>
@Scripts.Render("~/bundle/fileUpload")❶
@using (Html.BeginForm("WithProgressPost", "FileUpload", FormMethod.Post, new { enctype =
"multipart/form-data", id = "fileUploadForm" }))
{
    <table>
        <tr>
            <td>File name : </td>
            <td>@Html.EditorFor(f=>f.FileName)</td>
        </tr>
        <tr>
            <td>File content : </td>
            <td>
```

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

```
                <input type="file" name="FileContent" id="FileContent" />
            </td>
        </tr>
        <tr>
            <td></td>
            <td><input id="fileUploadSubmit" type="submit" value="Submit" /></td>
        </tr>
    </table>

    <div id="myModal" tabindex="-1">❷
        <div>
            <div id="myPnl">
                <div id="myInfo"></div>
                <div id="result"></div>
            </div>
        </div>
    </div>
}

<script type="text/javascript">
    $("#fileUploadSubmit").click(function () {
        $("#fileUploadForm").beginUpload(
        {
            queryProgress: true,
            queryProgressUrl: '@Url.Content("~/UploadProgressQuery.axd")',❸
            dataType: "json",

            onUploadProgress : function(uploadSession) {
                showProgressInfo(uploadSession);❹
            },

            success: function (file) {❺
                var html =
                    "<br/><b>File Name: </b>" + file.FileName + "<br/>";
                if (file.FileContent != null) {
                    html += "<b>File Content : </b>" + file.FileContent.FileName + "(<font
color='red'>" +
                            Math.round((file.FileContent.ContentLength * 100 / (1024 * 1024)))
/ 100 + "MB</font>)" + "<br/>";
                }

                $("#result").html(html);
            }
        });

        function showProgressInfo(uploadSession) {
            $("#myInfo").html(
                        "<b>Upload session id: </b>" + uploadSession.uploadSessionId + "<br/>" +
                         "<b>Local time begin: </b>" + uploadSession.localTimeBegin + "<br/>" +
                         "<b>Total bytes: </b>" + uploadSession.totalBytes + "<br/>" +
                      "<b>Percentage: </b><span style='color:Red;'>" + uploadSession.percentage
+ "%</span>" + "<br/>" +
                         "<b>Seconds elapsed: </b>" + uploadSession.secondsElapsed + "<br/>" +
                         "<b>Bytes perSecond (average): </b>" +
uploadSession.bytesPerSecondAverage + "<br/>" +
                         "<b>Second left (average): </b>" + uploadSession.secondLeftAverage +
"<br/>" +
                         "<b>Bytes per second (current): </b>" +
uploadSession.bytesPerSecondCurrent + "<br/>" +
                         "<b>Second left (current): </b>" + uploadSession.secondLeftCurrent +
"<br/>" +
                         "<b>Upload state: </b>" + uploadSession.uploadState + "<br/>");
        }
    });
</script>
```

❶ Register JavaScript resources on page.

❷ Add a pannel to display progress information.

❸ Ajax request to report progress when uploading files.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

❹ Show progress information.

❺ Show upload result.

- Then, you can see upload progress.

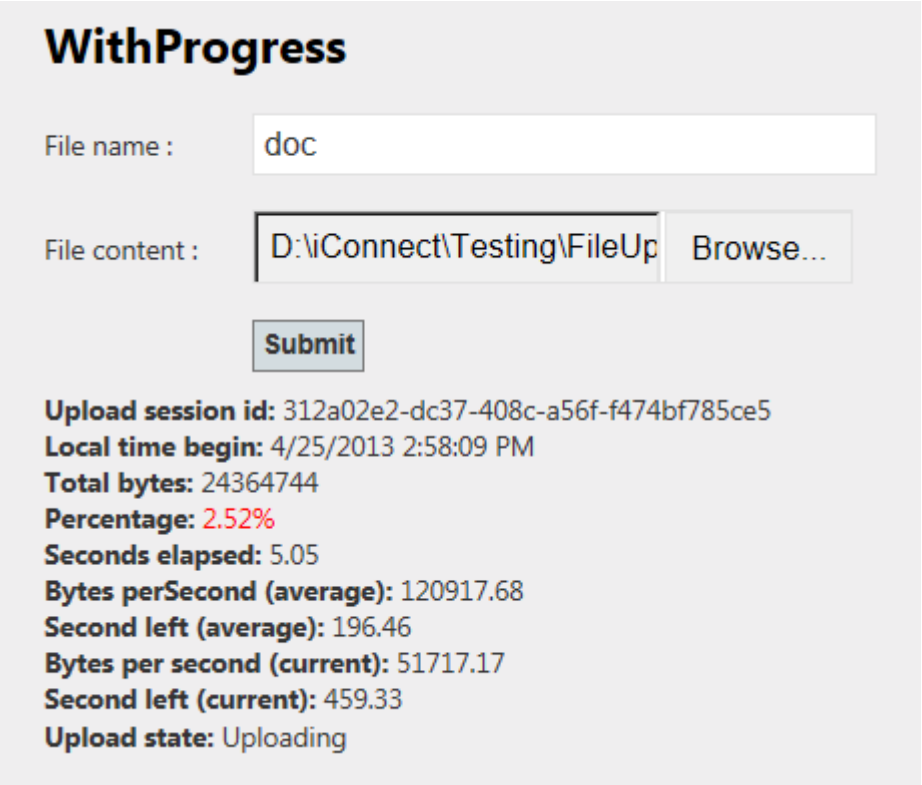


Figure 27.8. Demo with Progress Resport

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Chapter 28. Configuration Console

28.1. Introduction

The configuration tool is used to configurate iConnect related setting using the EnLibCofig tool.It makes the setting of iConnect easier by adminsitration or the related users.

It includes the following settings:

- iConnect audit trail settings
- iConnect batch job settings
- iConnect batch job service settings
- iConnect batch job service settings
- iConnect calendar settings
- iConnect code table settings
- iConnect common settings
- iConnect application context settings
- iConnect message compression settings
- iConnect data mapping settings
- iConnect data access settings
- iConnect hierarchy settings
- iConnect LMS delivery settings
- iConnect LMS manager settings
- iConnect logging settings
- iConnect instrumentation settings
- iConnect message settings
- iConnect security scheduler settings
- iConnect session management settings
- iConnect state settings

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

- iConnect string resource settings

28.2. The usage flow

Opens the editor configuration file with the tool of "EntLibConfig" or creates the new configuration file.

The tool is located on tools menu under iConnect V4.4 menu.

28.2.1. iConnect audit trail settings

It is used to configurate iConnect audit trail.

1. Clicks "Blocks" -> "iConnect Audit Trail Settings"
2. Clicks "+"in the audit trail providers, adds the related provider type and then sets the related value for the setting;
3. Clicks "+" in the encoders, adds the related encoder type and then sets the related value for the setting;
4. Clicks "+" in the table mappings, adds the table mapping and then sets the related value for the setting.

   Note: for more detail, please refer to audit trail chapter.

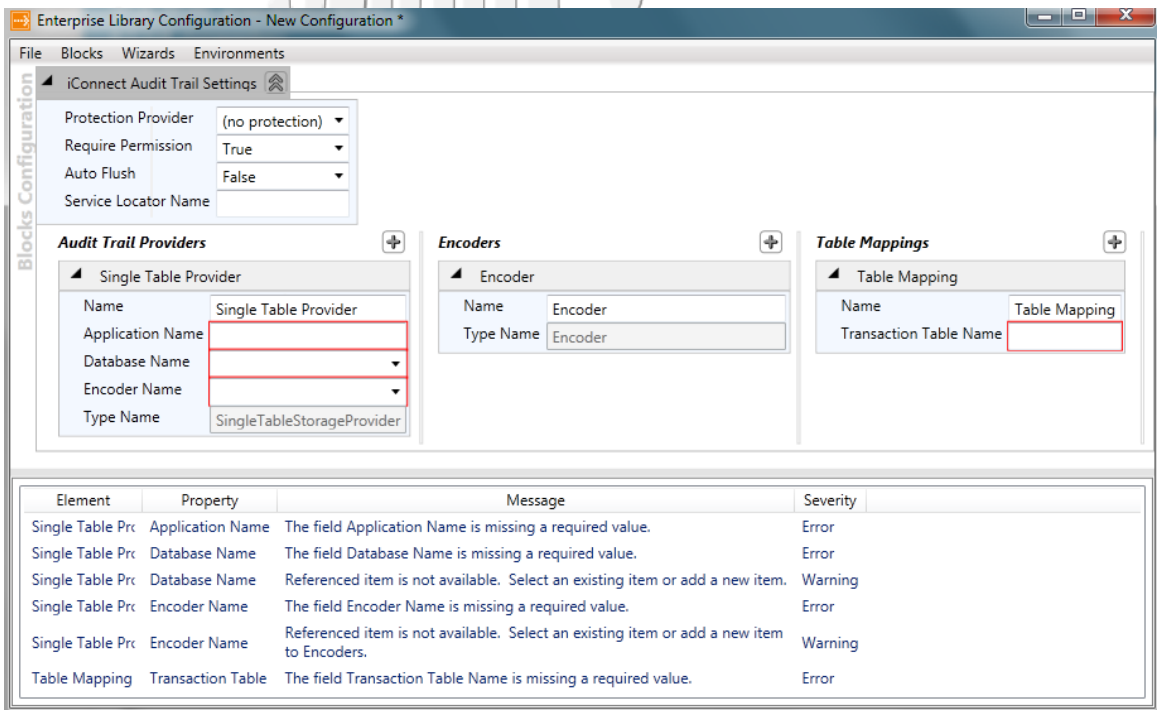


Figure 28.1. The layout of audit trail setting

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

28.2.2. iConnect batch job settings

It is used to configurate iConnect batch job.

