webMethods JI

User's Guide

Version 4.5

January 2014
(originally released December 2004)

webMethods

This document applies to webMethods JI Version 4.5 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1999–2014 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, United States of America, and/or their suppliers. All rights reserved.

The name Software AG, webMethods, and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. Other company and product names mentioned herein may be trademarks of their respective owners.

Document ID: JI-UG-45-20140129

Table of Contents

About this Guide 17
Before You Begin . . . 17
Organization . . . 17
Formatting Conventions . . . 19
Documentation Set . . . 19
Viewing the Documentation Online . . . 21

Chapter 1. webMethods JI Overview . . . 23
About webMethods JI . . . 23
The Middleware Model . . . 23
The webMethods JI Environment . . . 24
Client Libraries and Interfaces . . . 24
The webMethods JI Runtime Server Environment . . . 25
webMethods JI Services . . . 25
Connectivity . . . 26
Load Balancing . . . 26
Components of the webMethods JI Server Environment . . . 26
webMethods JI Graphical User Interfaces (GUIs) . . . 28

Chapter 2. Managing the webMethods JI Environment . . . 29
webMethods JI Server Environment Overview . . . 30
Environment Managers . . . 31
Resource Servers . . . 31
Resource Database . . . 31
Client to Host Communication Process Overview . . . 32
Starting and Stopping the webMethods JI Environment . . . 32
Starting the Environment . . . 33
The *ea_start* Command . . . 34
Stopping the Environment . . . 37
The *ea_shutdown* Command . . . 37
Using ea_start and ea_shutdown with Multiple Server Components . . . 40
Pre- and Post-Start and Shutdown Scripts . . . 43
Environment Managers . . . 44
Starting the Environment Manager . . . 44
Using Batch Mode in UNIX . . . 44
Required Information at Startup . . . 45
Allowed Command Line Parameters . . . 46
Environment Manager Configuration File . . . 48
Starting Multiple Environment Managers . . . 48
Running Multiple Environment Managers on the Same Machine . . . 48
Starting Multiple Environment Managers from the Same Installation . . . 49
Shutting Down the Environment Manager . . . 50

Shutting Down the Environment Manager from the System Monitor . . . 51
Configuring Environment Managers . . . 51
Configuration Files. . . . 51
Environment Manager Configurations in the Resource Database . . . 52
Environment Manager Redundancy . . . 53
resource.cache file . . . 53
Multicast Channel Information . . . 53
Proxy Server. . . . 54
Environment Manager Logging . . . 54
JClusters . . . 54
JCluster Logging . . . 55
Starting JClusters . . . 56
Shutting Down JClusters . . . 56
JServices . . . 56
Shutting Down JServices . . . 56
Resource Server . . . 57
Resource Names . . . 57
Starting the Resource Server . . . 58
Required Information at Startup. . . . 58
Allowed Command Line Parameters . . . 59
Resource Server Configuration File. . . . 60
Starting Multiple Resource Servers . . . 61
Running Multiple Resource Servers on the Same Machine. . . . 61
Starting Multiple Resource Servers from the Same Installation . . . 61
Shutting Down the Resource Server . . . 62
Proxy Resource Server . . . 62
The Resource Database . . . 63
Starting the Resource Database . . . 63
Shutting Down the Resource Database . . . 64
Resource Database Redundancy . . . 64
Load Balancing. . . . 65
Load Balancing Groups . . . 65
Configuring Load Balancing Groups . . . 65
Environment Manager Configurations . . . 66
Configuring Environment Manager Configurations . . . 68
‘Fast’ Versus ‘Accurate’ Load Balancing . . . 70
Using Service Details to Balance Load . . . 71
Proxy Server. . . . 72
Monitoring . . . 73
Monitored Information . . . 73
Environment Manager Events and Activities . . . 73
JCluster Events and Activities . . . 74
JService . . . 74
Licensing . . . 74
Editing the License File . . . 75
Server Concurrency Restrictions . . . 75
JService Concurrency Restrictions . . . 75
Expiration Functionality . . . 76

License Exceeded Behavior 76
Redundant Licensing Behavior 76
Proxy Server 76
Configuring the Hub and Alternate Hub Proxy Servers 78
Configuring Satellite Proxy Servers 78

Chapter 3. MapMaker 81
Starting MapMaker 81
Command Line 82
The MapMaker Interface 82
MapMaker Views 83
Tree View 83
Presentation View 84
Properties View 85
MapMaker Menus 86
File Menu 87
Map Menu 89
Tools Menu 90
Help Menu 93
Shortcut Menus 93
MapMaker Toolbar 94
Naming Conventions 96
Configuring MapMaker Properties 96
General Tab 97
Appearance Tab 99
Default Directories Tab 100
Java Tab 102
Code Generation Tab 104
Servers Tab 105
Licensing Tab 107
Graph Window 108
Printing the Display 109
Generating Reports 111
Custom Classes 112
Defining Custom Classes 112
Compatibility Mode 114
Map Convertor 115
Method Compatibility Mode 115
Input Compatibility Mode 116
Output Compatibility Mode 116
Step Compatibility Mode 117
Batch Mode 117
Exporting 118
Exporting Maps to SDFX 118
Exporting Screens to SDFX 119
MapPlayer 120
Starting MapPlayer 120
Command Line 121

The MapPlayer Interface 121
MapPlayer Menus 121
MapPlayer Port Identification 122
MapPlayer Status Windows 122
Using MapPlayer to Play a Map in MapMaker 122

Chapter 4. Creating Maps in MapMaker 125
The Mapping Process 125
Defining Communications Connections 126
Block Mode Data Streams 126
NVT Mode Support 126
Character Mode Data Streams 128
Advanced Host Configuration 131
Character Mode Screen Transition Keys 134
Connecting to the Host 135
Trail Recording 137
The Trail Tab 138
Host Node 139
Trail Nodes 139
Snapshot Nodes 139
Field Nodes 140
Renaming Trails and Screen Snapshots 140
Using Multiple Trails 140
Adding Trails 140
Deleting Trails 141
Importing Trails 141
Enabling and Disabling Trails and Screen Snapshots 144
Stopping Trail Recording 144

Chapter 5. Customizing the Map 145
The Map Tab 145
Map Properties 147
Global Variables 150
Adding Global Variables to the Map 151
Global Actions 153
Using Global Actions with Table Templates 153
Adding Global Actions to the Map 153
Screen Properties 154
Cursor Settle 155
Settle Position Context 156
Customizing MapMaker's Screen Recognition 157
Block Mode 157
Character Mode 158
Tags 158
Block Mode Field Attributes 159
Character Mode Runtime Screen Recognition 159
Assigning a Tag 159

Scroll Tag Configuration . . . 161
Customizing Fields Used for Screen Recognition . . . 163
Using Floating Tags to Identify the Location of Floating Data Fields . . . 164
Actions . . . 164
Block Mode Only . . . 164
Character Mode Only . . . 165
Both Modes . . . 165
Modifying Actions . . . 165
Action General Tab . . . 165
Action Input Tab . . . 166
Action Destination Tab . . . 169
Comparing Screen Images . . . 171
Comparing One Screen Image with Another . . . 172
Viewing Results in The Differences Tab . . . 175
Viewing Results in The Screen 1 and Screen 2 Tabs . . . 178
Viewing Results in The Details Tab . . . 178
Comparing One Screen Image with All Screens in a Map . . . 179

Chapter 6. Data . . . 181
The Data Tab . . . 181
Data Templates . . . 183
Adding Data Templates . . . 183
Managing Data Templates . . . 183
Data Template Properties > General Tab . . . 184
Data Template Properties > Data Field Handling Tab . . . 185
Data Fields . . . 186
Adding Data Fields to a Data Template . . . 186
Resizing Data Fields . . . 187
Managing Data Fields . . . 187
Data Field Properties . . . 188
Data Field Properties > General Tab . . . 188
Using Floating Tags to Identify the Location of Floating Fields . . . 189
Table Templates . . . 190
Creating a Table Template . . . 190
Table Template Properties . . . 191
Table Template Properties > General Tab . . . 191
Table Template Properties > Table Record Controls Tab . . . 193
Table Template Properties > Table Paging Controls Tab . . . 196
Table Template Properties > Data Field Handling Tab . . . 198
Assigning Global Variables to Control Scrolling . . . 199
Adding Fields to Table Templates . . . 200

Chapter 7. Data Modeling . . . 201
Data Typing . . . 203
Internal Business Entities . . . 204
ErrorReport IBE Type . . . 205
Global Variables . . . 206

External Business Entities . . . 207
XML/XSD . . . 207
Map-List-Map . . . 208
Legacy Screen . . . 209
The Business Entity Editor . . . 210
The Business Entity Editor Shortcut Menus . . . 215
Managing Business Entities . . . 221
The Business Entity Editor Confirm Change Dialog Box . . . 234
Formatting Data Types . . . 236
The Business Entity Editor Tabs . . . 241
The Internal Data Definitions Tab . . . 242
The Global Variables Tab . . . 243
The XML/XSD Tab . . . 244
The MLM Tab . . . 246
The Legacy Screen Tab . . . 247
Defining Data Structure Relationships . . . 249
The Structure Relationship Editor . . . 249
Defining a Relationship Between Two Data Types . . . 251
Data Type Conversion Rules . . . 252
Using XPath Expressions in the Structure Relationship Editor . . . 254
Verifying the Result of an XPath Expression . . . 257
Specifying Data Sources in XPath Expressions in the Structure Relationship Editor . 258
XPath Expressions Not Reversible . . . 259
XPath Axes . . . 259
XPath Expressions and Changes to Business Entities . . . 261
Additional XPath String Functions . . . 262
Numeric Operators in XPath Expressions . . . 263
Indexing in XPath Expressions . . . 263
Data Mapping . . . 265
Chronological Order of Events in a Data Mapping Instance . . . 265
Different Types of Data Mapping Instances . . . 266
Data Mapping Location in a Method . . . 266
The First Data Mapping Instance . . . 266
The Last Data Mapping Instance . . . 267
A Data Mapping Instance in the Middle of a Method . . . 267
The Data Mapping Editor . . . 268
Working with the Data Mapping Editor . . . 269
Suggested Workflow for Creating a Method . . . 272

Chapter 8. Methods . . . 275
The Methods Tab . . . 275
Managing Methods . . . 277
Adding Methods . . . 277
Duplicating Methods . . . 278
Deleting Methods . . . 278
Configuring Method Properties . . . 278
Method Properties > General tab . . . 279

Inputs . . . 280
Adding Inputs . . . 280
Setting Input Properties . . . 281
Input Properties > General Tab . . . 281
Enabling and Disabling Inputs . . . 282
Method Variables . . . 282
Adding a Method Variable to a Method . . . 283
The Edit Variables Dialog Box Shortcut Menus . . . 285
Managing Method Variables . . . 286
Method Steps . . . 288
Adding Method Steps . . . 291
Deleting Method Steps . . . 292
Configuring Method Step Properties . . . 292
Start and Stop Steps . . . 292
Start Step . . . 293
Stop Step . . . 296
Screen-Based Steps . . . 297
Validating the Current Screen . . . 298
Recovery Action . . . 299
Traverse Steps . . . 300
Fetch Steps . . . 304
Perform Steps . . . 308
Write Steps . . . 311
UserInteraction Steps . . . 313
Invoke Steps . . . 315
Internal Invoke Steps . . . 315
External Invoke Steps . . . 316
JClient3 Steps . . . 329
Conditional Steps . . . 333
Working with Conditional Steps — Overview . . . 334
IfCondition Steps . . . 338
IfElseCondition Steps . . . 339
ForCondition Steps . . . 340
WhileCondition Steps . . . 342
DoWhileCondition Steps . . . 343
EndIf Steps . . . 343
Continue Steps . . . 344
Break Steps . . . 344
Set Steps . . . 345
Set Step Properties > General Tab . . . 346
Thread Steps . . . 350
ThreadJoin Steps . . . 352
Setting ThreadJoin Step Properties . . . 352
EndThread Steps . . . 354
Method Step Links . . . 355
Multiple-Mode Links . . . 356
Adding Links . . . 357
Method Step Restrictions . . . 359

Deleting Steps or Links . . . 359
Outputs . . . 360
Adding Outputs . . . 360
Enabling and Disabling Outputs . . . 361
Methods Presentation View Color Code . . . 361
Keyboard Shortcuts and Search Options . . . 361

Chapter 9. Debugging Methods . . . 363
General Debug Procedure. . . . 363
Workflow for Debugging a Method . . . 364
Step 1: Accessing the Debugger . . . 365
Debugger Views . . . 365
Debugger Menus . . . 373
Debugger Toolbars . . . 373
Step 2: Enabling and Disabling Breakpoints . . . 377
Step 3: Defining Debug Cursor Properties . . . 378
Step 4: Viewing the Connected Host . . . 379
Accessing the Host Screen Viewer. . . . 379
Stepping Through a Screen-Based Step . . . 380
Step 5: Viewing and Editing Method Data . . . 383
Accessing the TypeDataModel Viewer . . . 383
TypeDataModel Viewer Components . . . 384
Step 6: Watching Values Update . . . 389

Chapter 10. Services . . . 391
The Services Tab . . . 391
Adding Services . . . 392
Configuring Service Properties . . . 392
Service Validation Checks . . . 395
Adding Methods to Services . . . 397
Generating the Service and Test Client . . . 398
Generating All Code . . . 398
Service Tab . . . 399
Client Tab . . . 401
Definitions Tab . . . 403
Supplemental Tab . . . 405
Resource Adapter Tab . . . 406
Generating Service Code . . . 414
Generating Client Code . . . 415
Generating Definition/Description Files . . . 415
Custom Code Conversion Utility . . . 416
ASP.NET Client Generation . . . 416
Test Client Code . . . 417
Compiling the JClient3 Test Client . . . 417
Compiling the JClient3 Test Client Without Using the Makefile . . . 417
Testing the Service . . . 418
JClient3 Test Client . . . 419

Deploying the Service 419
Deploying the Map and/or Custom EAServiceBean 419
The webMethods JI Tab 420
Selecting the Resource Server for Deployment 421
Selecting the Database for Deployment 421
Deploying the Service as a Web Service 422
Defining Users as Administrators in Tomcat 423
The Web Services Tab 423

Chapter 11. Customizing MapMaker-Generated Service Code 427
Introduction to Custom Classes 427
JavaDoc 427
Using Threads with Custom Code 427
Custom Classes -- Examples of Common Uses 428
Example One: Pre and/or Post Processing 428
Example Two: Accessing External Data Sources 429
Example Three: Branching 429
Defining Custom Classes in MapMaker 430
Extending the EAServiceBean 430
Deploying Extended EAServiceBeans 431
Deploying into the Resource Database 431
Using Extended EAServiceBeans from the Local File System 431
Example Code for Extended EAServiceBean 431
Customizable Components in MapMaker 431
Template Data Architecture 433
Example Code 435
JDBC Access 435
Branching 436
Echo Test/Ping 436
Limiting Login Attempts 436
Custom Output Variables 436
Client Host Selection 436
Processing Extracted Data and Inputting the Processed Data 437
Using Custom Code to Access Variable Data 437
Overview 437
Conventions Used in These Examples 438
Code Example 1a - Get Value from Top Level InternalString 439
Code Example 1b - Get Value from Top Level InternalNumber 440
Code Example 1c - Set Value of Top Level InteralString 441
Code Example 1d - Set Value of Top Level InternalNumber 443
Code Example 1e - Set Value of InternalString Within an IBE 444
Code Example 1f - Set Value of InternalString Array in an IBE 446
Code Example 1g - Set the Value of an InternalString Sub-element in an IBE Array . . . 447
Code Example 2 - Add a Structure to an Array Within an IBE 448
Code Example 3 - Determine the Type of an Element in a BE 450
Code Example 4 - Determine Whether an Element in a BE is an Array or Not 451
Code Example 5 - Copy Data from Structure to Structure 452
Value Classes for setValue Calls 453