1. Clicks "Blocks" -> "iConnect batch job settings"
2. Clicks "+" in the schedule controller providers, adds the related provider type and then sets the related value;
3. Expands the setting selection, sets the related value for Default schedule controller;

   Note: for more detail, please refer to batch job chapter.

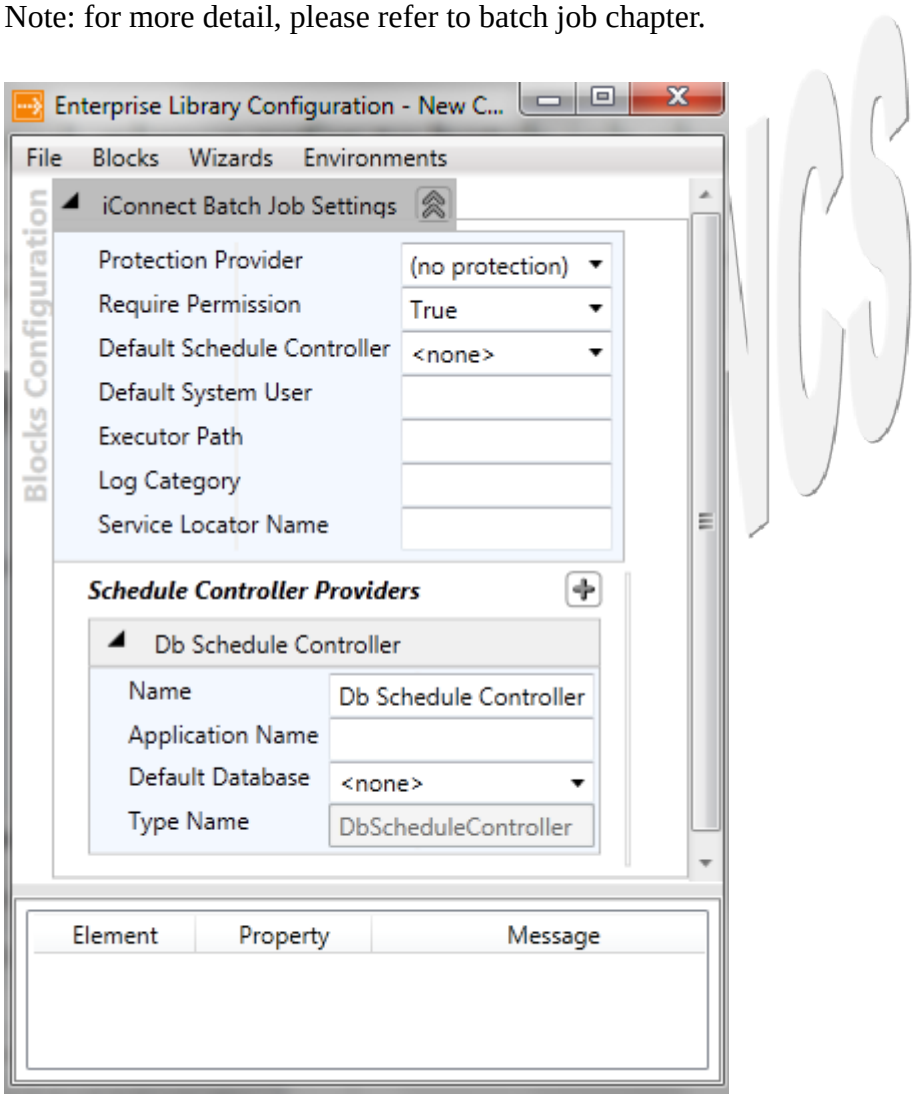


Figure 28.2. The layout of batch job setting

28.2.3. iConnect batch job service settings

1. Clicks "Blocks" -> "iConnect batch job service settings"
2. Expands the setting of iConnect batch job service and sets the related value.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

Note: for more detail, please refer to batch job chapter.

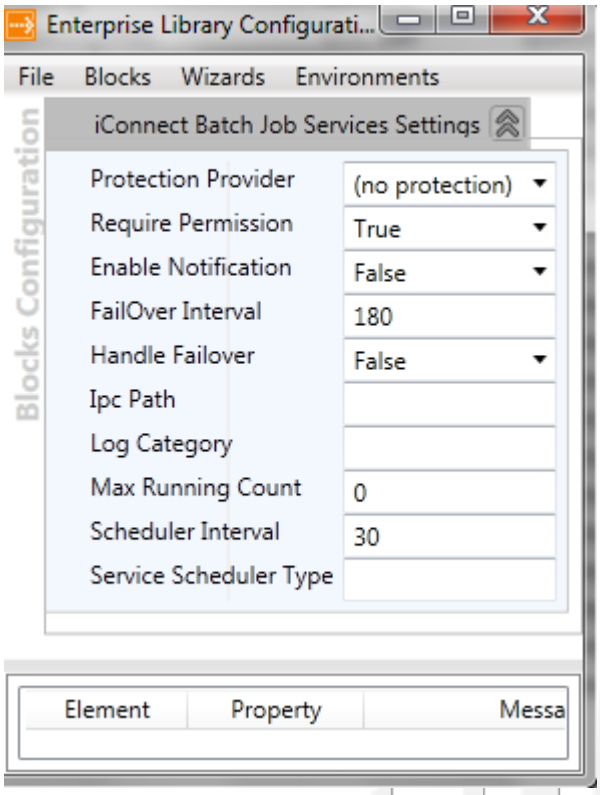


Figure 28.3. The layout of batch job service settings

28.2.4. iConnect calendar settings

1. Clicks "Blocks" -> "iConnect calendar settings"
2. Expands the selection and sets the related value for default calendar provider;
3. Clicks "+" in the storage provider providers, adds the related provider type then sets the related value for the setting;

   Note: for more detail, please refer to calendar chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

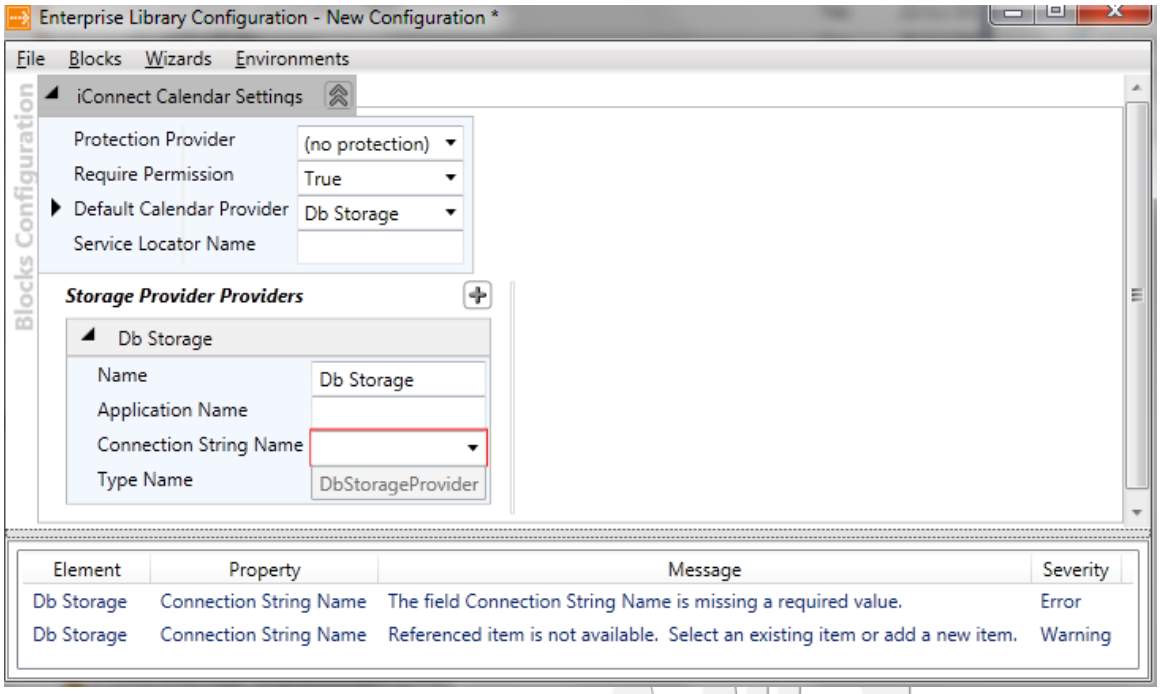


Figure 28.4. The layout of calendar

28.2.5. iConnect code table settings

1. Clicks "Blocks" -> "iConnect Code Table Settings"
2. Expands the selection of code table and then sets the related value for the setting.
3. Clicks "+" in the code table providers, adds the related code table provider and then sets the related value for the setting.
4. Clicks "+" in binding options, adds the related binding option and then sets the related value for the setting.
5. Clicks "+" in control bindings, adds the related control binding and then sets the related for the setting.
6. If system need consider cache manager, sets the related value for the setting.
7. Clicks "+" in name converters, adds the related name converter and then sets the related value for the setting.

   Note: for detail each field, please refer to code table chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

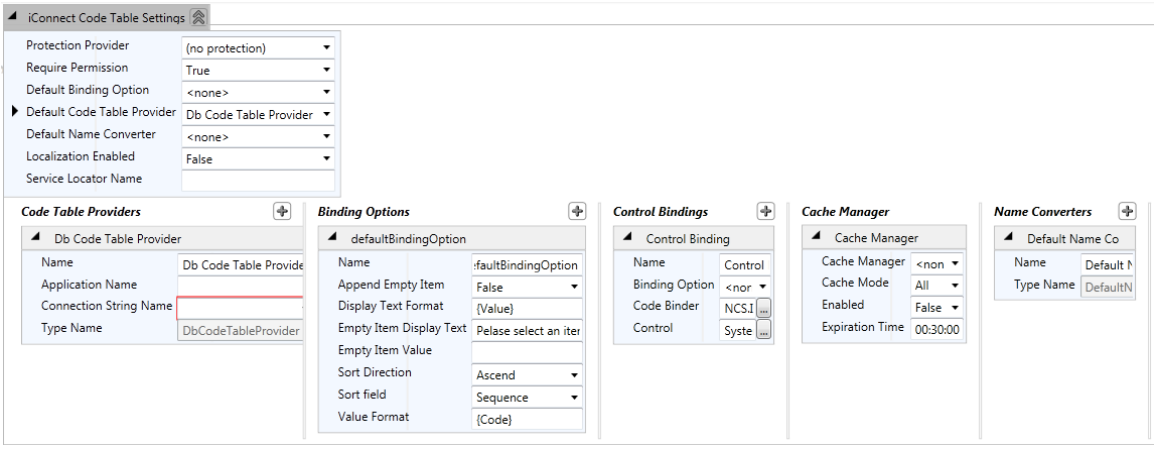


Figure 28.5. The layout of code table

28.2.6. iConnect common settings

1. Clicks "Blocks" -> "iConnect common settings"
2. Expands iConnect common settings, sets the access denied url value for the setting.

   Note: for more detail, please refer to the related chapter.

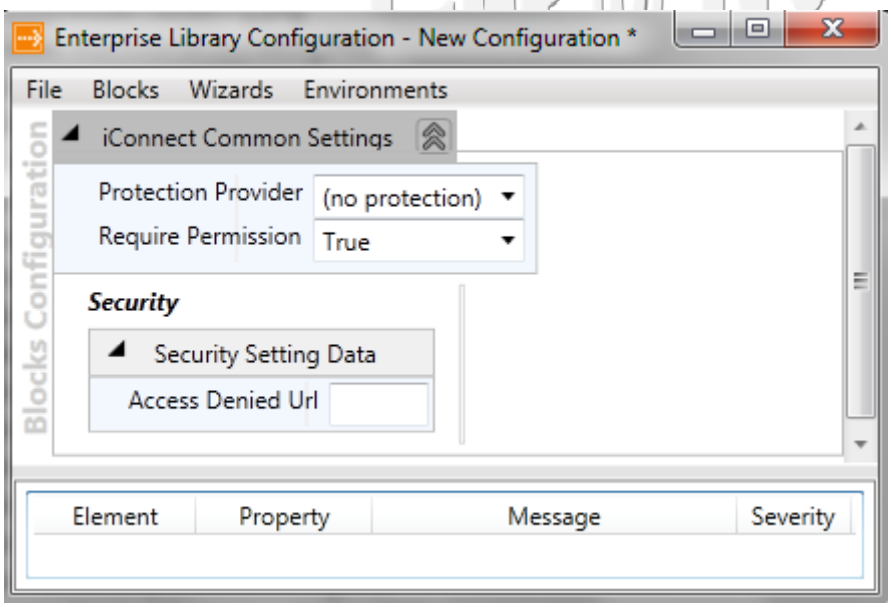


Figure 28.6. The layout of iConnect common settings

28.2.7. iConnect application context settings

1. Clicks "Blocks" -> "iConnect application context settings"
2. Expands iConnect application context settings, sets the related value for "Context attach behavior" and "Default context locator".

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

3. Clicks "+" in the Context locators, adds the related context locator provider and then sets the related value for name;

   Note: for more detail, please refer to application context chapter.

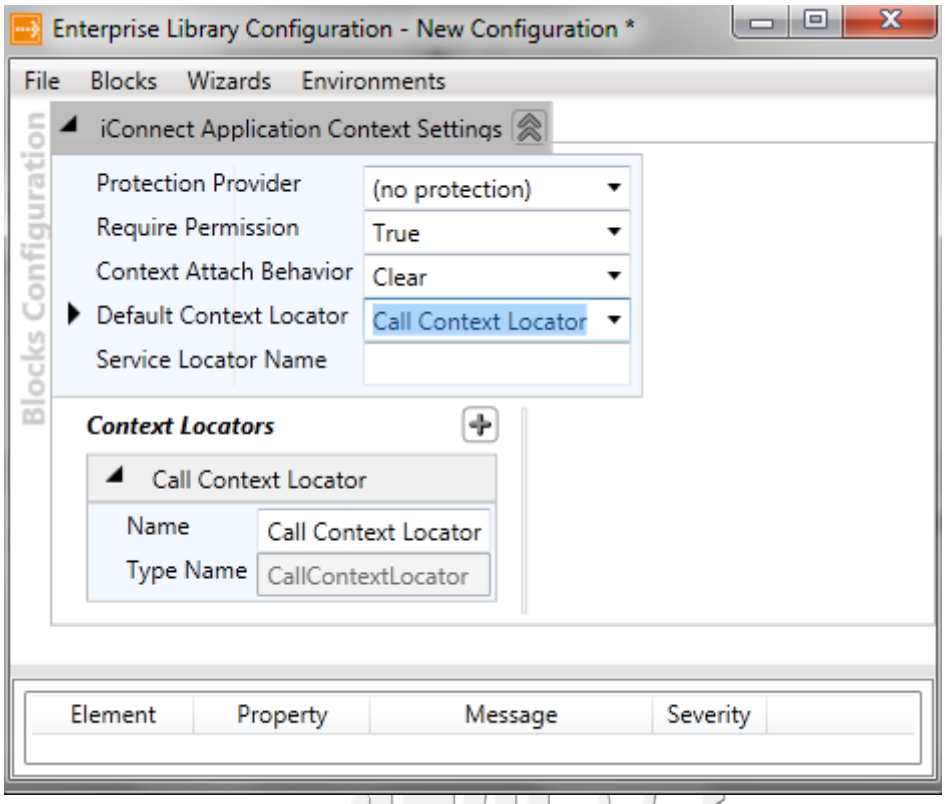


Figure 28.7. The layout of application context

28.2.8. iConnect message compression settings

1. Clicks "Blocks" -> "iConnect message compression settings"

2. Expands the setting, sets the related value for "Algorithm", "Min Message Size" and "Streaming Mode";

3. Clicks "+" in the message compressors, adds the related message compressor provider and then sets the related value for "name","Algorithm","Min Message Size" and "Streaming Mode";