Chapter 12. Configuration Manager 455
Starting the Configuration Manager 455
Command Line 455
The Configuration Manager Interface 456
Configuration Manager Views 456
Tree View 457
Property View 457
Configuration Manager Menus 457
File Menu 458
Edit Menu 460
Tools Menu 461
Help Menu 461
Shortcut Menus 462
Configuration Manager Toolbar 462
Getting Help from Within Configuration Manager 464
Configuring Environment Managers 464
About Environment Managers 464
Locating Environment Managers 464
Automatically Locating Environment Managers 465
Manually Locating Environment Managers 465
Environment Manager Configuration Properties 466
Configuring Resource Servers 470
About Resource Servers 470
Locating Resource Servers 470
Automatically Locating Resource Servers 470
Manually Locating Resource Servers 471
Resource Server Configuration Properties 472
Configuring Resources 474
About Resources 474
Resource Databases 475
JI License Files 475
Adding Resources to the Configuration Manager Tree 475
Resource Configuration Parameters 476
Configuring Databases 477
Opening Resource Databases for Editing 477
Environment Manager Configurations 478
Environment Manager and JCluster Configuration 479
Service Details 486
Service Masters 489
Adding Service Masters 489
Copying and Pasting Service Masters 489
Setting Properties for Service Masters 490
Host Connections 498
Adding Host Connections 498
Setting Properties for Host Connections 498
Deployed Maps and Custom EAServiceBeans 499
Deployed Maps 499
Deployed Custom EAServiceBeans 502

Licensing 504
The License Key 504
License Properties 504
Feature Type 505
License Count 506
Expire Flag 506
The License File (license.txt) 506
Adding and Editing a License Key 507
Adding an Additional License Key Node 509
Adding an Additional License Data Source 509
Checking the Status of webMethods JI Licenses 510
License Expiration Email Notification 511
Resource Server Configuration File (ressvr.cfg) Email Definitions 513
Duplicate Licenses 513
License Compatibility 514

Chapter 13. System Monitor 515
Starting the System Monitor 515
Command Line 515
The System Monitor Interface 516
System Monitor Views 516
Tree View 517
Panel View 517
Details View 518
System Monitor Menus 519
File Menu 520
View Menu 520
Actions Menu 521
Tools Menu 522
Help Menu 523
Shortcut Menus 524
System Monitor Properties 524
Monitoring Your webMethods JI Environment 526
Environment Manager Events and Activities 526
The Environment Manager Tree View 527
The Environment Manager Panel View 527
The Environment Manager Detail View 531
JCluster Events and Activities 534
The JCluster Tree View 535
The JCluster Panel View 535
The JCluster Detail View 539
JService Events and Activities 543
Dynamically Changing the JService Trace Level 543
The JService Tree View 543
The JService Panel View 544
The JService Detail View 549

Chapter 14. Access Control List for webMethods JI 551
Program Resources Controlled by ACL 551
Command Line Actions Controlled by ACL 552
Permissions 553
Creating the Initial Administrator Account 553
Enabling Access Control 556
Adding New ACL Accounts and Permission Records. 557
Adding a New ACL Account Record 558
Modifying the ACL Account Record. 559
Adding a New ACL Permission Record 560
Modifying the ACL Permission Records. 561
Permission Records Usage. 561
Application Behavior With ACL-Controlled Environments 562
Compatibility with Previous Versions of webMethods JI 563
Pre-ACL Environment Manager Access to ACL-Enabled Resource Servers 564
Prior Application Access to ACL-Enabled Environments 565

Chapter 15. Protocol Agents 567
Overview of Protocols Supported with webMethods JI. 567
Defining the Communications Protocol 567
Trace Information 567
Protocol Agent Terminal Window 568
Creating a Protocol Agent Terminal (PATerm) Window 568
Creating a Server-side Terminal Window 568
Creating a Client-side Terminal Windows. 569
Components of a Protocol Agent Terminal Window 570
Server-side Terminal Window 570
PATerm. 570
JTerm 571
Macro and Cut/Paste 573
Default Keyboard Mapping for JTerm Windows 573
Write-enabled Terminal Windows 581
Remote JTerm (PATerm) Windows 582
Installing and Configuring the Remote JTerm 583
Installing the Remote JTerm Feature - Stand Alone 583
Installing the Remote JTerm Feature - Java Applet. 584
Configuring the Remote JTerm Applet 585
Importing the Certificate in Internet Explorer 588
Enabling and Using SSL 589
Granting Permissions to the JSSE JAR Files. 589
Signing the JSSE JAR Files. 590
Launching a Remote JTerm From the Command Line 591
Using the Batch/Shell Script File to Launch the Remote JTerm 594
Launching a Remote JTerm From a Browser 595
Remote JTerm Errors 597
HTTP and MQ Gateway Changes 598
Siebel Specific Instructions 598
Error Logging 598

Status 599

Chapter 16. Secure Socket Layer (SSL) Implementation 601
About SSL 601
SSL Requirements and Limitations 602
Limitations 602
Requirements - Server Changes 602
Key and Certificate Files 602
genkeys Usage Notes 603
Environment Manager Configuration 603
JClient3 SSL-Related API Information 603
Methods 603
Exceptions 604
Enabling Encryption Using SSL 605
Enabling Encryption 605
Enabling Server Authentication 605
Logging 606
Server Logging 606
JClient3 Logging 606
Java SSL Debugging 606

Chapter 17. Troubleshooting 607
Installation Issues 607
Temporary Directory 607
UNIX 607
Windows 607
Runtime Issues 608
Using the *ea_start* and *ea_shutdown* executables 608
The EA_ENV Environment Variable 608
DISPLAY Environment Variable 608
Localhost 609
Client/Service Interaction Issues 609
Services Running after Client Disconnects 609

Glossary 611

Index 621

About this Guide

Welcome to the *webMethods JI User's Guide*.

> **Note:** Appendices formerly found at the end of this document are now part of the webMethods JI Supplementary Reference Guide.

Before You Begin

This guide is intended for software developers who want to create applications using webMethods JI.

In order to use webMethods JI, developers should have a working knowledge of the following:

- UNIX and/or Windows® system administration.
- Java™ programming language for developing webMethods JI services.
- The location and communication requirements of existing legacy data and applications.

Optional requirements include:

- Java, C and/or ActiveX programming languages for developing webMethods JI clients.
- HTML design for using the webMethods JI Common Gateway Interface.
- Siebel Enterprise Relationship Management applications to integrate webMethods JI with Siebel environments.

Organization

This guide is organized as follows:

- Chapter 1 - "webMethods JI Overview" on page 23 summarizes the features and benefits of the webMethods JI development tool.
- Chapter 2 - "Managing the webMethods JI Environment" on page 29 provides information about how to configure webMethods JI runtime environments, including load balancing, redundancy, and other environment issues.
- Chapter 3 - "MapMaker" on page 81 introduces the MapMaker graphical development environment and describes how to use MapMaker to create

maps to legacy applications, and how use those maps to generate webMethods JI services and test clients.

- Chapter 4 - "Creating Maps in MapMaker" on page 125 describes how to connect MapMaker to Host and how to perform Trail Recording using MapMaker's Trails tab.
- Chapter 5 - "Customizing the Map" on page 145 describes how to use the MapMaker's Map tab to customize the map created in the Trail Recording stage of development.
- Chapter 6 - "Data" on page 181 describes how to use the MapMaker's Data tab to create Data Templates and Table Templates.
- Chapter 7 - "Data Modeling" on page 201 describes how to use the MapMaker's Data Modeling capabilities, which include Data Typing, Defining Structure Relationships and Data Mapping.
- Chapter 8 - "Methods" on page 275 describes how to use the MapMaker's Methods tab and how to build and maintain methods
- Chapter 9 - "Debugging Methods" on page 363 describes how to use the MapMaker's Debugger feature to locate and correct errors in a method.
- Chapter 10 - "Services" on page 391 describes how to use the MapMaker's Services tab and how to generate webMethods JI services and test clients.
- Chapter 11 - "Customizing MapMaker-Generated Service Code" on page 427 describes the areas in MapMaker that allow Custom Classes to be added, and a description of important architectural details regarding the functions of webMethods JI services.
- Chapter 12 - "Configuration Manager" on page 455 provides a reference for the webMethods JI Configuration Manager, a GUI used to administer configuration data within webMethods JI environments.
- Chapter 13 - "System Monitor" on page 515 describes the webMethods JI System Monitor, a GUI that is used to monitor webMethods JI environments.
- Chapter 14 - "Access Control List for webMethods JI" on page 551 describes the Access Control List (ACL) functionality in webMethods JI, which allows control over clients accessing the webMethods JI server and certain webMethods JI functions.
- Chapter 15 - "Protocol Agents" on page 567 provides information about the Protocol Agents used by webMethods JI.
- Chapter 16 - "Secure Socket Layer (SSL) Implementation" on page 601 describes how to implement SSL to encrypt data transferred between a client and a JI Service.
- Chapter 17 - "Troubleshooting" on page 607 provides information used to troubleshoot webMethods JI environments.
- "Glossary" on page 611
- "Index" on page 621

Formatting Conventions

The following formatting conventions are used in this manual:

Table 1. Formatting conventions

Convention	Used for..
Italics	Italics are used for files, directories, programs, and book titles. For example: *<JI_install_dir>/bin/ea_mapmaker.exe*.
`Monotype font`	A monotype font is used to represent examples of code, characters that the user enters, and prompts or messages from the system. For example: Type `ea_start` at the command prompt.
Sans-serif font	A sans-serif font is used to represent Graphical User Interface (GUI) features, such as buttons. For example: Press the Help button to display a list of help topics.
Serif bold font	This font is used for notes and warnings that require special attention. For example: Warning: You must install webMethods JI in an empty directory.

Documentation Set

webMethods JI is supplied with the manuals shown below. The documentation is delivered in Adobe Acrobat Reader Portable Document Format (PDF). No hardcopy documentation is provided, but you can print the PDF files on your local printer.

Use this guide in conjunction with other manuals provided with webMethods JI:

Table 2. webMethods JI documentation set

Title	Description
webMethods JI Release Notes	Provides information about additions and revisions to the current release of webMethods JI. The release notes are distributed in two forms, as a PDF and as a text file
webMethods JI Installation and Configuration Guide	Details installation procedures for webMethods JI.
webMethods JI Tutorial	Provides hands-on instruction about how to write webMethods JI services using MapMaker
webMethods JI User's Guide	Describes how to configure the webMethods JI environment, as well as how to use webMethods JI graphical development and system monitoring tools.
webMethods JI Client Developer's Guide	Describes how to design and develop webMethods JI client applications, as well as how to integrate webMethods JI services into third-party development environments.
webMethods JI Integration Guide	Contains information about integrating webMethods JI Services with other technologies, such as Siebel eBusiness Applications, MQSeries, Web Services, and more.
webMethods JI API Function Reference	Describes the webMethods JI C and Java functions that are used to develop webMethods JI clients and services. The reference provides syntax, parameters, return values, examples, and related information.

Table 2. webMethods JI documentation set

Title	Description
webMethods JI Supplemental Reference Guide	Contains additional information on webMethods JI commands, error codes, language translation, keyboard mapping, logging, licensing, file formats, and field attributes. This guide replaces the Appendices that were duplicated across the manual set.

Viewing the Documentation Online

Online documentation is available in the following locations:

- In webMethods JI Graphical User Interfaces (GUIs), click on the **Help** menu and select **Help Topics** to display the webMethods JI documentation.
- webMethods JI documentation is available on-line in Adobe® Acrobat™'s Portable Document Format (PDF). If the documentation has been installed, open the file *<JI_install_dir>/doc/_ji_doc.pdf* in Adobe Acrobat Reader™ or a Web browser (where *<JI_install_dir>* is the location of your webMethods JI installation).

You can also access the latest version of the documentation for Software AG products at http://documentation.softwareag.com/. As new versions become available, the documentation on this web site will be updated and the previous versions will be migrated to the Empower Product Support Web site at https://empower.softwareag.com/. If you have a maintenance contract, you can view all versions of documentation on this web site. You will find instructions for registering and obtaining a userid and password on the documentation web site.

Chapter 1. webMethods JI Overview

This chapter provides an introduction to webMethods JI. This tool provides enterprise-wide, client/server access to legacy or mainframe applications and databases.

The following topics are discussed in this chapter:

- Client libraries and interfaces that are used to interact with webMethods JI environments and services.
- The webMethods JI runtime server environment.
- Tools used to create webMethods JI services.

About webMethods JI

webMethods JI is an application development tool and runtime server environment. It provides an industry leading, enterprise-strength solution for real-time interaction between users in client/server networks and applications running on legacy mainframe computer systems. The webMethods JI application development tool and Application Program Interfaces (APIs) allow customers to create modern client/server interfaces to their data without re-engineering their legacy applications. The webMethods JI runtime server environment allows for large-scale implementation of these clients and services. By using load-balancing and other architectural features, a solution that can be used in large enterprise networks is provided for mainframe access.

The Middleware Model

webMethods JI is a middleware solution. It connects front-end users (“clients”) with back-end, legacy applications running on mainframes or on other legacy systems. The webMethods JI runtime server environment sits in the “middle” controlling and monitoring connections between the clients and legacy applications. A robust service API allows services to implement modern business logic, to facilitate the client-to-host communications.

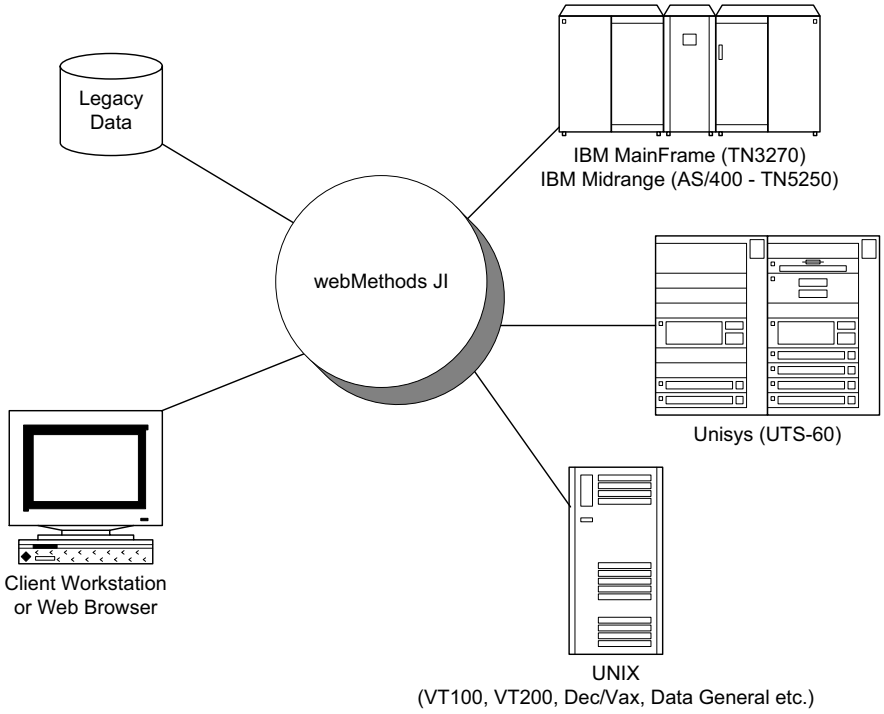


Figure 1. webMethods JI as a Middleware Solution

The webMethods JI Environment

The webMethods JI environment consists of three primary components:

- Client libraries and interfaces.
- The runtime server environment. This includes user-developed services that communicate with legacy applications.
- Graphical User Interfaces (GUIs), used to develop services and to configure and monitor the runtime environment.

Client Libraries and Interfaces

In the webMethods JI middleware model, clients are the first tier of the environment. They are the software tools that are used by end-users to interact with legacy applications. Clients that interact with webMethods JI are implemented using two general tools:

- **webMethods JI client libraries:** Client libraries are available to develop clients that can be deployed in all major environments (including client

libraries developed in Java, C, and ActiveX). Client applications can be Java applets, applications or servlets; or C, C++ or Visual Basic applications.

- **webMethods JI client interfaces:** These interfaces are pre-packaged implementations, developed by Software AG using the webMethods JI client libraries. These include JavaServer Pages, JavaBeans, and interfaces to the Siebel development and runtime environment.