4. Clicks "+" in the table mappings, adds the table mapping and then sets the related value for the setting.

   Note: for more detail, please refer to IoC chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

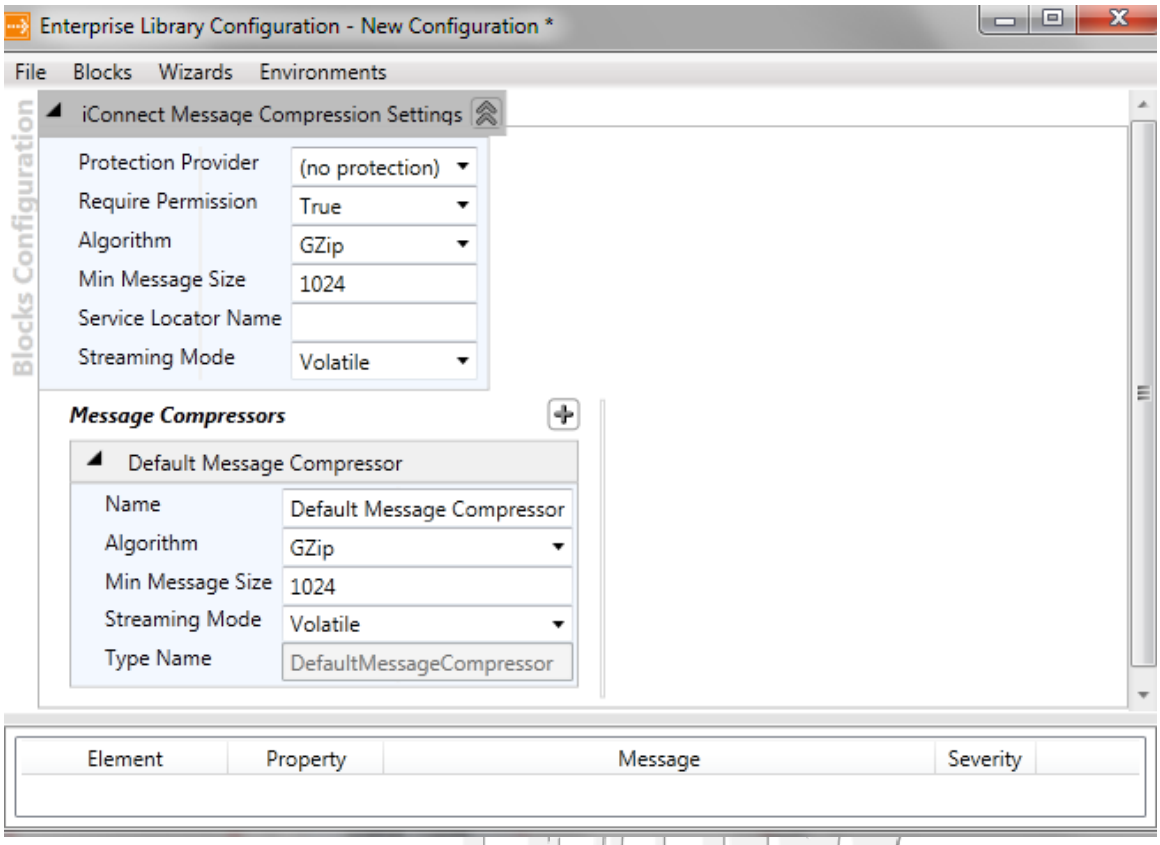


Figure 28.8. The layout of iConnect Message Compression Settings

28.2.9. iConnect data mapping settings

1. Clicks "Blocks" -> "iConnect data mapping settings"
2. Expands the iConnect data mapping settings, sets the related value for security fields;
3. Clicks "+" in the control setting, sets the related value for "Name","Binding Property" and "Control Type";
4. Clicks "+" in the source setting, sets the related value for "Name","Data Type" and "Format String".
5. Expands naming converter selection, sets the value for "Type".

   Note: for more detail, please refer to data mapping chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

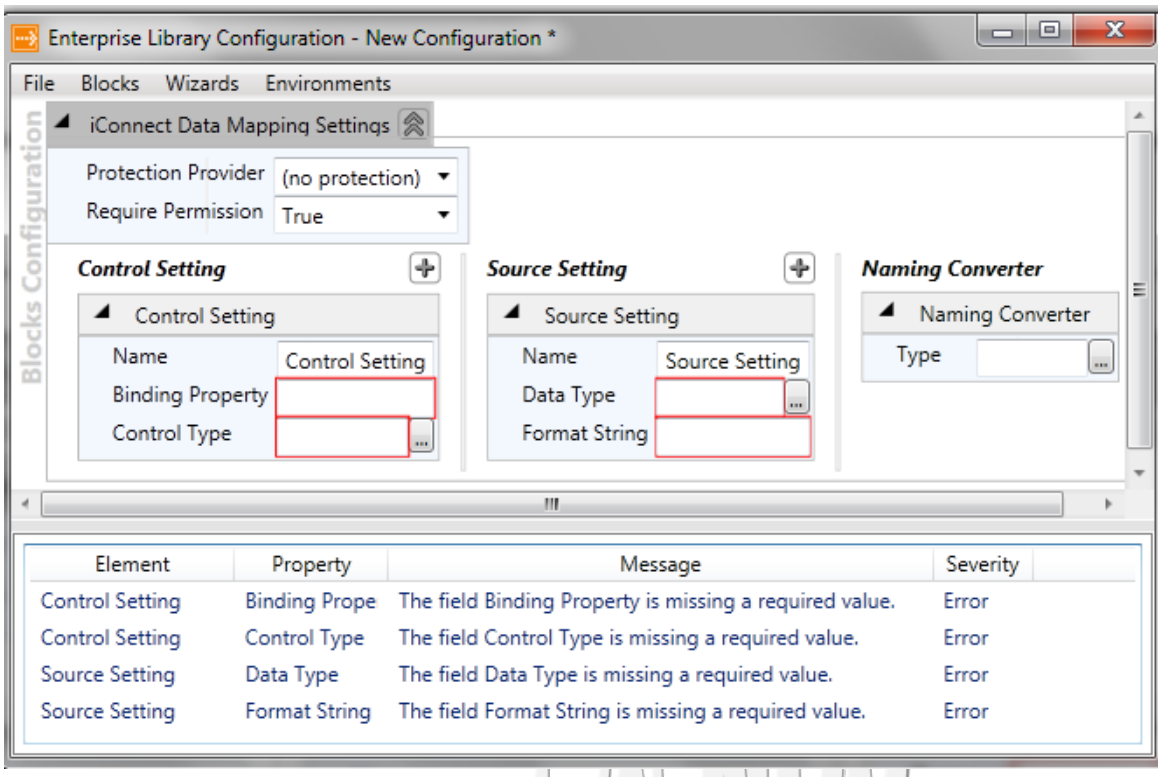


Figure 28.9. The layout of data mapping

28.2.10. iConnect data access settings

1. Clicks "Blocks" -> "iConnect data access settings"
2. Expands the setting of iConnect data access, sets the value for "command timeout";

   Clicks "+"in the commands, adds the related command data and then sets the related value for "Name" and "Format";
3. Clicks "+" in the system fields, adds the related system field and then sets the related value for "Name" and "Field Name";
4. Clicks "+" in the parameters, adds the parameter data and then sets the related value for "Name" and "Format";

   Clicks "+" in the schema mappings, adds the schema mapping and then sets the related value for "Name" and "Format";

   Note: for more detail, please refer to database helper chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

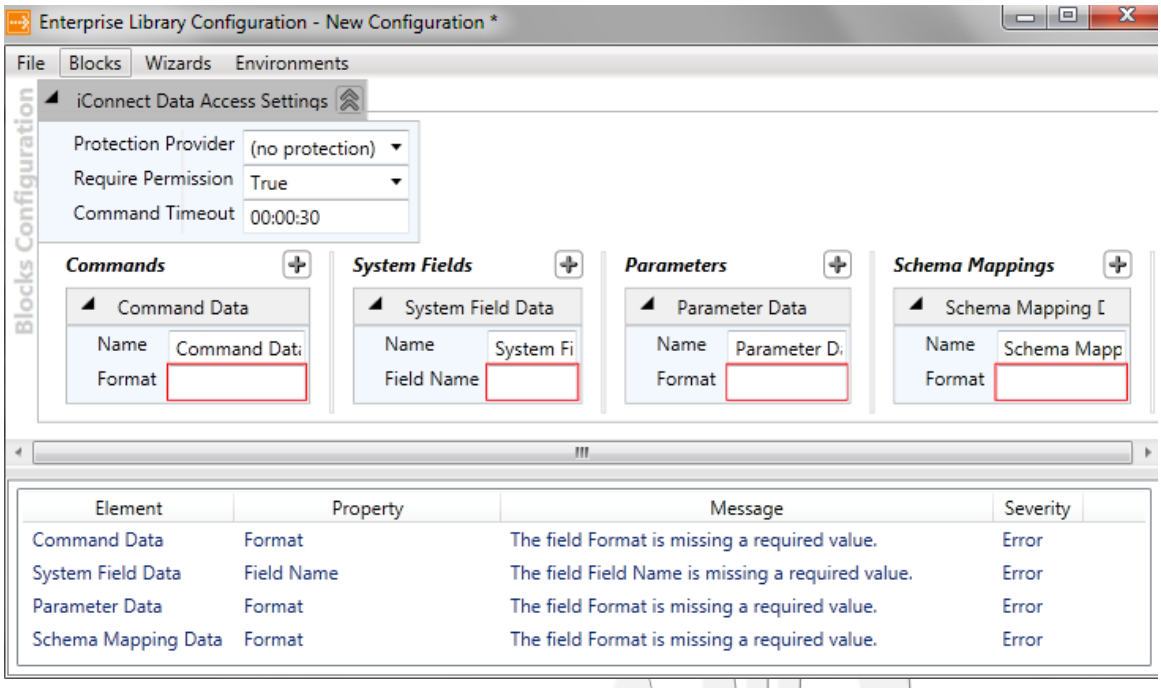


Figure 28.10. The layout of iConnect data access settings

28.2.11. iConnect hierarchy settings

1. Clicks "Blocks" -> "iConnect hierarchy settings"