For more information about webMethods JI client libraries, see the *webMethods JI Client Developer's Guide*.

The webMethods JI Runtime Server Environment

webMethods JI provides a robust runtime environment that includes the components necessary to provide client-to-service connectivity, service-to-legacy application connectivity, and the tools necessary to configure and monitor multiple environments.

Note: Unless otherwise noted, tools in the webMethods JI environment are developed entirely in the Java programming language.

webMethods JI Services

webMethods JI services are implemented in the Java programming language and are developed using the MapMaker graphical user interface. These Java services are deployed in the webMethods JI runtime environment, and run within a "JService" interface. The JService runs as a single thread within a larger process, called a JCluster.

Clients connect to a service using either client function calls or via one of the client interfaces. Once connected to the service, the client invokes service methods that connect to the legacy application and perform whatever tasks are required by the method. The service then returns any data from the legacy application to the client.

C Services

All webMethods JI services are now implemented using the Java programming language. However, in previous webMethods JI versions, services were written using the C programming language. A "compatibility kit" included with webMethods JI allows users of previous webMethods JI versions to use their existing C services. C services are compiled and configured using a JService to access the legacy C service. For more information about C services and the compatibility kit, see the *webMethods JI Compatibility Kit User's Guide*.

Connectivity

The following protocols are supported for communication between webMethods JI services (JServices) and legacy host machines:

- TN3270/TN3270E Models 2-5: (Models 3, 4, and 5 are supported with new Java services only)
- TN5250 Standard (80 column or 132 column): The session capability of TN5250E is also supported, although no other aspects of TN5250E are implemented.
- Telnet (VT100, VT220, VT320 and ANSI emulation): Telnet is used for Character Mode and referred to as such throughout this manual.

The communications infrastructure for service-to-legacy interaction is provided within webMethods JI services and the webMethods JI runtime environment. The transparent nature of this implementation means that there is no need to spend valuable development time with communication details. Furthermore, no communication code needs to be included with client applications.

All host communications include built-in support for single byte Western European Latin alphabet languages. TN3270 and TN5250 also include support for double byte Asian languages. This support consists of predefined tables for conversion between ISO 8859 ASCII and the national variants of EBCDIC on the back end systems.

For more information about ASCII to EBCDIC and EBCDIC to ASCII conversion, see Chapter 6 - "National Language Support" on page 173 of the *webMethods JI Supplemental Reference Guide.*

Load Balancing

webMethods JI allows for balancing of client and service load, by directing requests from clients to the environment that can best handle the request. This provides for redundant failover protection as well. For more information about load balancing and other configuration issues, see Chapter 2 - "Managing the webMethods JI Environment" on page 29.

Components of the webMethods JI Server Environment

- **Clients:** Technically, clients that interact with webMethods JI services are not a part of the server environment, but they are the software component with which the end-user interacts. Client-to-server interaction can be developed using the webMethods JI client libraries in Java, C or ActiveX, or can be developed using one of the webMethods JI client interfaces. For more information about webMethods JI clients, see the *webMethods JI Client Developer's Guide*.

- **Services:** webMethods JI services provide the connection between the webMethods JI environment and the legacy application. Services are developed in the MapMaker GUI and are implemented in the Java programming language. JServices provide an interface between services and the webMethods JI runtime environment.

> **Note:** C services, implemented in the C programming language using previous versions of webMethods JI, can also be used in the webMethods JI environment. For more information about C services and the compatibility kit, see the *webMethods JI Compatibility Kit User's Guide*.

- **Legacy Applications:** webMethods JI services communicate with legacy applications, which can be running on mainframe, AS400, or UNIX hosts.

  The following protocols are supported for communication between the webMethods JI environment and legacy host machines:
  - TN3270/TN3270E Models 2-5
  - TN5250 Standard (80 column) and Large (132 column) screens.
  - Telnet (VT100, VT220, VT320 and ANSI)
- **Environment Managers:** Environment Managers are processes that run within the webMethods JI environment. These act as the central management and control point for webMethods JI server-side processing. The Environment Manager performs the following tasks:
  - Provides a central access point for clients to communicate with the webMethods JI environment.
  - Connects clients to the appropriate JClusters.
  - Manages processes, including starting and stopping JClusters and Resource Servers.
  - Load balancing.
  - Provides a central monitoring point for the entire webMethods JI environment.
- **JClusters:** A JCluster is a single process that runs within the webMethods JI environment. JClusters are managed by the Environment Manager, and are used to manage individual JService threads. JClusters connect clients to the appropriate JServices.
- **JServices:** A JService is a thread that runs within the JCluster process. JServices serve as interfaces to webMethods JI Java services.
- **Resource Server:** A Resource Server is a process that runs within the webMethods JI environment and acts as a centralized resource provider. The resource server facilitates access to webMethods JI server-side data (such as information stored in the webMethods JI Resource Database).
- **Resource Database:** The webMethods JI Resource Database, implemented using the H2 Database Engine, is used to store runtime environment configuration information, maps generated in the MapMaker graphical interface, and runtime webMethods JI Java service code.

Note: For C services that were developed using previous versions of webMethods JI, the Resource Database stores Protocol Agent definitions, screen definitions, and service metadata.

webMethods JI Graphical User Interfaces (GUIs)

webMethods JI includes three graphical user interfaces (GUIs). These GUIs are Java applications that use standard elements of modern graphical interfaces. The following GUIs are included with webMethods JI:

- **MapMaker:** MapMaker is a Java-based GUI development tool, which facilitates webMethods JI service code development. Java services are generated from the MapMaker interface based on “maps” which correspond to navigation and field information from the legacy application.

Note: MapMaker can output webMethods JI services as webMethods JI Java services for use within the standard webMethods JI runtime server environment. For more information about MapMaker, see Chapter 3 - "MapMaker" on page 81.

- **Configuration Manager:** The Configuration Manager is a Java-based GUI, used to administer configuration information within the webMethods JI environment. This configuration information is stored in the Resource Database and is retrieved at runtime by various components of the environment. For more information about the Configuration Manager, see Chapter 12 - "Configuration Manager" on page 455.
- **System Monitor:** The System Monitor is a Java-based GUI, used to monitor activity within the webMethods JI runtime environment. For more information about the System Monitor, see Chapter 13 - "System Monitor" on page 515.

Chapter 2. Managing the webMethods JI Environment

This chapter provides a detailed description of the server-side components of the webMethods JI environment, and describes how to organize and configure the environment. The following items are discussed:

- "webMethods JI Server Environment Overview" on page 30
- "Client to Host Communication Process Overview" on page 32
- "Starting and Stopping the webMethods JI Environment" on page 32
- "Environment Managers" on page 44
- "Resource Server" on page 57
- "The Resource Database" on page 63
- "Load Balancing" on page 65
- "Monitoring" on page 73
- "Licensing" on page 74
- "Proxy Server" on page 76

webMethods JI Server Environment Overview

The webMethods JI server environment provides an infrastructure to manage client/service communications, including:

- Load balancing
- Monitoring
- Logging

The following components are included in the webMethods JI environment:

- Environment Managers
- JClusters
- JServices
- Resource Servers
- Resource Databases
- Other resources

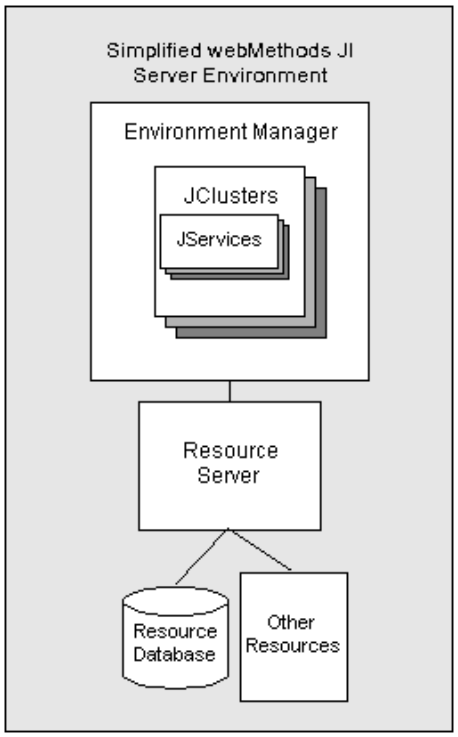


Figure 2. Overview of the webMethods JI Server Environment

Environment Managers

Environment Managers serve as the central access and control point for the webMethods JI environment. Clients use the host and port of the Environment Manager to contact the Environment Manager. Once contacted, the Environment Manager uses multicasting to query each Environment Manager running in its load balancing group. This is done to determine which Environment Manager is best equipped to handle the client request. A variety of load balancing parameters allow for fine-tuning of Environment Manager interaction. For example, fast machines can be set to respond quickly to client requests, or the highest-speed machines can do the most common services.

Multiple Environment Managers can be running within the enterprise. Typically, one Environment Manager is running on a single host machine. Environment Managers can be scaled using JCluster configuration and other information such as timeouts. For more information about Environment Managers, see "Environment Managers" on page 44.

Resource Servers

Resource Servers provide access to resources running in the webMethods JI environment. This can be a one-to-one relationship between Environment Managers and Resource Servers. It could also be a single Resource Server servicing any number of Environment Managers. Environment Managers locate Resource Servers by using multicasting technology to request a specific resource. The Resource Server serving that resource responds with its host and port. After the response, the Environment Manager and Resource Server interact using Remote Method Invocation (RMI). For more information about Resource Servers, see "Resource Server" on page 57.

Resource Database

The Resource Database is used to store Environment Manager load balancing group and configuration information, as well as host connection and other information. webMethods JI services generated from within MapMaker can be deployed to the database. There should be one Resource Database for each Environment Manager load balancing group. Different groups can share the same Resource Database, or each Environment Manager could have its own database. Additionally, two or more instances of identical Resource Databases can be used to provide redundant fail-over functionality. For more information about Resource Databases, see "The Resource Database" on page 63.

Another item included with the resources is licensing information.

Client to Host Communication Process Overview

Clients use webMethods JI functions to send messages to the webMethods JI environment. These messages provide this environment with information about the location of the Environment Manager, the name of the service and any input parameters that the service uses to interact with the legacy application.

This client/server interaction consists of the following steps:

1. The client sends a message to the Environment Manager, making a “Request for Service” (RFS).
2. The Environment Manager uses load balancing criteria and other information to determine which Environment Manager should process the client request. The appropriate Environment Manager then forwards the RFS to a JCluster. The JCluster accesses the JService that interfaces with the webMethods JI service that was requested. The client is then able to communicate directly with the service.
3. Once the client and service are connected, the client sends a “service message” back. This message consists of name/value pairs, defining fields and values for those fields. For example: `acct=135`.
4. The service queries the legacy application based on the information provided by the client.
5. The service retrieves information from the legacy application and constructs a new service message containing the new information. This message can contain data from a number of legacy application screens, or even more than one application and/or more than one legacy host.
6. The service then sends the new service message to the client.

Starting and Stopping the webMethods JI Environment

Executables are included to start up and shut down all server components in the webMethods JI environment. The Resource Database server, the Resource Server and the Environment Manager can all be started and stopped using these programs.

These executables (*ea_start* and *ea_shutdown*) provide flexibility over using the command line executables and batch files for each individual server component. However, they require some consideration for their use:

- **The EA_ENV environment variable:**

  The EA_ENV environment variable must be set to the *<JI_install_dir>/config* directory. The command line option `-c <configuration directory>` can be used instead of setting EA_ENV.

- ***environment.ccf* file:**

  The *environment.ccf* file must be located in the directory referenced by the EA_ENV environment variable. It must contain the settings for the EA_HOME, DB_TCP_PORT, and EA_ENVMGR parameters:
  - EA_HOME: The root directory of your webMethods JI installation.
  - DB_TCP_PORT: The port number of your Resource Databasee server.
  - EA_ENVMGR: The server port for the Environment Manager, used to ping the Environment Manager to determine if it is running.

  Also, the EA_HOME, DB_TCP_PORT, and EA_ENVMGR parameters can be defined as environment variables if the *environment.ccf* file is not used.
- ***ressvr.cfg* file:**

  Use the default name, *ressvr.cfg*, for the Resource Server configuration file. It must contain the correct RMI port number for the `config.rmi` parameter. This file must be located in the directory referenced by the EA_ENV environment variable.
- ***envmgr.cfg* file:**
  Use the default name, *envmgr.cfg*, for the Environment Manager configuration file. It must contain the correct path and filename for the `javacmd` parameter, that points to the Java Virtual Machine command. This file must be located in the directory referenced by the EA_ENV environment variable.

These restrictions also apply to the *ea_admin*, *ea_ping*, and *ea_status* command line variables.

Starting the Environment

There are two ways to start components in the webMethods JI server environment:

- By using the *ea_start* command. This command starts all components in the environment at once, or selects various components. For more information, see “The ea_start Command” on page 34.

  Pre-start and post-start scripts can be included to customize the startup executable. For more information, see “Pre- and Post-Start and Shutdown Scripts” on page 43.
- By using individual commands for each component. This option allows more startup parameters to be passed at the command line. These parameters can include: log level, log file location, and other information. The following commands are available:
  - ***ea_envmgr:***
    Starts the Environment Manager. For more information, see “Starting the Environment Manager” on page 44.
  - ***ea_start -r:***
    Starts the Resource Server. For more information, see “Starting the Resource Server” on page 58.

- ***ea_dbsvr:***
  Starts the Resource Database server. For more information, see "Starting the Resource Database" on page 63.

Note: In Windows, individual start commands can be installed as "shortcuts". These allow the user to select the shortcut from the Windows graphical interface.

The *ea_start* Command

ea_start is a command line executable that can be used to start all or any combination of the components in the webMethods JI server environment. Use the following command:

```
cd <JI_install_dir>\bin
.\ea_start
```

Required Information for *ea_start*

In order to execute the *ea_start* command correctly, this program must be able to determine the following information:

- **webMethods JI installation directory:**
  To find the installation directory, *ea_start* requires one of the following:
  - The `-i <JI_install_dir>` command line flag must be used to identify the installation directory.
  - The EA_ENV environment variable must be set and must identify the location of the *environment.ccf* file, which contains a definition of EA_HOME.
- **Port Numbers of the Server Components:**
  The port numbers for each of the server components are determined as follows:
  - **Environment Manager**
    Command line option `-s <EnvMgr_port>`
    The port entry in the Resource Database for the Environment Manager's configuration
    In order to ping the Environment Manager to determine if it is running, the value of the EA_ENVMGR parameter in the *environment.ccf* file should be changed to match the entry for the server port in the Resource Database.

Note: If this parameter is not set, *ea_status* looks for the value of the EA_ENVMGR environment variable. If this is not found, it uses the default value for the Environment Manager port (`30001`).

- **Resource Server**
  Command line option `-p <RMI_port>`
  The value of the `config.rmiport` parameter in *ressvr.cfg*, the default Resource Server configuration file. The *ressvr.cfg* file is located in the *<JI_install_dir>/config* directory.
- **Resource Database Server**
  Command line option `-Q <DB_port>`
  The value of the DB_TCP_PORT parameter in *environment.ccf* file.
  The value of theDB_TCP_PORT environment variable.
  The default Resource Database server listen port (`30000`)

- **Java command:**

  Because *ea_start* executes underlying Java programs to start the webMethods JI environment, it is necessary to locate the correct Java Virtual Machine (JVM) executable (*java* in UNIX or *java.exe* in Windows).

  The location of the JVM is located by using either of the following:

  - The value of the `javacmd` parameter in *envmgr.cfg*, the default Environment Manager configuration file. The location of the *envmgr.cfg* file is assumed to be the *<JI_install_dir>/config* directory.
  - `Java` is used as a default command line if the *envmgr.cfg* file is not found or the `javacmd` parameter is not set. If this option is used, the JVM executable must be in the PATH environment variable.

Allowed Command Line Parameters for the *ea_start* command

Parameters	Description
`-i <JI_install_dir>`	The webMethods JI installation directory.
`-s <EnvMgr_port>`	The Environment Manager server port.
`-p <RMI_port>`	The Resource Server RMI port.
`-Q <DB_port>`	The Resource Database server listen port.
`-e`	Starts only the Environment Manager.
`-r`	Starts only the Resource Server.
`-D`	Starts only the Resource Database server.
`-n <retries>`	The number of times that *ea_start* checks to see if a server component has started. The default setting is `10`.
`-t <seconds>`	The number of seconds that *ea_start* waits before rechecking if a server component has started.
`-x`	Enable debugging output.
`-h`	Show help.

Note: For more information about the *ea_start* command line interface, see the *webMethods JI Supplemental Reference Guide*.

Examples for Using *ea_start*

The following example starts an Environment Manager, a Resource Server, and the Resource Database server:

```
ea_start -i d:\JI -s 16024 -p 10024 -Q 16524
```

In this example, the webMethods JI installation location is identified as d:\JI, using the -i parameter. The following port numbers are also identified:

- Environment Manager: `-s 16024`
- Resource Server: `-p 10024`
- Resource Database server: `-Q 16524`

Often, the Resource Database server is shared among several environments and will be already running. In this situation, the user may only need to start the Resource Server and Environment Manager. This can be done by executing the *ea_start* command using the -e and -r command line options:

```
ea_start -i d:\JI -e -r
```

> **Note:** This example does not use the `-s <EnvMgr_port>` and `-p <RMI_port>` command line options to set the port numbers for the Environment Manager and Resource Server. It assumes that the port numbers can be determined by *ea_start* using the environment.ccf (or EA_ENVMGR environment variable) and ressvr.cfg files.

Stopping the Environment

The webMethods JI server environment is shut down using the *ea_shutdown* command. This allows the user to either shutdown all components in the environment at once, or to select individual ones. Pre-shutdown and post-shutdown scripts can be included to customize the shutdown executable. For more information, see "Pre- and Post-Start and Shutdown Scripts" on page 43.

The *ea_shutdown* Command

ea_shutdown is a command line executable that can be used to shut down all or any combination of components in the webMethods JI server environment. Use the following command:

```
cd <JI_install_dir>\bin
.\ea_shutdown
```

Required Information for *ea_shutdown*

In order to execute the *ea_shutdown* command correctly, this program must be able to determine the following:

- **The webMethods JI installation directory:**

  To find the installation directory, *ea_shutdown* requires one of the following:

  - The `-i <JI_install_dir>` command line flag must be used to identify the installation directory, *Or*

  - The EA_ENV environment variable must be set and must identify the location of the *environment.ccf* file, which contains a definition of EA_HOME.
- **Port Numbers of the Server Components:**

  The port numbers for each of the server components are determined as follows:
  - **Environment Manager**
    Command line option `-s <EnvMgr_port>`
    The value of the EA_ENVMGR parameters in the *environment.ccf* file.
    The value of the EA_ENVMGR environment variable.
  - **Resource Server**
    Command line option `-p <RMI_port>`
    The value of the `config.rmiport` parameter in *ressvr.cfg*, the default Resource Server configuration file. The *ressvr.cfg* file is located in the *<JI_install_dir>/config* directory.
  - **Resource Database Server**
    Command line option `-Q <DB_port>`
    The value of the DB_TCP_PORT parameter in *environment.ccf* file.
    The value of the DB_TCP_PORT environment variable.
    The default Resource Database server listen port (`30000`)
- **Java command:**
  Because *ea_shutdown* executes underlying Java programs to start the webMethods JI environment, the correct Java Virtual Machine (JVM) executable (*java* in UNIX or *java.exe* in Windows) must be located.

  The location of the JVM is located by using either:
  - The value of the `javacmd` parameter in *envmgr.cfg*, the default Environment Manager configuration file. The location of the *envmgr.cfg* file is assumed to be the *<JI_install_dir>/config* directory.
  - If the *envmgr.cfg* file is not found or the `javacmd` parameter is not set, then `java` is used as a default command line. If this option is used, the JVM executable must be in the PATH environment variable.

Allowed Command Line Parameters for the *ea_shutdown* Command

Parameter	Description
`-i <JI_install_dir>`	The webMethods JI installation directory.
`-s <EnvMgr_port>`	The Environment Manager server port.
`-p <RMI_port>`	The Resource Server RMI port.
`-Q <DB_port>`	The Resource Database server listen port.
`-e`	Stops only the Environment Manager.
`-r`	Stops only the Resource Server.
`-D`	Stops only the Resource Database server.
`-x`	Enable debugging output.
`-h`	Show help.

Note: For more information about the ea_shutdown command line interface, see the *webMethods JI Supplemental Reference Guide*.

Examples for Using *ea_shutdown*

The following example shuts down an Environment Manager, a Resource Server, and the Resource Database server:

```
ea_shutdown -i d:\EA2000 -s 16024 -p 10024 -Q 16524
```

In this example, the webMethods JI installation location is identified as d:\EA2000 using the -i parameter.

The following port numbers are also identified:

- Environment Manager: `-s 16024`
- Resource Server: `-p 10024`
- Resource Database server: `-Q 16524`

Often, the Resource Database server is shared among several environments and should not be shut down when the Environment Manager and Resource Server are shut down. This can be done by executing the *ea_shutdown* command using the -e and -r command line options. This instructs *ea_shutdown* to only shutdown the Environment Manager (-e) and Resource Server (-r):

```
ea_shutdown -i d:\EA2000 -e -r
```

Because this example does not use the -s <EnvMgr_port> and -p <RMI_port> command line options to set the port numbers for the Environment Manager and Resource Server, it assumes that the port numbers can be determined by *ea_shutdown* using the *environment.ccf* (or EA_ENVMGR environment variable) and *ressvr.cfg* files.

Using *ea_start* and *ea_shutdown* with Multiple Server Components

If multiple environments are executed from the same installation or on the same machine, it may be easy to forget which port numbers were used to launch each of the server components.

The following UNIX shell script is a simple way to get around this problem:

```
#!/bin/sh

ENVMGR_PORT=4154
RMI_PORT=2000
DB_PORT=10020

# JI environment control script

# The script assumes that <JI_install_dir>/bin is in the PATH

# Must pass in the command as the first parameter
if [ -z "$1" ]
then
    echo ""
    echo "Usage: $0 <command>"
    echo ""
    echo "  <command>:"
    echo "    start"
    echo "    shutdown"
    echo "    status"
    echo ""
    exit 1
```

```
if

COMMAND=$1

case $COMMAND in
"start")
  ea_start -Q $DB_PORT -p $RMI_PORT -s $ENVMGR_PORT
  ;;

"shutdown")
  ea_shutdown -Q $DB_PORT -p $RMI_PORT -s $ENVMGR_PORT
  ;;

"status")
  ea_status -Q $DB_PORT -p $RMI_PORT -s $ENVMGR_PORT
  ;;

*)
  echo "ERROR: \"$COMMAND\" is an unknown command"
  exit 1
  ;;
esac

exit 0
```

The following PERL program performs the same actions as the above shell script:

```
#!/usr/local/bin/perl
# env1.pl: Control the JI environment

$envmgr_port = 4154;
$rmi_port    = 2000;
$db_port  = 10020;

#######################################################################
# Check if the command is present
#######################################################################
$argc=@ARGV;
if ($argc == 0)
    {
    print "usage: $0 <command>\n";
    print "  <command>:\n";
    print "    start\n";
```

```
    print "    shutdown\n";
    print "    status\n";
    exit 1;
    }

$command = $ARGV[0];
$_ = $command;
if (/start/) {
    system("ea_start -s $envmgr_port -p $rmi_port -Q $db_port");
    }
elsif (/shutdown/) {
    system("ea_shutdown -s $envmgr_port -p $rmi_port -Q $db_port");
    }
elsif (/status/) {
    system("ea_status -s $envmgr_port -p $rmi_port -Q $db_port");
    }
else {
    print "ERROR: \"$command\" is an unknown command\n";
    exit 1;
    }

exit 0;
```

The file name of the script or PERL program can be the name of the environment to be started.

For example, if the name of the script is “env1”, start the environment as follows:

```
$env1 start
Checking if the database daemon is already running . . .
Starting the database daemon . . .
Checking if the database daemon started . . .
Database daemon was started on port 10020.
Checking if the Resource Server is already running . . .
Starting the Resource Server . . .
Checking if Resource Server started . . .
Resource Server was started, RMI port = 2000.
Checking if the Environment Manager is already running . . .
Starting the Environment Manager . . .
Checking if Environment Manager started . . .
Environment Manager was started, listen port = 4154.
$
```

If more environments are required, copy the script or PERL program to a new name and edit the port parameters.

If the Resource Database is to be shared, the script can be easily edited to just start the Resource Server and Environment Manager.

For example, in the shell script, change the *ea_start* and *ea_shutdown* lines from:

```
ea_start -Q $DB_PORT -p $RMI_PORT -s $ENVMGR_PORT
ea_shutdown -Q $DB_PORT -p $RMI_PORT -s $ENVMGR_PORT
```

to

```
ea_start -r -e -p $RMI_PORT -s $ENVMGR_PORT
ea_shutdown -r -e -p $RMI_PORT -s $ENVMGR_PORT
```

Similar changes can be made if the Resource Database and Resource Server are to be shared among several Environment Managers:

```
ea_start -e -s $ENVMGR_PORT
ea_shutdown -e -s $ENVMGR_PORT
```

Pre- and Post-Start and Shutdown Scripts

The *ea_start* and *ea_shutdown* programs can execute user supplied shell scripts (UNIX) or batch files (Windows). These can be used to delete log files. The scripts are located in the *<JI_install_dir>/config/ActionScripts* directory.

The following scripts are used:

Script/Batch File	Description
PreStart/PreStart.bat	Executed before any servers are started.
PostStart/PostStart.bat	Executed after all servers are started.
PreShutdown/PreShutdown.bat	Executed before any servers are shutdown.
PostShutdown/PostShutdown.bat	Executed after all servers are shutdown.

In order to use pre- and post-start and shutdown scripts, create the *ActionScripts* directory in *<JI_install_dir>/config* and create the script or batch files that are included in that directory.

Environment Managers

The Environment Manager is a process that runs on a webMethods JI server. It provides a central management and control point for webMethods JI server-side processing. The Environment Managers provide the following functionality:

- Process management, including:
  - Startup and shutdown of JCluster processes.
  - Startup and shutdown of appropriate Resource Servers and other resource providers.
- Central access point for webMethods JI client applications.
- Distribution of client requests to the most available JCluster running on any Environment Manager within its group.

Typically, there should be one Environment Manager per server machine. If there are more than one Environment Managers on one machine, they should be configured to be in different groups.

For information about configuring Environment Managers, see "Configuring Environment Managers" on page 51.

Starting the Environment Manager

The Environment Manager can be started using the *ea_start* command. See "Starting and Stopping the webMethods JI Environment" on page 32. Alternatively, the Environment Manager can be started using the *ea_start -e* command, as described below:

UNIX:

```
cd <JI_install_dir>/bin
./ea_start -e
```

Windows:

```
cd <JI_install_dir>\bin
.\ea_start -e
```

The Environment Manager can also be launched using shortcuts created during installation.

Using Batch Mode in UNIX

In order to detach the Environment Manager process from the terminal window and to allow the user to log off without killing the application, it is recommended that the Environment Manager be started in "batch" mode.

To run the Environment Manager in the background, type the following at the command prompt:

```
batch
at> ./ea_envmgr
at> <CTRL-D>
```

Note: `at>` is used in the above example to represent the standard batch prompt.

Required Information at Startup

Environment Managers require the following information in order to launch:

- **Configuration Name:**

  Represents the name of the Environment Manager configuration, as defined in the Resource Database. This information is defined in the Environment Manager's configuration file using the `configname` parameter. A configuration record must exist in the Resource Database and the name of the configuration must match the name used for the `configname` parameter in the Environment Manager's configuration file. For more information about Environment Manager configurations, see "Environment Manager Configurations" on page 66.
- **Resource name:**

  The name of the Resource Database that contains the configuration information. This information can be included in the `resourcename` parameter of the Environment Manager's configuration file. It can be passed at the command line using the `-r resourcename` command line option. The command line option overrides the setting in the configuration file.
- **Installation Directory:**

  The directory into which webMethods JI has been installed. This information can be included in the `installdir` parameter of the Environment Manager's configuration file, or can be passed at the command line using the `-i install_dir` command line option. The command line option overrides the setting in the configuration file.

Note: The Resource Server and Resource Database must be running prior to starting an Environment Manager. For more information about starting the Resource Server, see "Starting the Resource Server" on page 58. For more information about starting the Resource Database, see "Starting the Resource Database" on page 63.

Allowed Command Line Parameters

These options can be included at the command line when the Environment Manager is started:

Parameter	Description
`-f config_file`	Identifies the name of the configuration file for the Environment Manager. If this parameter is not used, the default value, *envmgr.cfg*, is used. The path can be identified in this parameter as well. If no path is identified, the path identified in the `install_dir` parameter is used.
`-i install_dir`	Identifies the path to the installation of webMethods JI.
`-d rmiport`	Identifies the port for Remote Method Invoke (RMI) activity for the Environment Manager and the Resource Server. If the Resource Server and Environment Manager are running on the same machine, the value of this port can match the value for the `config.rmiport` parameter in the Resource Server configuration file, or the `-p rmiport` command line option for the Resource Server. If the same RMI port is used for both of these and the Resource Server disconnects, the Environment Manager has to be restarted. This option overrides the value of the `rmiport` parameter in the Environment Manager's configuration file, and the rmiport record in the Resource Database for the Environment Manager's configuration.
`-p`	Identifies the port on which the Environment Manager listens for client connections. This is the host and port the client should identify as the Environment Manager's. This option overrides the value of the `serverport` parameter in the Environment Manager's configuration file, and the `serverport` record in the Resource Database for the Environment Manager's configuration.

Parameter	Description
`-r resourcename`	Identifies the name of the Resource Database that contains configuration information for the Environment Manager. The location of the database is defined in the Resource Server node in the Configuration Manager. The value of this parameter must correspond to the name of the appropriate resource listed in the Configuration Manager. This option overrides the value of the `resourcename` parameter in the Environment Manager's configuration file.
`-l logfile`	Identifies the name of the logfile for the Environment Manager. Optionally, the path can be identified in this parameter as well. If no path is identified, the default value, *<JI_install_dir>/logs*, is used.
`-n`	Determines the level of debug logging. Allowable values for *n* are `0` through `9`, with `0` representing minimal logging and `9` representing the most verbose logging.

There are two command line options that do not start the Environment Manager. They return information to the *envmgr_stdout.txt* log file, located in *<JI_install_dir>/logs*:

- `-v` causes version information about the Environment Manager to be written to the *envmgr_stdout.txt* file
- `-h` causes help information to be written to the *envmgr_stdout.txt* file.

Example Environment Manager Startup Command Line

Following is an example of a command line that could be used to start the Environment Manager:

```
./ea_start -e -d 14404 -f /usr/local/ea/config/dev_envmgr.cfg -9
```

This example creates an Environment Manager that listens for client connections on port 14404, uses the configuration file *dev_envmgr.cfg* located in the directory */usr/local/ea/config*, and logs information to the default log file at the most verbose level (9).

Note: Any information identified in both the `dev_envmgr.cfg` configuration file and in command line options (such as `serverport)`, will be overridden by the command line options used in this example.

Environment Manager Configuration File

Many of the above command line options can also be included in the Environment Manager's configuration file. As long as the file location is identified at the command line when the Environment Manager is started, this file can have any name and can be stored in any location. If no filename is provided, the default name of *envmgr.cfg* will be used.

Any information contained in the Environment Manager's configuration file is overridden by command line options, and information in the configuration file overrides configuration information stored in the Resource Database.

For more information about the Environment Manager configuration file, see "Configuring Environment Managers" on page 51.

Starting Multiple Environment Managers

Multiple Environment Managers can be run on the same machine, and from the same installation of webMethods JI.

Note: Because of webMethods JI licensing restrictions, multiple Environment Managers can be run under a single IP address. However, multiple Environment Managers cannot be run on a single host using different IP addresses (for example, if the host has more than one network card). This is considered by the environment to be two separate servers even though they may be on the same host machine.