2. Expands iConnect hierachy setting, sets the related value for the setting;
3. Clicks "+" in hierachy provider, adds the related hierachy procider and then sets the related value for the setting;
4. Note: for more detail, please refer to hierarchy chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

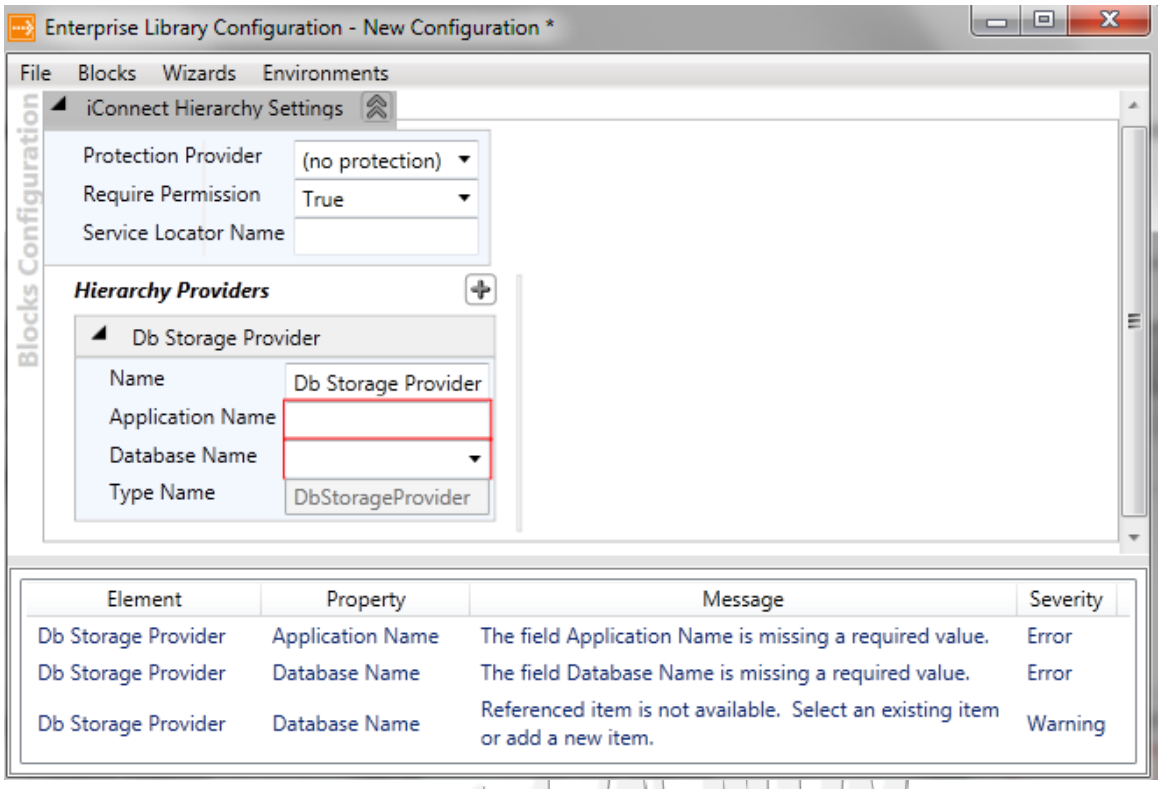


Figure 28.11. The layout of iConnect hierarchy setting

28.2.12. iConnect LMS delivery settings

1. Clicks "Blocks" -> "iConnect LMS delivery settings"
2. Expands the setting of iConnect LMS delivery, sets the value for "Default Document Storage" and other;
3. Clicks "+"in the delivery filters, adds the related delivery filter and then sets the related value for the setting;
4. Clicks "+" in the delivery listerners, adds the related delivery listener and then sets the related value for the setting;
5. Clicks "+" in the document storages, adds the document storage and then sets the related value for the setting.

   Note: for more detail, please refer to LMS chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

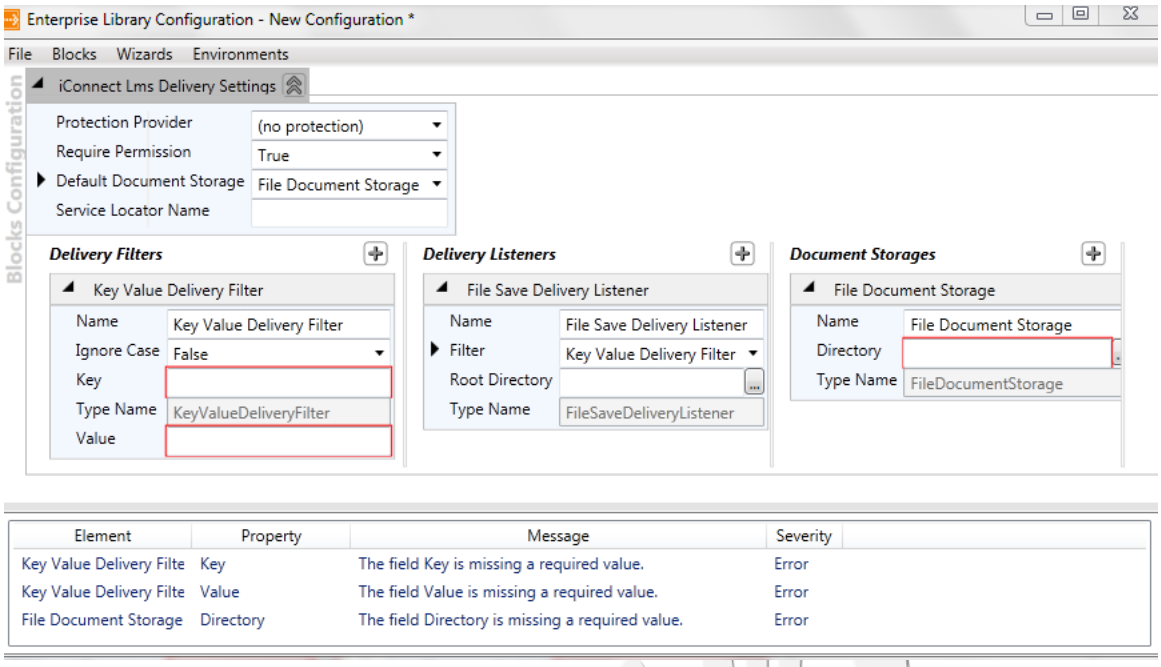


Figure 28.12. The layout of LMS delivery settings

28.2.13. iConnect LMS manager settings

1. Clicks "Blocks" -> "iConnect LMS manager settings"
2. Expands the setting of iConnect LMS manager, sets the related value for "Default document storage", "Delivery Endpoint" and "Template File Root Directory";
3. Clicks "+"in the metadata provider, adds the related metadata provider and then sets the related value for the setting;
4. Clicks "+" in the inspector filter, adds the related inspector filter and then sets the related value for the setting;
5. Clicks "+" in the document inspectors, adds the related document inspector and then sets the related value for the setting;
6. Clicks "+" in the dispatch context initializer, adds the related dispatch context initializer and then sets the related value for the setting.