Running Multiple Environment Managers on the Same Machine

Multiple Environment Managers can be run on the same machine. However, because of the way that webMethods JI balances load, there is no reason to do this unless those Environment Managers belong to different load balancing groups. An example of two load balancing groups could be: one for production and one for development. Two different Environment Managers could be on one machine, with each belonging to their separate load balancing group.

Multiple Environment Managers can be launched on the same machine by installing the product twice and configuring the Environment Managers correctly to use different ports. Two ports are required by each Environment Manager, one used by the Environment Manager to listen for client connections,

and the other to facilitate Remote Method Invocation (RMI). These ports can be set in the Configuration Manager using the client connect port and RMI port records under the Environment Manager configurations node. They can also be set in the Environment Manager configuration file using the serverport and rmiport parameters, or can be passed at the command line using the `-p serverport` and `-d rmiport` command line options.

Starting Multiple Environment Managers from the Same Installation

Multiple Environment Managers can also be started from the same installation.

This can be done in one of two ways:

- **Using different Environment Manager configuration files:**

  This can be accomplished by copying the <*JI_install_dir*>/*config* directory to create a second folder of configuration files. The location of the Environment Manager's configuration file is then passed in at the command line when the Environment Manager is started, using the `-f config_file` command line option.

  For example, there can be two different directories for configuration files, /*JI*/*dev_config* and /*JI*/*prod_config*. Each directory would contain their own copy of the *envmgr.cfg* file, which would set a different configuration and other parameters for each of the Environment Managers. Port numbers and other information is then defined in either the configuration file or the Environment Manager configuration in the Resource Database. Set the `serverport` and `rmiport` parameters in each Environment Manager configuration file (*envmgr.cfg*) The Environment Managers can then be started using commands at the command prompt.

Note: You must also copy the environment.ccf file to the same directory and modify it to use the proper `serverport` and `rmiport`. ea_start and ea_shutdown use the environment.ccf file to determine which port numbers to use.

Start the Environment Managers at the command prompt using the following commands.

```
./ea_start -e -f /JI/dev_config/envmgr.cfg
```

and

```
./ea_start -e -f /JI/prod_config/envmgr.cfg
```

To log any Environment Manager information, set the `logfile` parameter to unique filenames in each Environment Manager's configuration file. The

same standard out and standard error files (*envmgr_stdout.txt* and *envmgr_stderr.txt*) are used.

This method allows the use of *ea_start* and *ea_shutdown* to start multiple environments by simply changing the value of the EA_ENV environment variable.

For example:

```
export $EA_ENV=/JI/dev_config
ea_start
```

starts the development environment, and

```
export $EA_ENV=/JI/prod_config
ea_start
```

starts the production environment.

- **Passing the port number and other information at the command line when the Environment Manager is started:**

  Using this method, both Environment Managers would use the same configuration file and therefore the same Environment Manager configuration. However, they could be launched in separate load balancing groups if the `-g groupname` parameter is used at the command line.

  For example, these Environment Managers could be started using the following commands:

  ```
  ./ea_start -e -p 30001 -g dev_group -l logfile1
  ```

  and

  ```
  ./ea_start -e -p 30002 -g prod_group -l logfile2
  ```

  These command line examples assume that the default environment manager, *envmgr.cfg*, is used, and this file is in the default location, *<JI_install_dir>/config*. Different logfile names are passed at the command line using the `-l logfile` option. The different names prevent the Environment Managers from overwriting each other's logging information.

Shutting Down the Environment Manager

The Environment Manager can be shut down using the *ea_shutdown* command. For more information, see "Stopping the Environment" on page 37.

Shutting Down the Environment Manager from the System Monitor

Environment Managers can also be shut down from within the System Monitor. To shut down an Environment Manager, select the Environment Manager in the Tree view and select **Action** > **Environments** > **Kill Environment**. For more information about the System Monitor, see Chapter 13 - "System Monitor" on page 515.

Configuring Environment Managers

Environment Manager configuration is performed in two separate places:

- Configuration Files.
- Environment Manager configurations in the Resource Database.

Configuration Files

Individual instances of Environment Managers can be configured using a text-based configuration file that identifies ports, the Environment Manager's load balancing group and configuration, and other information. Many of the parameters allowed in this file can also be passed at the command line when the Environment Manager is started. For more information about passing command line options to the Environment Manager, see "Starting the Environment Manager" on page 44.

Configuration File Location

By default, the configuration files are located in *<JI_install_dir>/config*, and the default name of the configuration file is *envmgr.cfg*. These defaults can be overridden by using -f config_file, which identifies the name and, optionally, the location of the Environment Manager configuration file.

Editing the Configuration File

Generally, the Environment Manager configuration file should be edited from within the Configuration Manager's interface. See "Configuring Environment Managers" on page 464 for information about editing this file from within the Configuration Manager.

Note: The configuration file can also be edited by any text editor.

The following parameters are set by default in the Environment Manager configuration file and are required:

Parameter	Description
`configname`	The name of the Environment Manager configuration as it appears in the Resource Database. **Note:** This parameter must be set to an existing configuration in the Resource Database otherwise the Environment Manager fails to start. Default: `eaconfig`
`resourcename`	The name of the resource database that contains configuration information for the Environment Manager. The location of the database is defined in the Configuration Manager. **Note:** The value of this parameter must correspond to the name of the appropriate resource listed in the Configuration Manager. Default: `eardb://ea`
`loglevel`	The level that information is logged to the logfile. Valid entries are `0` through `9`, with `0` representing no logging and `9` representing the most verbose logging. **Note:** Setting the loglevel to `0` turns logging off. Default: `0`

Note: These parameters can be overridden by command line options.

For information about additional configuration parameters that are supported in the Environment Manager configuration file, see Chapter 4 - "Configuration Files" on page 127 of the *Supplemental Reference Guide*.

Environment Manager Configurations in the Resource Database

The bulk of Environment Manager configuration is done in the Resource Database, under the Environment Manager configurations node of the Configuration Manager. The configurations are created within this node. Environment Manager configurations can apply to one Environment Manager or

multiple Environment Managers, and set a number of options regarding how load is balanced in the environment. For more information, see "Load Balancing" on page 65.

Environment Manager Redundancy

Multiple Environment Managers can be used to provide redundancy. This enables clients to connect to the webMethods JI environment by accessing a different Environment Manager if there is a failure. When connecting to the webMethods JI environment, clients send a Request for Service (RFS) to the Environment Manager by identifying the host and port of the Environment Manager. To provide redundant behavior, clients can be developed to use multiple host/port combinations and to connect to the first Environment Manager that responds to their request. This allows the clients to access webMethods JI even when the Environment Manager they would normally connect to is no longer running.

resource.cache file

The Environment Manager creates a local file named *resource.cache*. This file is located in the directory the Environment Manager was started in and contains the last known address (host and port) for a given resource. This file is used by the Environment Manager as a backup resource location mechanism.

To find network resources, the Environment Manager first queries the network for the resource. If the resource is not found, it then checks its local memory cache to determine if the resource has been located in the past. If the resource still has not been located, the Environment Manager looks in the *resource.cache* file as a last resort. To permanently flush cached resources, delete the *resource.cache* file.

Multicast Channel Information

Environment Managers and Resource Servers use multicasting technology to locate resources and other server components within the webMethods JI environment. The following multicast channels are used within the webMethods JI server framework for inter-process communication and location of components and resources:

- **Global resource location channel:** Used for location of resource providers for specific resources on a network-wide basis.
- **Global Environment Manager channel:** Environment Managers use this channel to locate other Environment Managers within their group.
- **Environment Manager group channel:** This multicast group is used within a group of related Environment Managers to balance load among their

JClusters. All responses to resource location requests are returned to a requesting component's multicast group to avoid unnecessary multicast traffic.

Note: Typically, the user does not have to configure any information to use multicast sockets with webMethods JI.

Proxy Server

Because most networks only provide for multicasting within the same sub-net, a Resource Server can be established as a "proxy" server. This facilitates load balancing between Environment Managers in the same group that are running on different sub-nets. See "Proxy Server" on page 76 for details.

Environment Manager Logging

Environment Manager logging is set in the configuration file for the Environment Manager using the loglevel parameter, it can also be set at the command line using the `-n` command line option (where n represents a number from 0-9).

Either of these options can be set to any number from 0 through 9, with 0 representing minimal logging and 9 representing the most verbose logging.

The default location for Environment Manager logging is *<JI_install_dir>/logs* and the default name is *envmgr.log*. The name and location of the logfile can be overridden by the logfile parameter in the Environment Manager's configuration file, or the -l logfile command line option. These settings determine both the name and, optionally, the directory location where the log files are saved. For more information about log files, see Chapter 5 - "Logging Information" on page 155 of the *Supplemental Reference Guide*.

JClusters

JClusters are processes that run on the same server machine as an Environment Manager. Each Environment Manager manages one or more running JClusters. They are used to control individual service instances, which run as threads within the JService process. The number of JClusters per Environment Manager varies depending on the number of processors in the Environment Manager host machine and the number of services running within the JCluster. The number of JClusters per Environment Manager is set in the Configuration Manager (see "Environment Manager and JCluster Configuration" on page 479).

> **Note:** It is recommended that a small number of JClusters be run on a single machine. On a machine with four processors, multiple JClusters could be running.

Using the JVM's -Xmx<size> Command Line Option

The -Xmx<size> option, passed at the command line when the Java Virtual Machine (JVM) is executed, sets the maximum heap size for the JVM. In environments where JClusters support approximately 40 services or more, the JVM may need to use a larger heap size than the default size.

> **Note:** Java services often use 600 Kilo Bytes to 1 MegaByte each, although this varies with the number of objects defined in the service.

There is no harm in setting a high heap size value; values of -Xmx100m to -Xmx200m are usually appropriate. To instruct the Environment Manager to launch JCluster JVMs using this command line option, edit the javacmd parameter in the Environment Manager's configuration file to include the command line option along with the JVM command.

This parameter can be edited in the Configuration Manager (see "Environment Manager Configuration Properties" on page 466), or it can be changed by editing the configuration file itself with a text editor. For information about editing configuration files, see Chapter 4 - "Configuration Files" on page 127 of the *Supplemental Reference Guide*.

JCluster Logging

The level of JCluster logging is determined by the Environment Manager's log level. This level is set using the loglevel parameter in the Environment Manager's configuration file or the `-n` command line option when the Environment Manager is started. Either of these options can be set to any number from 0 through 9, with 0 representing minimal logging and 9 representing the most verbose logging.

The default location for JCluster logging is *<JI_install_dir>/logs*. This location can be overridden by the logfile parameter in the Environment Manager's configuration file, or the -l logfile command line option. This option can be used to determine the location of the Environment Manager's logfile. JCluster logfiles are saved in the same location as their Environment Manager's configuration file. JCluster logfiles use the IP address and port number of the JCluster for their names: *<IP_address>.<port_num>.txt*. For more information about log files, see Chapter 5 - "Logging Information" on page 155 of the *Supplemental Reference Guide*.

Starting JClusters

JClusters are started automatically as they are required to serve client requests. The maximum number that can be running is configured in the Environment Manager configuration in the Configuration Manager. Configurations can set the minimum number of JClusters that are started when the Environment Managers is started. For example, an Environment Manager running on a high performance machine might be set to launch two JClusters automatically upon startup of the Environment Manager.

This way, two JCluster processes are running immediately and are ready to accept client requests. This machine might have a maximum number of JClusters set to four, so that after four JClusters are started, the Environment Manager does not start additional JCluster processes.

JCluster processes can also be started in the System Monitor by selecting the Environment Manager in the Tree view and selecting **Actions** > **Environments** > **Spawn JCluster**. For more information about the System Monitor, see Chapter 13 - "System Monitor" on page 515.

Shutting Down JClusters

If more than the configured minimum number of JClusters are running, the Environment Manager automatically stops the JCluster when it is no longer serving client requests. JClusters can also be shut down from within the System Monitor. To shut down a JCluster, select the JCluster in the Tree view and select **Action** > **JClusters** > **Kill JCluster**. For more information about the System Monitor, see Chapter 13 - "System Monitor" on page 515.

JServices

JServices are started automatically as they are required to serve client requests. The maximum number that can be running is configured in either the Service Master entry in the Resource Database or the Service Detail entry in the Environment Manager configuration in the Configuration Manager. Also, configurations can set the minimum number of JServices that are started in each JCluster when the Environment Manager is started.

Shutting Down JServices

If more than the configured minimum number of JClusters are running, the Environment Manager automatically stops JServices when they are no longer serving client requests. All JService threads running within a JCluster process are killed when the JCluster is killed.

Resource Server

The Resource Server is a process that runs in the webMethods JI server environment. This is used to provide access to various data sources within the webMethods JI environment. Data sources managed by the Resource Server include:

- Resource Databases
- Licensing files

The Resource Server provides information about data sources, such as the Environment Managers and JClusters, to other components in the webMethods JI server environment. The Resource Server also provides data location information to the Configuration Manager.

Note: The Resource Server does not provide information to webMethods JI clients.

There can be one Resource Server per Environment Manager or one Resource Server per Environment Manager group. One Resource Server per Environment Manager is generally best. If the machine the Resource Server is running on fails, all Environment Managers in the group are no longer able to create new services or host connections. At a minimum, a large group of multiple Environment Managers should have two or more Resource Servers to provide redundant failover behavior. If one Resource Server goes down another one is available to serve Environment Manager requests.

Resource Names

Resources are given names and tied to Resource Servers in the Configuration Manager. For example, one instance of the Resource Database server might be serving more than one Resource Database, or multiple instances of the Resource Database server, running on different host machines, might be serving identical, redundant databases. Each database can be given a unique name and can be tied to a single Resource Server or could be tied to multiple Resource Servers.

When an Environment Manager makes a request for a Resource Database or other resource, it sends a multicast message asking for the location of the Resource Server that serves that resource. As a result, it is generally better to use unique names for all webMethods JI resources running on your network.

For redundancy purposes it might be preferred to have two or more identical resources, with each running on different machines and being served by different Resource Servers. For example, two identical Resource Databases could be running on different machines, that are being served by two separate Resource Servers. These Resource Databases could be given the same resource name in the

Configuration Manager. When an Environment Manager makes a request, either Resource Server can respond. Because the Resource Databases are identical, it does not matter which Resource Server responds to the request and which resource is served to the Environment Manager. In this manner, if one of the Resource Databases or Resource Servers fail, the other one can automatically respond to Environment Manager requests without any interference in client/service communication.

Starting the Resource Server

The Resource Server can be started using the *ea_start* command. See "Starting and Stopping the webMethods JI Environment" on page 32. Alternatively, the Resource Server may be started using the following command:

```
cd <JI_install_dir>\bin
.\ea_start -r
```

> **Note:** The Resource Server can also be launched using shortcuts in Windows, if they were created during installation.

Using Batch Mode in UNIX In order to detach the Resource Server process from the terminal window and to allow the user to log off without killing the application. It is recommend that the Resource Server be started in "batch" mode. To run the Resource Server in the background, type the following at the command prompt:

```
batch
at> ./ea_start -r
at> <CTRL-D>
```

> **Note:** `at>` is used in the above example to represent the standard batch prompt.

Required Information at Startup

Resource Servers require Installation Directory information in order to launch. Installation Directory information can be included in the installdir parameter of the Resource Server configuration file. It can also be passed at the command line using the -i install_dir command line option. The command line option overrides the setting in the configuration file.

Allowed Command Line Parameters

The following options can be included at the command line when the Resource Server is started:

Parameter	Description
`-f config_file`	Identifies the name of the configuration file for the Resource Server. If this parameter is not used, the default value, *ressvr.cfg*, is used. Optionally, the path can be identified in this parameter as well. If no path is identified, the path identified in the `install_dir` parameter is used.
`-i install_dir`	Identifies the path to the installation of webMethods JI.
`-p rmiport`	Identifies the port for Remote Method Invoke (RMI) activity for the Resource Server and the Environment Manager. If the Resource Server and Environment Manager are running on the same machine, the value of this port can match the value for the `rmiport` parameter in the Environment Manager configuration file or the `-d rmiport` command line option for the Environment Manager If the same RMI port is used for the Environment Manager and the Resource Server, and the Resource Server goes down, the Environment Manager has to be restarted. This option overrides the value of the `rmiport` parameter in the Resource Server configuration file.
`-l logfile`	Identifies the name of the logfile for the Resource Server. Optionally, the path can be identified in this parameter as well. If no path is identified, the default value, *<JI_install_dir>/logs*, is used.
`-n`	Determines the level of debug logging. Allowable values for *n* are 0 through 9, with 0 representing minimal logging and 9 representing the most verbose logging.

`-r`	Causes RMI debugging information to be logged to the log file.

There are two command line options that do not start the Resource Server. They return information to the *ressvr_stdout.txt* log file, located in *<JI_install_dir>/logs*.

- `-v` causes version information about the Resource Server to be written to the *ressvr_stdout.txt* file
- `-h` causes help information to be written to the *ressvr_stdout.txt* file.

Example Resource Server Startup Command Line

The following is an example of a command line that could be used to start the Resource Server:

```
./ea_start -r -p 14404 -f /usr/local/ji/config/dev_ressvr.cfg -9
```

This example creates a Resource Server that uses port 14404 as its RMI port. It uses the configuration file *dev_ressvr.cfg* located in the directory */usr/local/ea/config*, and logs information to the default log file at the most verbose level (9).

Note: Any information identified in both the *dev_ressvr.cfg* configuration file and command line options, such as `rmiport`, will be overridden by the command line options used in this example.

Resource Server Configuration File

Many of the above command line options can also be included in the Resource Server configuration file. This file can have any name and be stored in any location, as long as the file location is identified at the command line when the Resource Server is started. If no filename is provided, the default name of *ressvr.cfg* will be used.

Any information contained in the Resource Server configuration file is overridden by command line options. For more information about the Resource Server configuration file, see Chapter 4 - "Configuration Files" on page 127 of the *Supplemental Reference Guide*.

Starting Multiple Resource Servers

Multiple Resource Servers can be run on the same machine, and from the same installation of webMethods JI.

Running Multiple Resource Servers on the Same Machine

Multiple Resource Servers can be launched on the same machine by installing the product twice and configuring the Resource Servers correctly so that they do not use the same port. One port is required and used by each Resource Server to facilitate Remote Method Invocation (RMI). This port can be set in the Resource Server configuration file using the config.rmiport parameter, or can be passed at the command line using the -p rmiport command line option.

Starting Multiple Resource Servers from the Same Installation

Multiple Resource Servers can also be started from the same installation. This can be done in one of two ways:

- **Using different Resource Server configuration files:**

  This can be accomplished by copying the *<JI_install_dir>/config* directory to create a second directory containing configuration files. The location of the configuration file is then passed in at the command line when the Resource Server is started, using the *-f config_file* command line option.

  For example, you could have two different directories containing configuration files, */JI/dev_config/ressvr.cfg* and */JI/prod_config/ressvr.cfg*. Each directory would contain a copy of the *ressvr.cfg* file, which would set a different RMI port and other information. The Resource Server can then be started by using the following commands at the command prompt:

  ```
  ./ea_start -r -f /JI/dev_config/ressvr.cfg
  ```

  and

  ```
  ./ea_start -r -f /JI/prod_config/ressvr.cfg
  ```

  To log any Resource Server information, set the `logfile` parameter to unique filenames in each Resource Server configuration file. The same standard out and standard error files (*ressvr_stdout.txt* and *ressvr_stderr.txt*) are used.

  This method also allows the use of *ea_start* and *ea_shutdown* to start multiple environments by simply changing the value of the EA_ENV environment variable.

For example:

```
export $EA_ENV=/JI/dev_config
ea_start
```

starts the development environment, and

```
export $EA_ENV=/JI/prod_config
ea_start
```

starts the production environment.

- **Passing the port number and other information at the command line when the Resource Server is started:**

  Using this method, both Resource Servers would use the same configuration file. For example, these Resource Servers could be started using the following commands:

  ```
  ./ea_start -r -p 30001 -l logfile1
  ```

  and

  ```
  ./ea_start -r -p 30002 -l logfile2
  ```

  These command line examples assume that the default Resource Server configuration file, *ressvr.cfg*, is used, and this file is in the default location, *<JI_install_dir>/config*. Different logfile names are passed at the command line using the `-l logfile` option, so that the Resource Servers do not overwrite each other's logging information.

Shutting Down the Resource Server

The Resource Servers can be shut down using the ea_shutdown command. For more information, see "Stopping the Environment" on page 37.

Proxy Resource Server

Because most networks only provide for multicasting within the same sub-net, a Resource Server can be established as a "proxy" server to facilitate load balancing between Environment Managers in the same group running on different sub-nets. See "Proxy Server" on page 76 for details.

The Resource Database

The webMethods JI Resource Database, implemented using the H2 Database Engine, is used to store configuration information and runtime executable Java service code.

Note: For C services that were developed using previous versions of webMethods JI, the Resource Database stores Protocol Agent definitions, screen definitions, and service metadata.

The following information is stored in the Resource Database:

- Configuration information for the Environment Manager, JCluster, and JServices.
- webMethods JI maps and service class files (maps and service class files can also be stored on the local file system rather than in the database).
- Host connection information.
- Information for "C" services (services used with the compatibility kit to integrate webMethods JI 2.x "C" service code):
  - Protocol Agent definitions
  - Screens and fields
  - Service metadata such as methods and parameters

Note: The best deployment of Resource Databases would be to have one database serving each load balancing group in the environment, except for the issue of redundancy (see "Resource Database Redundancy" on page 64).

Starting the Resource Database

The Resource Database can be started using the *ea_start* command. See "Starting and Stopping the webMethods JI Environment" on page 32. Alternatively, the Resource Database server can be started using the following command:

UNIX:

```
./ea_dbsvr -i <JI_install_dir> -p <port>
```

Windows:

```
.\ea_dbsvr -i <JI_install_dir> -p <port>
```

Where <install_dir> represents the directory in which webMethods JI was installed, and <port> represents the port on which the Resource Database server is running. If no port is specified, the default port, 30000, is used.

When using window shortcuts, the <JI_install_dir> option is passed from within the shortcut. If the installation directory changes, you have to edit the shortcut. Also, if you want to use a port number other than the default of 30000, you can edit the shortcut to pass the new port number on the command line.

For a more complete description on the ea_dbsvr command interface, see Chapter 1 - "webMethods JI Commands" on page 11 of the *Supplemental Reference Guide*.

Shutting Down the Resource Database

The Resource Database can be shut down using the ea_shutdown command. For more information, see “Stopping the Environment” on page 37. Alternatively, the Resource Database server can bu shut down using the following command:

UNIX:

```
./ea_dbshutdown -i <JI_install_dir> -p <port>
```

Windows:

```
.\ea_dbshutdown -i <JI_install_dir> -p <port>
```

Where <install_dir> represents the directory in which webMethods JI was installed, and <port> represents the port on which the Resource Database server is running. If no port is specified, the default port, 30000, is used.

Resource Database Redundancy

Using multiple, identical Resource Databases servers allows for redundant fail-over protection within the webMethods JI environment. For example, there can be two identical Resource Databases servers in the production group, each running on different machines. If one of the machines fails, the other Resource Database server can be used in its place.

For more information about redundant resources, see “Resource Names” on page 57.

Load Balancing

webMethods JI includes the ability to configure the environment for load balancing purposes. This load balancing is extremely flexible, using multiple parameters to tune the load balancing, along with combining Environment Managers into groups to control the pool of Environment Managers that respond to client requests. Additionally, Environment Manager configurations are used to set different load balancing parameters for each machine or each class of machines.

Load Balancing Groups

The webMethods JI environment consists of one or more load balancing groups. For example, there could be a production group that is used to respond to client requests from customers or other end users. There might also be a development group that is used for future project development. Finally, there might be a group dedicated to training or support.

When an Environment Manager receives a request from a client, the Environment Manager uses multicasting technology to determine which Environment Manager in its group is best able to serve the client request. Different load balancing configuration parameters allow the determination of whether the Environment Manager should connect the client to the first Environment Manager that responds to its load balancing request, or wait for a configured amount of time and then determine which Environment Manager is best able to handle the load.

Configuring Load Balancing Groups

There is little configuration information related to load balancing groups. Load balancing parameters are actually determined when Environment Manager configurations are configured. The only purpose of load balancing groups is to determine which Environment Managers running on the network belong to the same group and therefore respond to multicasted requests from Environment Managers within that group.

Therefore, the only configuration tasks associated with Environment Manager load balancing groups are to set the group for each instance of Environment Managers running in your network and to set the load balancing timeout to a non-zero value.

This is accomplished in one of three ways:

- In the Resource Database, when Environment Manager configurations are defined. Because each Environment Manager subscribes to a configuration, the Environment Manager can determine its group based on the group record in its configuration in the Resource Database.
- In the Environment Manager's configuration file, using the `groupname` parameter. If this setting is used, it overrides the group record in the Resource Database.
- At the command line using the `-g groupname` command line option. This option overrides both the setting in the Environment Manager's configuration file and in the Resource Database.

Environment Manager Configurations

Environment Manager configurations, stored in the Resource Database, are where much of the load balancing tuning is performed. Each Environment Manager within webMethods JI subscribes to a configuration, and that configuration contains load balancing information such as the load balancing group, minimum and maximum number of JClusters, timeouts, and other information used by Environment Managers that subscribe to this configuration. Some of these entries can be overridden by settings in the Environment Managers' configuration files or at the command line when the Environment Managers are started.

The deployment of Environment Manager configurations is highly flexible; there can be a one-to-one relationship between configurations and Environment Managers, or multiple Environment Managers can subscribe to the same configuration. For example, if the environment consists of a number of different-sized machines, you may want to create one configuration for each machine. On the other hand, if the environment consists of a few different classes of machines, some large machines with multiple processors and some small machines with single processors, you may create two configurations, one for the large machines and one for the smaller machines. This centralizes your ability to tune load balancing parameters by only having to change configuration settings for a minimal number of configurations, rather than one configuration for each of your machines.

The following diagram illustrates such an arrangement:

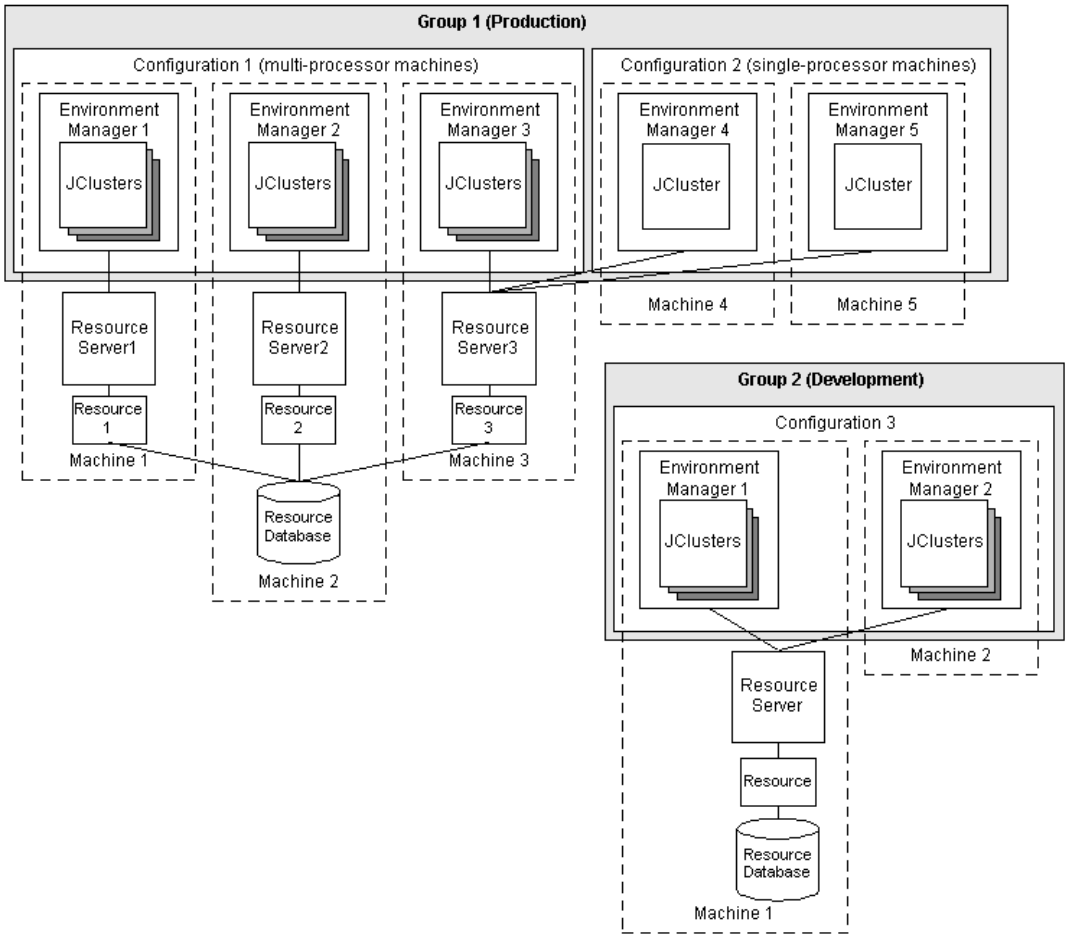


Figure 3. Deployment of Environment Manager Configurations

- **Groups:**

  This diagram shows a webMethods JI environment that consists of two groups, one for production and one for development. The production group has two configurations, one for multi-processor machines and one for single-processor machines.

- **Resource Servers and Resource Databases:**

  The diagram shows one possible arrangement of Resource Servers and Resource Databases. Each multi-processor machine in the production environment has its own Resource Server with unique resources. However, each of these resources is actually configured to point to the same Resource Database. This arrangement allows for multiple, redundant Resource Servers, so if one machine fails, clients can still locate their requested services. It also provides for a central database which facilitates simple management by only having to make changes to one database.

- **JClusters:**

  Each configuration uses a different number of JClusters. The configuration for the multi-processor machine is configured to use multiple JClusters, while the configuration for the single-processor machines only uses one JCluster. Therefore, when a client request is made, load is balanced correctly among the high performance and low performance machines.

  A special load balancing algorithm is used to provide proper and consistent load balancing. The load balancing uses the following load criteria to evaluate the load on each individual JCluster:

  - Quantity of JServices compared to configured maximum.
  - Quantity of webMethods JI client connections to JServices.

Configuring Environment Manager Configurations

The Environment Manager's configuration is specified in the config parameter in the Environment Manager configuration file. Environment manager configurations are created and configured in the Configuration Manager, and this information is stored in the Resource Database.

For information about using the Configuration Manager to create and configure configurations, see "Environment Manager and JCluster Configuration" on page 479.

The following items are configured:

- **Configuration Name:**

  The name of the configuration. This name is supplied by the user and can be any name, although all configuration names should be unique throughout the webMethods JI environment. This name should correspond to the `configname=<configname>` parameter in the configuration file for each Environment Manager that subscribes to this configuration.
- **Load Balancing Group Name:**

  The name of the Load Balancing group the Environment Managers using this configuration subscribes to. For more information about Load Balancing Groups, see "Load Balancing Groups" on page 65.
- **Port number for incoming client requests:**

  The Environment Manager's listen port.
- **RMI Port number:**

  The port number of the Environment Manager's RMI (Remote Method Invocation) registry. If the Environment Manager is running on the same machine as a Resource Server, both applications can share the same RMI port.
- **Minimum number of concurrent JCluster processes:**

  The minimum number of JClusters that should be started when the Environment Managers are started.

- **Maximum number of concurrent JCluster processes:**

  The maximum number of JClusters that the Environment Managers can start.
- **Maximum JServices per JCluster**.

  The maximum number of JServices that can be executed within each JCluster.
- **Load balancing timeout:**

  The amount of time, in milliseconds, the Environment Manager waits for load balancing requests to return on the multicast channel. When this timeout has been reached, the Environment Manager uses its load balancing information, along with the information retrieved from any or all of the other Environment Managers in the load balancing group, to route the client's Request for Service to the appropriate JCluster.