7. Clicks "+" in the document storages, adds the related document storage provider and sets the related value for the setting;

   Note: for more detail, please refer to LMS chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

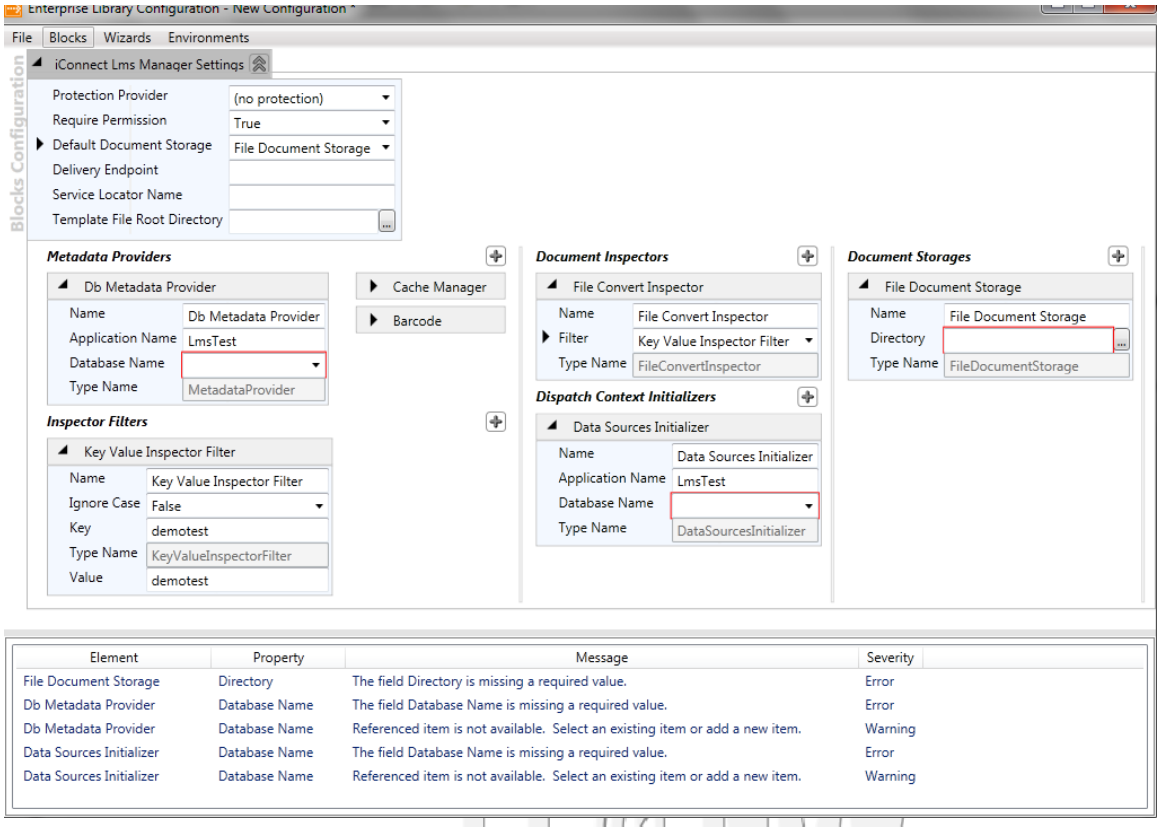


Figure 28.13. The layout of iConnect Lms Manager Settings

28.2.14. iConnect logging settings

1. Clicks "Blocks" -> "iConnect logging settings"
2. Expands the setting of iConnect Logging, sets the related value for "Default Log Viewer provider" and other;
3. Clicks "+"in the log viewer providers, adds the related provider type and then sets the related value for the setting;

   Note: for more detail, please refer to Logging chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

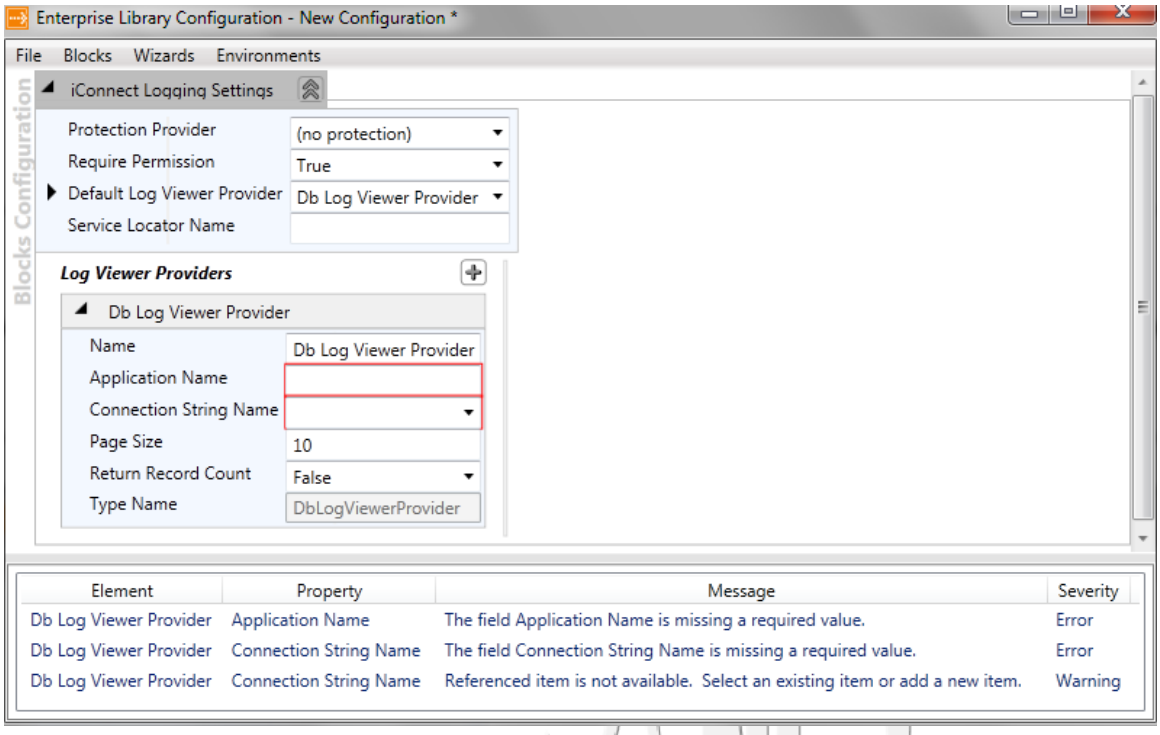


Figure 28.14. The layout of iConnect Logging settings

28.2.15. iConnect instrumentation settings

1. Clicks "Blocks" -> "iConnect instrumentation settings"
2. Expands the setting of instrumentation, sets the related value for the setting;
3. Expands the monitoring setting, sets the related value for the setting;
4. Clicks "+" in the monitoring providers, adds the related monitoring provider and sets the related value for the setting;