  This property provides load balancing tuning. A larger value, such as 2000 milliseconds, provides for better load balancing. A smaller number, such as 100 milliseconds, provides faster performance for the client's Request for Service, but provides less precise load balancing. A value of 0 (zero) disables load balancing between the other Environment Managers, meaning only the primary Environment Manager (the one to which the client connects) runs the services.
- **Load balancing options:**

  This property provides two options:
  - Use “fast” load balancing: After a load balancing request, the Environment Manager uses information from the first respondent to determine load balancing.
  - Use “accurate” load balancing: After a load balancing request, the Environment Manager waits the full Load Balancing Timeout duration, and then uses information from all respondents to determine load balancing.
- **Load balancing cache expiration time:**

  The amount of time, in milliseconds, that load balancing information is cached by the Environment Manager.
- **Rollover Timeout:**

  Maximum amount of time, in milliseconds, to wait for services to expire before they are killed while a JCluster is being shut down.
- **Service Configuration Options:**

  This property provides two options:
  - Use all Service Master entries: If this option is selected, all services configured in the Service Masters node are available for Environment Managers using this configuration. In this case, the latest version of the JService is used.
  - Use configured service entries for this environment: If this option is selected, only the services specifically configured for this configuration are available.

- **Environment Behavior for Service Configuration Changes:**

  Determines how the Environment Manager behaves when there have been changes to the Service Master or Service Detail records in the Resource Database.
  - Flush and kill all services: Instructs the Environment Manager to refresh all configuration data and identify all instances of affected services. Those instances that do not have clients currently connected are killed, those that do have clients are killed upon client disconnect.
  - Flush only: Instructs the Environment Manager to refresh all configuration data but do nothing to currently running services.
  - Flush and restart applicable JClusters: Instructs the Environment Manager to refresh all configuration data and shut down and restart any JClusters that are configured to use the service. When the JCluster is stopped, it either waits until no clients are connected or waits for its configured roll over timeout.
  - Do nothing: Instructs the Environment Manager to do nothing after changes have been made. If this option is selected, the Environment Manager must be restarted before any changes take effect.

‘Fast’ Versus ‘Accurate’ Load Balancing

The “Load balancing options” parameter, described above, provides for tuning of load balancing by providing for two different options. This parameter allows for load balancing to be either “fast” or “accurate.” These two settings determine the behavior of the Environment Manager upon receiving a client request for a service.

- **Fast:**
  - Client contacts the Environment Manager and requests a service.
  - Environment Manager uses mutlicasting to broadcast an Environment Request For Service (ERFS) to determine which JCluster in the Environment Manager’s Load Balancing Group is best able to handle the request.
  - The Environment Manager waits either for the Load Balancing Timeout to expire, or for the first multicast response.
  - The first environment that responds to the ERFS is compared to the current environment to decide which one should be used to execute the client request.
- **Accurate:**
  - Client contacts the Environment Manager and requests a service.
  - Environment Manager uses mutlicasting to broadcast an ERFS (Environment Request For Service) to determine which JCluster in the Environment Manager’s Load Balancing Group is best able to handle the request.
  - Environment Manager waits for the Load Balancing Timeout to expire.

- After the Load Balancing Timeout has expired, the Environment Manager uses all multicast responses to determine which JCluster has the least load. That JCluster is used to execute the client request.

Load Balancing Cache Expiration

The Load Balancing Cache Expiration Time parameter, described in "Environment Manager Configurations" on page 66, is also used to tune load balancing behavior. This value specifies how long JCluster load information is cached before the JCluster updates the load information again. This is useful when Environment Request For Service requests come in rapidly. Rather than have the JCluster constantly responding to the Environment Managers' requests for load information, the JCluster caches this information.

Load information, for local JClusters running on the same machine as the Environment Manager requesting the load information, is not retrieved over the multicasting mechanism. So the Environment Manager must compare multicast load information with the load values for its local JClusters to determine the least-loaded JCluster.

This means that the two load balance configuration timeout values are impacted by the number of Environment Managers within the load balance group:

- For groups with only one Environment Manager, the Load Balancing Timeout should be set low (for example, 100 milliseconds). The Load Balancing Cache Expiration Time should be set high, so that network load balancing requests are minimized. In this configuration, the Load Balancing Options setting does not matter.
- For "accurate" load balancing, the Load Balancing Cache Expiration Time should be set low, (for example, 5000 milliseconds), so that balancing information is not more that 5 seconds old.

Using Service Details to Balance Load

Service details, which override default settings in "service masters", can be added to configurations. Service Masters are default configuration settings for services that are stored in the Resource Database. Service details apply to only one configuration, and override configuration settings for service masters. This allows individual settings for services to be configured on a per-configuration basis.

For example, in a configuration for multi-processor machines, the minimum number of service instances could be set high, so a large number of service instances are running automatically when the Environment Manager is started. In a configuration for single-processor machines, set this number lower, perhaps even at 0, so the load on the machine is relatively low. These different settings are made in service details.

For more information about service masters and service details, see Chapter 12 - "Configuration Manager" on page 455.

Limiting Environment Mangers to Use Only Service Details

Environment Manager configurations can be limited to use only Service Details that have been configured specifically for that configuration. This is done by setting the Service Configuration Options record in the Configuration Manager to “Use configured service entries for this environment”. This can be helpful to create services that receive priority from the highest performance machines. For example, there might be a number of services that clients can execute, but a limited set of services have the highest priority; account balance retrieval services for example.

A configuration can be created for high-performance machines that has service details for these important services. Using the Service Configuration Options record, this configuration could be limited to only using its service details. Additional configurations for less powerful machines can be set to use all service master records. This way, when a client requests one of the important services, they are quickly served by your fastest machines. Other, less important services continue to be served, but only by the less powerful machines.

Proxy Server

Because most networks only provide for multicasting within the same sub-net, a Resource Server can be established as a “proxy” server to facilitate load balancing between Environment Managers in the same group that are running on different sub-nets. See “Proxy Server” on page 76 for details.

Monitoring

webMethods JI includes a monitoring function that allows the environment to be monitored on a number of levels. The System Monitor, a graphical user interface included with webMethods JI, allows the user to “drill down” from high level monitoring to successively lower levels. This allows the entire webMethods JI environment to be monitored. Examples of high level and lower level monitoring include:

- **High level:** Monitoring JClusters and their concurrent service count.
- **Mid level:** A particular JCluster is selected and all services running in that JCluster are monitored.
- **Lower level:** Client information for the JCluster’s list of services is monitored.
- **Low level:** A particular service is selected and tracing and/or messages for the service is monitored.

For more information about using the System Monitor, Chapter 13 - "System Monitor" on page 515.

Monitored Information

The System Monitor monitors the following information:

Environment Manager Events and Activities

The System Monitor provides a monitoring interface to the following items for each Environment Manager running in the webMethods JI environment:

- Number of running JClusters
- Environment Manager start/stop events
- Client connection/disconnection events
- Client Request For Service (RFS) events
- Result of RFS requests
- Error conditions

The System Monitor is also used as an interface to Environment Managers for starting and stopping JCluster processes.

JCluster Events and Activities

The System Monitor also provides an interface to JClusters to monitor the following information:

- Number of running services
- Types of services and service version information
- Service start/stop events
- Client Request For Service (RFS) events
- JCluster status
- JCluster start/stop events
- Error conditions
- Current load (the number of clients communicating to services through this JCluster)

JService

The System Monitor provides the following information about JServices:

- Current state of the JService (idle, connected, etc.)
- JService starting, stopping, and initializing
- Protocol Agent events
- Client connection/disconnection events
- Method invoke events
- Error conditions

Licensing

The webMethods JI license key contains one to five licenses, each containing a combination of webMethods JI features required by a specific customer. The available features include:

- JI_SERVER determines the maximum number of servers on a network, permitted to run a webMethods JI Environment Manager.
- JI_SERVICE determines the maximum number of concurrent Services.
- JI_CLIENT determines the maximum number of concurrent run-time clients that are external to the webMethods JI Server and are connected to active services.

- JI_HOST determines the maximum number of concurrent host sessions.
- JI_MAPMAKER allows an unlimited number of MapMaker GUIs to run simultaneously.

For a detailed description of the licensing mechanism, see "Licensing" on page 504.

Editing the License File

The License file is edited from within the Configuration Manager user interface. See "The License File (license.txt)" on page 506.

Server Concurrency Restrictions

As a result of server concurrency restrictions, the following issues should be considered:

- If only one concurrent server is allowed, it is necessary to shut down the webMethods JI server prior to upgrading or running the software on another machine. No Environment Manager redundancy or multiple-server load balancing is allowed in this configuration.
- If two or more concurrent servers are allowed, redundancy can be accomplished through the use of two or more redundant Environment Managers. Further, if one machine goes down, the webMethods JI environment can be brought up on another machine without any interruption in service. Similarly, load balancing between multiple server machines is available.
- Licenses are available to set the number of concurrent environments as high as necessary, including "unlimited."

Note: Because of webMethods JI licensing restrictions, you can run multiple Environment Managers under a single IP address. However, the multiple Environment Managers cannot be run on a single host using different IP addresses (for example, if the host has more than one network card). This is considered by the environment to be two separate servers even though they may be on the same host machine.

JService Concurrency Restrictions

The JService concurrency restriction limits the use and scalability of the individual webMethods JI service components. This value restricts the concurrent number of JServices permitted across all JClusters running in the user's environment. Licenses are available to permit a value of "unlimited."

Expiration Functionality

Also available is an expiration date functionality that allows for control over evaluation software. If an expiration date is specified for a feature, certain JI server-side components log the amount of time remaining before the feature expires.

License Exceeded Behavior

When a licensed value is exceeded, the following items are controlled:

- When the concurrent number of servers is exceeded, Environment Managers on additional servers do not start until the concurrent server count goes below the licensed value. “License exceeded” messages are sent to the Environment Manager’s logfile.
- When the concurrent number of JServices is exceeded, additional JServices do not start until the concurrent count goes below the licensed value. “License exceeded” messages are sent to the Environment Manager’s logfile.
- When expiration timeout is exceeded, Environment Managers no longer start. Any running Environment Managers continue to function.

Redundant Licensing Behavior

In order for an Environment Manager to launch, a Resource server must be serving the License Name indicated by the license parameter in the Environment Manager’s configuration file. In the event that the Resource Server goes down, duplicate license entries can be entered into the license files served by different Resource Servers.

Proxy Server

Because most network configurations do not allow multicasting across sub-nets, “Proxy Server” functionality is included with webMethods JI to allow load balancing and resource location among multiple servers running on separate sub-nets. All webMethods JI components that use multicasting use an existing Proxy Server, to provide seamless integration of environments over multiple subnets. This includes Environment Managers, Resource Servers, databases, licensing, the Configuration Manager, System Monitor and Mapmaker.

Proxy Servers are actually normal instances of Resource Servers that are configured to function as Proxy Servers in addition to their normal Resource Server behavior.

There are two types of Proxy Servers in the webMethods JI environment:

- **The "hub" Proxy Server:** The hub Proxy Server can be considered the "primary" server because it is the server that all satellite Proxy Servers contact in order to communicate across sub-nets. Also, one or more alternate "hub" servers can be established for redundancy. There must be only one active hub Proxy Server (and one or more alternate hubs) in your network.
- **"Satellite" Proxy Servers:** These servers are instances of Proxy Servers that communicate with the hub Proxy Server and must be running on sub-nets other than the hub's sub-net. Satellite Proxy Servers serve as their sub-net's conduit for communication across sub-nets. There must be one and only one satellite server for each sub-net.

Figure 4 illustrates one possible configuration of Proxy Servers.

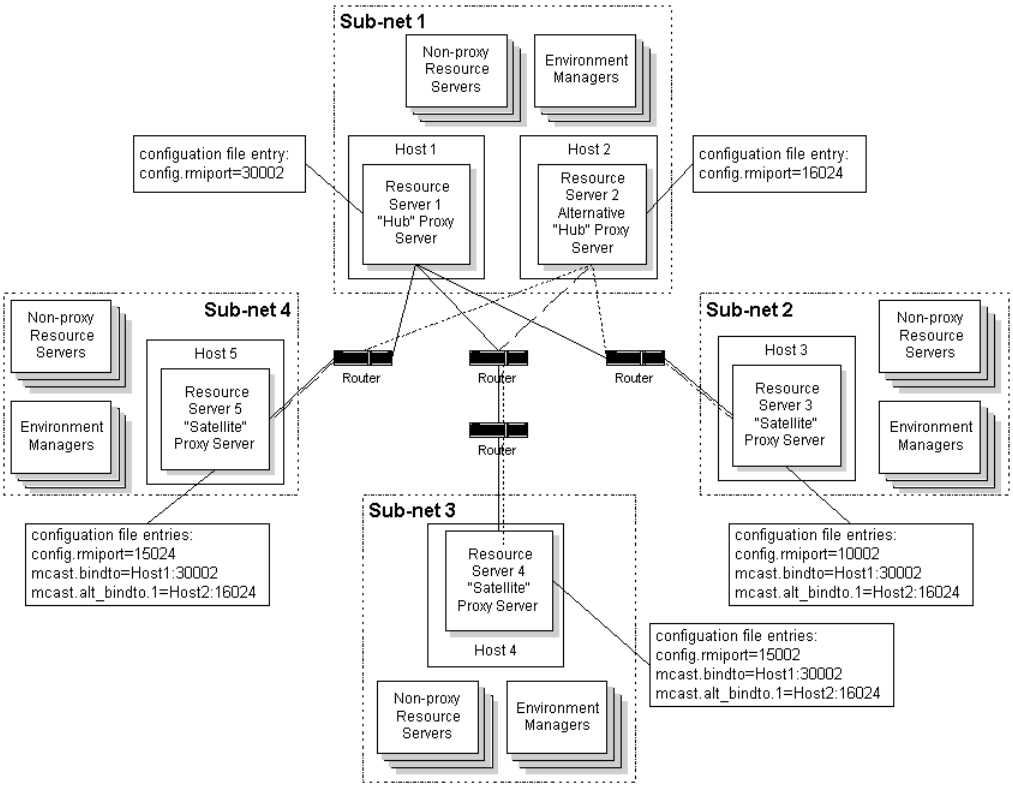


Figure 4. Possible Proxy Server Configurations

In this configuration, the primary hub Proxy Server is Resource Server 1, running on Host1. There is one alternate Hub server, running on Host2. There are three Satellite servers, one in each sub-net.

This example uses a "hub and spoke" network topology. Alternative topologies, such as a ring or serial configuration, can be used, but these are less efficient than configurations that have a single hub server and multiple satellite servers.

Configuring the Hub and Alternate Hub Proxy Servers

No specific configuration is required for a Resource Server to serve as a hub Proxy Server or an alternate hub. All required configuration takes place at the satellite server(s), who set their mcast.bindto and/or mcast.alt_bindto to the host and port of the hub server. Then, when a Resource Server receives a multicast packet from a satellite Proxy Server, it opens a multicast channel (if the channel is not already open) and begin multicasting information over that channel.

For improved performance by the Proxy Server, a Resource Server can be configured to load the required Proxy Server classes before the first satellite server sends a multicast packet to the hub server. This is accomplished by setting the following parameters in the Resource Server configuration file:

```
service.alias.mcast=mcast,com.jacada.ea.resserver.service.MCastProxyService
service.start.mcast=mcast
```

Both of these parameters are included by default in the configuration file for all Resource Servers, but the service.start.mcast=mcast parameter is commented out (using #). To instruct this instance of the Resource Server to automatically load the Proxy Server classes on startup, simply uncomment the service.start.mcast=mcast parameter.

Because only one hub or satellite Proxy Server can be running on a single sub-net, only one Resource Server on each sub-net should have the service.start.mcast=mcast parameter uncommented. If the Proxy Server classes are already running for one Resource Server on the sub-net, no other Resource Server can load the classes or act as a multicast proxy hub or satellite.