5. Clicks "+" in the encoders, adds the related encoder type and then sets the related value for the setting;
6. Clicks "+" in the parameter formatters, adds the parameter formatter and then sets the related value for the setting.
7. Note: for more detail, please refer to instrumentation chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

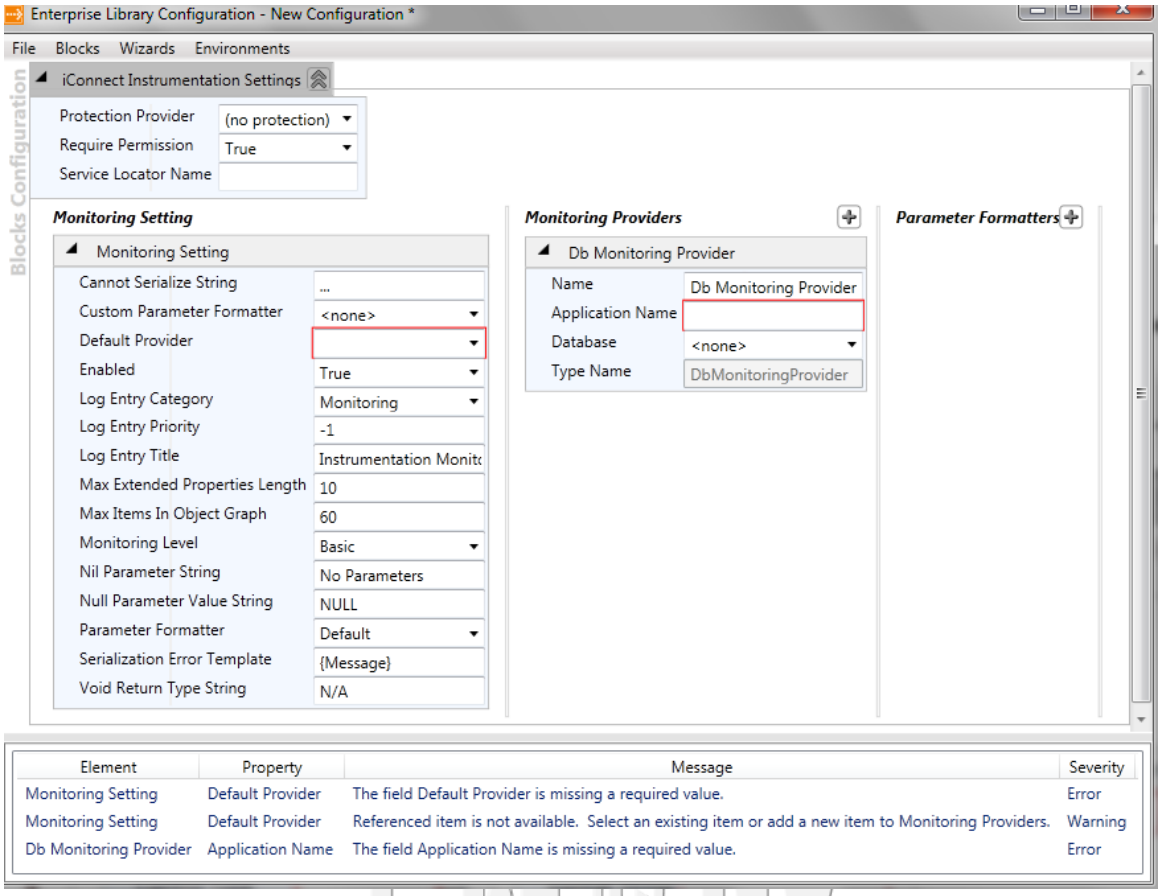


Figure 28.15. The layout of iConnect instrumentation settings

28.2.16. iConnect message settings

1. Clicks "Blocks" -> "iConnect message settings"
2. Expands the setting of iConnect message, sets the related value for "Format Specifications" and other;
3. Clicks "+"in the message providers, adds the related message provider and then sets the related value for the setting;

   Note: for more detail, please refer to message chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

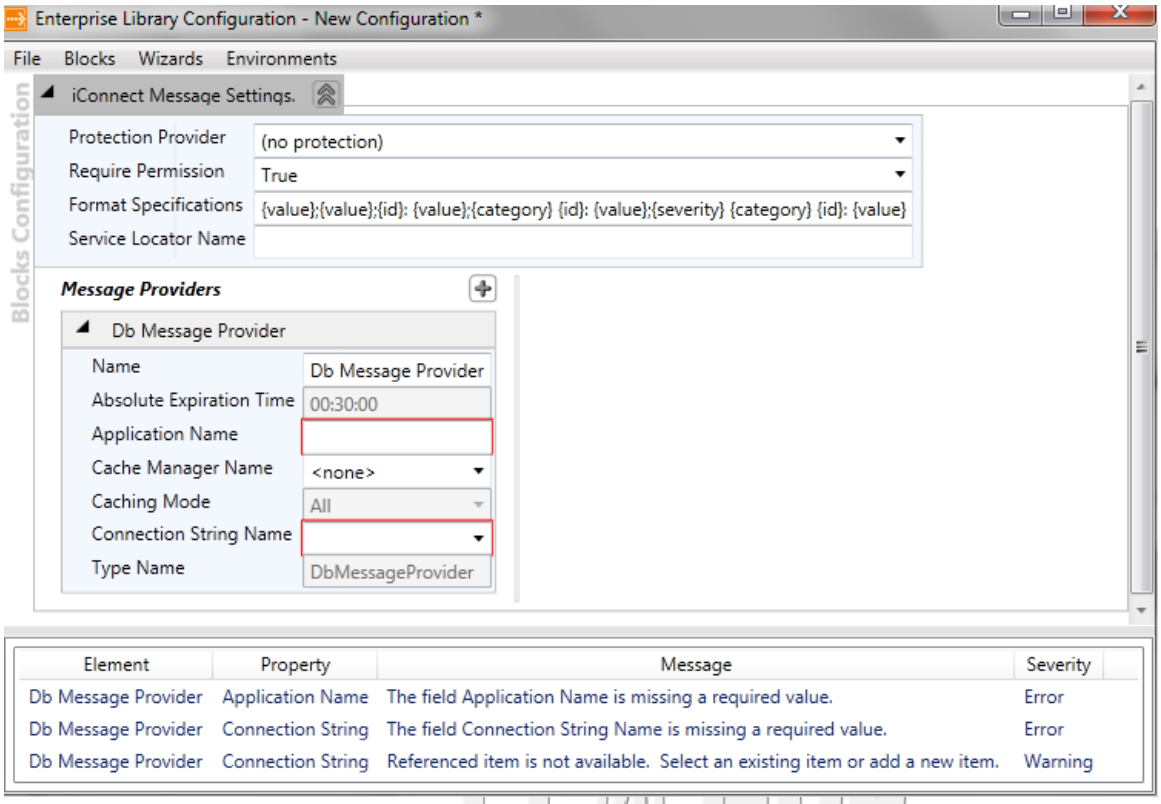


Figure 28.16. The layout of iConnect message settings

28.2.17. iConnect security scheduler settings

1. Clicks "Blocks" -> "iConnect securtiy scheduler settings"
2. Expands the setting of iConnect security scheduler setting;
3. Expands the base setting in the base setting, sets the related value for the setting;
4. Expands the scavenging setting in the scavenging setting, sets the related value for the setting;
5. Expands the suspension setting, sets the related value for the setting.

   Note: for more detail, please refer to security chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

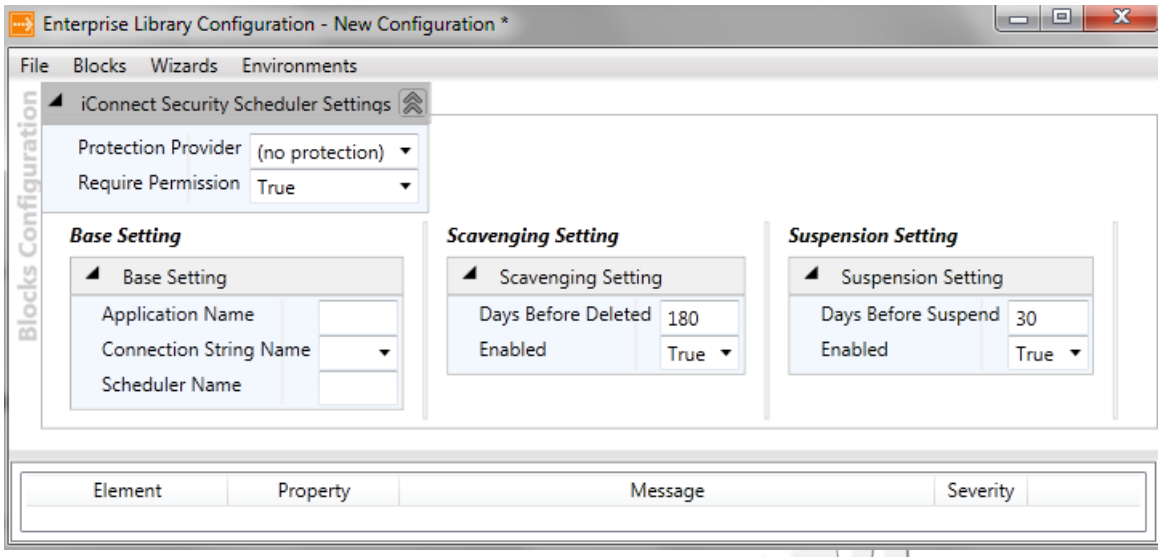


Figure 28.17. The layout of iConnect security scheduler setting

28.2.18. iConnect session management settings

1. Clicks "Blocks" -> "iConnect session management settings"
2. Expands the setting of iConnect session management, sets the related value for "Default provider", "Default Scavenger" and other.
3. Clicks "+"in the session providers, adds the related provider and then sets the related value for the setting;
4. Clicks "+" in the session scavengers, adds the related background scavenger and then sets the related value for the setting.

   Note: for more detail, please refer to session management chapter.

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

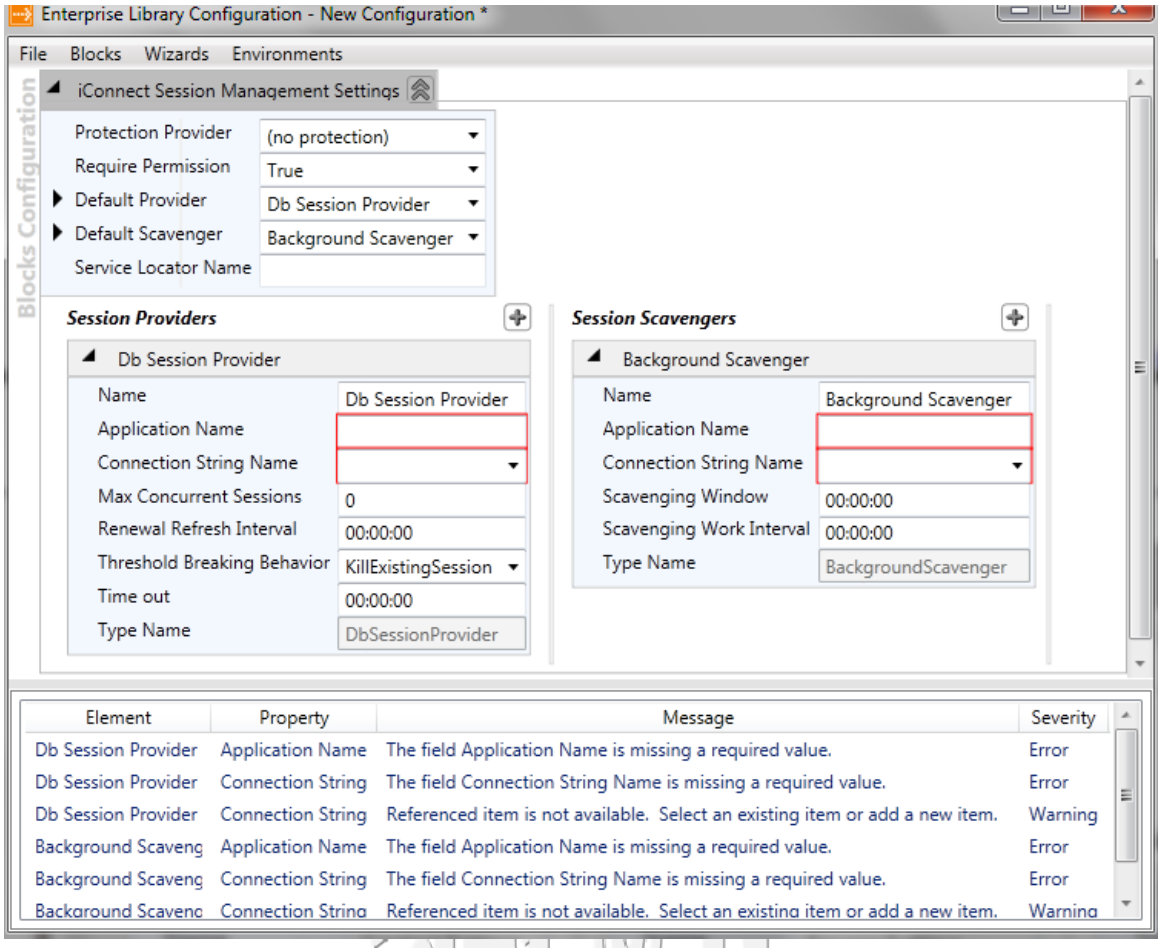


Figure 28.18. The layout of iConnect session management

28.2.19. iConnect state settings

1. Clicks "Blocks" -> "iConnect state settings"
2. Expands the setting of iConnect state, sets the related value for "Default Backing Store","Scavenging Interval" and Other;
3. Clicks "+"in the backing stores, adds the related backing store and then sets the related value for the setting;

   Note: for more detail, please refer to state chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

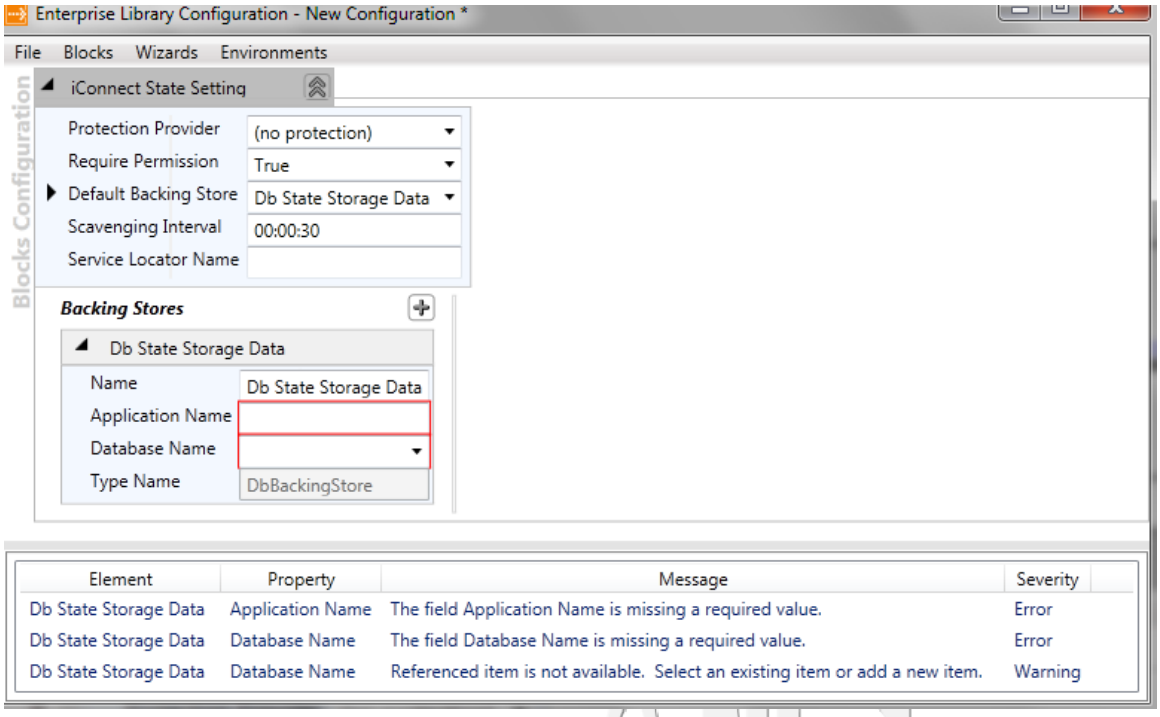


Figure 28.19. The layout of iConnect state setting

28.2.20. iConnect string resource settings

1. Clicks "Blocks" -> "iConnect string resource settings"
2. Expands the setting of iConnect string resource, sets the related value for "Default string resource provider" and other;
3. Clicks "+"in string resouce providers, adds the related provider and then sets the related value for the setting;
4. Expands the web integration option, sets the related value for the setting;

   Note: for more detail, please refer to string resource chapter.

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/

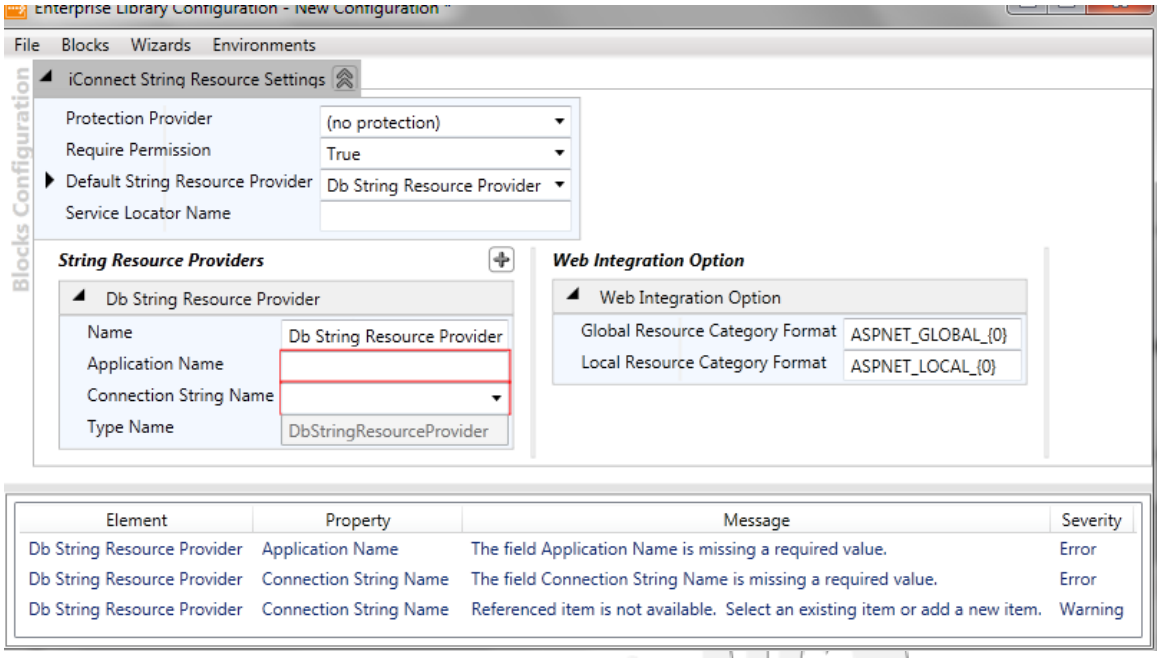


Figure 28.20. The layout of iConnect string resource settings

Copyright © NCS

Copyright © NCS Pte Ltd. All rights reserved.
No part of this publication may be reproduced or transmitted, in any form or by any means,whether electronic or mechanical, including photocopying and recording, without the written permission of the publisher.

XML to PDF by RenderX XEP XSL-FO Formatter, visit us at http://www.renderx.com/