Configuring Satellite Proxy Servers

The primary parameter that needs to be configured for satellite Proxy Servers is the mcast.bindto parameter. This parameter must be set to the host and RMI port of the Resource Server with which the satellite server communicates. For example:

```
mcast.bindto=host1.acme.com:30002
```

Alternate Proxy Server hubs are established using mcast.alt_bindto. The value for this parameter is a comma-separated list of host:port combinations, for example:

```
mcast.alt_bindto=host2.acme.com:31002, host3.acme.com:32002
```

In this example, the satellite server first attempts to connect to its primary hub Proxy Server, the Resource Server running on host1.acme.com:30002. If that server does not respond, it attempts to connect to host2.acme.com:31002.

If that server does not respond either, it attempts to connect to host3.acme.com:32002. Any number of satellite Proxy Servers can be configured in this manner.

> **Note:** The `mcast.bindto` parameter must be set for all satellite Proxy Servers. However, this parameter (and the `mcast.alt_bindto` parameter) cannot be set for the hub Proxy Server. If either of these parameters are set in the configuration file for the hub Proxy Server, that Resource Server does not function as a hub.

Satellite Proxy Servers must also include the following parameters in the Resource Server configuration file:

```
service.alias.mcast=mcast,com.jacada.ea.resserver.service.MCastProxyService
service.start.mcast=mcast
```

Both of these parameters are included by default in the configuration file for all Resource Servers, but the service.start.mcast=mcast parameter is commented out (using #). For any instance of the Resource Server that runs as a satellite Proxy Server, the service.start.mcast=mcast parameter must be uncommented.

Because only one hub or satellite Proxy Server can be running on a single sub-net, only one Resource Server on each sub-net should have the service.start.mcast=mcast parameter uncommented. If the Proxy Server classes are already running for one Resource Server on the sub-net, no other Resource Server can load the classes or act as a multicast proxy satellite.

Chapter 3. MapMaker

MapMaker is a Java-based application included with webMethods JI that allows users to automatically create interfaces to legacy applications. These interfaces are then output from MapMaker as webMethods JI Java services for use in the webMethods JI server environment.

MapMaker is designed to simplify the mapping of legacy application screens into Java application logic. This is accomplished by creating a "map" of the host application, which consists of the application screens, the navigational information required to traverse through the sequence of screens, and the tags and fields included on each legacy screen. This information is stored in a map file and in Java class files that are produced from within MapMaker.

MapMaker supports English and Western European single-byte and Asian double-byte data streams. However, only 7-bit US ASCII is supported for "names" of items (screen, tag, and field names, log file names, map and service names, etc.). In addition, only alpha characters (A-Z, upper or lower case), should be used as the first character of most "names".

This chapter describes:

- "Starting MapMaker" on page 81
- "The MapMaker Interface" on page 82
- "Configuring MapMaker Properties" on page 96
- "Graph Window" on page 108
- "Printing the Display" on page 109
- "MapPlayer" on page 120

Starting MapMaker

MapMaker can be started from the command line. Additionally, in Windows, MapMaker can be started using shortcuts, if they were created during webMethods JI installation. For example, if shortcuts were added to the **Start** menu during installation, MapMaker can be started by selecting that shortcut from the appropriate program menu in the **Start** menu.

Command Line

Starting MapMaker from the command line can be accomplished by executing the following command at the command line:

```
cd <JI_install_dir>\bin
.\ea_mapmaker
```

This executable uses a “lax” file to specify the runtime behavior of the application. If certain changes occur to the system, it may be necessary to edit the `ea_mapmaker.lax` file. An example of this could be using a different Java Runtime Environment (JRE) than the one selected during the webMethods JI installation. For information about editing lax files, see Chapter 7 - "LAX Files" on page 177 of the *Supplemental Reference Guide*. For information about command line options, see Chapter 1 - "webMethods JI Commands" on page 11 of the *Supplemental Reference Guide*.

The MapMaker Interface

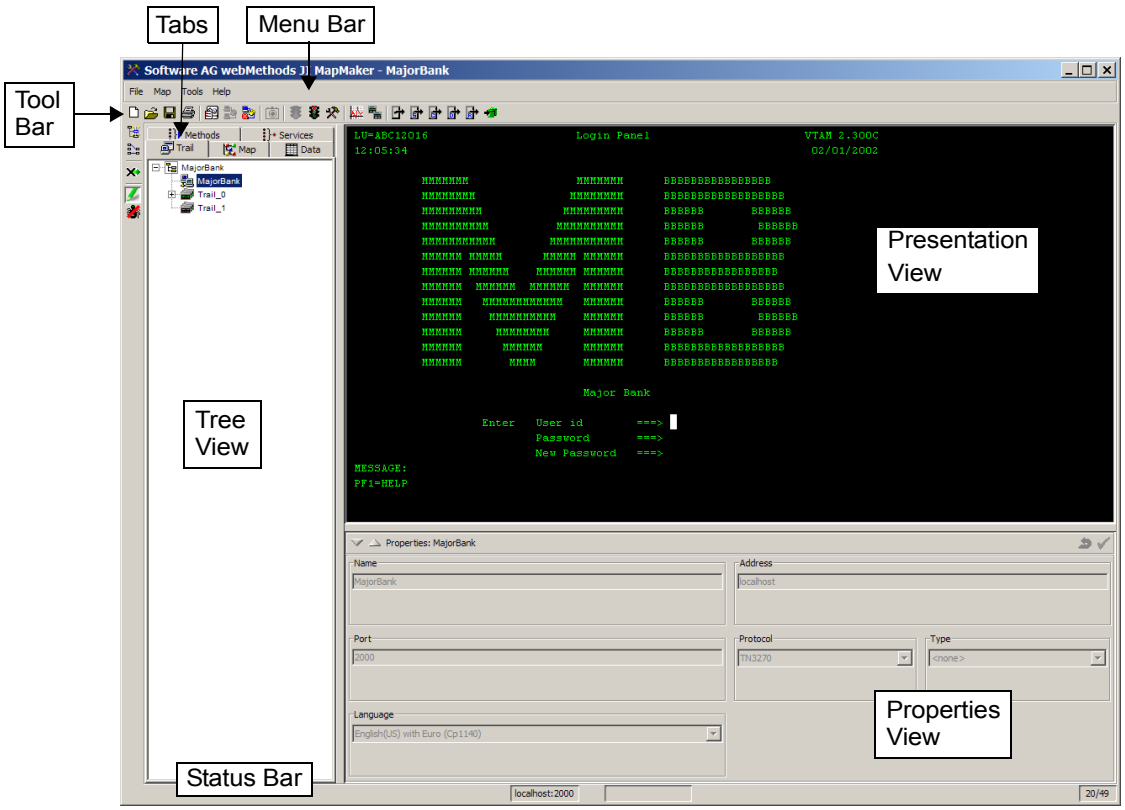


Figure 5. The MapMaker Interface

Note: The "look and feel" of the MapMaker graphical interface can be customized by the user. Available options are Metal, CDE/Motif and Windows (for information about changing these options, see "Appearance Tab" on page 99). The screen snapshots in this manual use the Windows look and feel.

MapMaker Views

The MapMaker interface consists of the following primary views:

- "Tree View" on page 83
- "Presentation View" on page 84
- "Properties View" on page 85

Tree View

The left pane of the MapMaker interface is the Tree view, providing a hierarchical view of the information. The root of the Tree shows the name of the map that is currently displayed (e.g. "**MajorBank**" in Figure 6). The root can be expanded to expose underlying sub-components, known as nodes or branches. Typically, these branches can be further expanded to reveal their own sub-branches, thereby creating a hierarchical tree structure. A "**+**" or "**-**" symbol before a branch allows you to expand or collapse it (respectively).

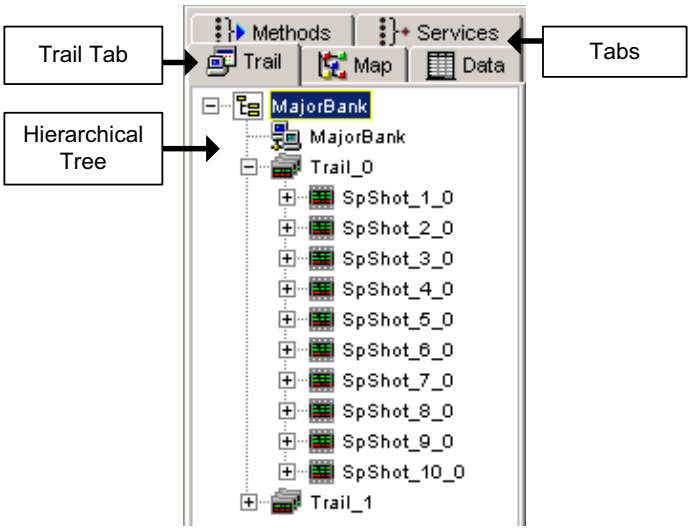


Figure 6. Tree View

Tabs

The Tree view consists of the following tabs:

- **Trail** tab: Graphically displays the trails that are recorded during navigation of the legacy application. For more information about trails, see "Trail Recording" on page 137.
- **Map** tab: Graphically displays the map of the legacy application, generated based on the trails and other information. For more information about Maps, see "Map Properties" on page 147.
- **Data** tab: Used to add data templates and table templates to the screens included in the map. For more information about data templates and table templates, see "Data Templates" on page 183 and "Table Templates" on page 190.
- **Methods** tab: Used to define Methods that, in turn, define transactions against the host application. For more information about Methods, see "The Methods Tab" on page 275.
- **Services** tab: Used to define the services to be generated by MapMaker. These are the runtime components within the server environment. For more information about Services, see "The Services Tab" on page 391.

The tab displayed in the Tree view determines the contents of the adjacent, right pane, which serves as the Presentation view. When the **Services** tab is displayed in the Tree view, the right pane of the MapMaker interface remains empty.

Presentation View

When the **Trail**, **Map**, or **Data** tabs are displayed in the Tree view, the right pane of the MapMaker interface serves as a Presentation view. This view allows you to process host screens in different ways, depending on the tab that is selected in the Tree view:

Tree Tab	Presentation View Functionality
Trail	Displaying an emulation of the current host session or previously recorded host screens.
Map	Defining tags in the recorded screens.
Data	Defining data templates and table templates in the recorded screens.
Methods	Illustrating the flow of the method that is selected in the Tree view.

Tree Tab	Presentation View Functionality
Services	Illustrating the map's JI Services and the Methods attached to each Service in the Tree view. The Presentation view remains empty.

Note: The Presentation view correctly interprets and handles all field attributes, including attributes (such as blinking) that are not displayed.

Properties View

The bottom right pane of the MapMaker interface is the **Properties** view. This view allows you to view and edit the properties of the object selected in the Tree view or Presentation view. Figure 7 shows an example Properties view, displaying the settings of the MajorBank host that is selected in the Tree view.

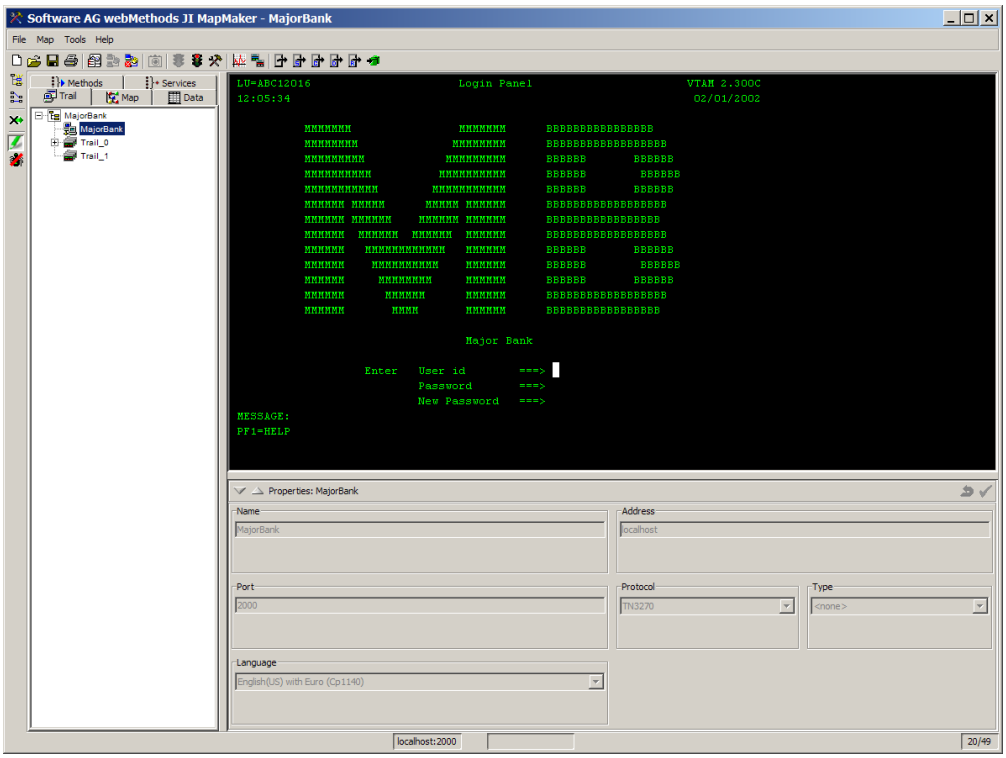


Figure 7. Properties View

The selected object's name is indicated in the Properties view's caption (e.g. **Properties: MajorBank**).

Collapsing and Expanding the Properties View

The Properties view can be minimized and expanded using the arrow buttons located at its upper left corner. Once these buttons are clicked, a corresponding floating arrow indicates the panel's previous position, allowing you to conveniently restore it:

- Minimize , followed by a floating expand button .
- Expand , followed by a floating contract button .

Cancelling and Applying Property Updates

When there are unsaved changes to the object's properties, the **Revert** and **Apply** buttons, located in the upper right corner of the Properties view, are enabled.

- Click **Revert** to cancel all recent updates made across all tabs, and restore the properties that were last saved.
- Click **Apply** to enforce all the updates made across all tabs since the last save.

Revert and **Apply** work across all tabs. The changes for a given component accumulate across all property tabs, until the **Revert** or **Apply** button is pressed.

> **Note:** Selecting another component performs an auto-apply.

MapMaker Menus

The MapMaker menu bar contains the following menus:

- "File Menu" on page 87
- "Map Menu" on page 89
- "Tools Menu" on page 90
- "Help Menu" on page 93

The following sections describe the options available in each menu.

File Menu

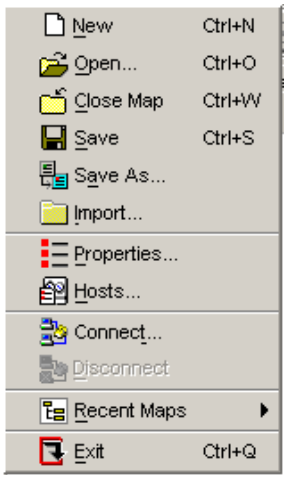


Figure 8. The File Menu

To access the **File** menu using the mouse, click **File** on the menu bar and then click the command of interest. To use the keyboard, press **Alt** + **F** simultaneously to open the menu and then press the key that corresponds to the underlined letter of the command.

The **File** menu offers the following options:

Option	Command
New (Alt + F, N or Ctrl + N)	Creates a new map file.
Open (Alt + F, O or Ctrl + O)	Accesses the **Open** dialog box that is used to open an existing map file.
Close Map (Alt + F, W or Ctrl + W)	Closes the current map.
Save (Alt + F, S or Ctrl + S)	Saves the current map file.

Option	Command
Save As (Alt + F, A)	Opens the **Save As** dialog box, where the file location for the map file can be selected and the map file can be given a new name. **Note:** A check box in the **Save As** dialog box allows the user to designate the map as a Shared Save. When so selected, the map is then available for shared development by multiple users.
Import (Alt + F, I)	Opens the **Select File For Import** dialog box, where a map file may be selected from which to import trails.
Properties (Alt + F, P)	Opens the **Properties** dialog box, where various options for the MapMaker application can be configured, such as the default locations for map files, the location of the Java Development Kit and other information.
Hosts (Alt + F, H)	Opens the **Hosts** dialog box, where legacy host information is entered, such as the name and/or IP address and port number of the host, as well as the communications protocol and other information.
Connect (Alt + F, C)	Opens the **Connect to Host** dialog box. This dialog box is used to select and connect to the appropriate host.
Disconnect (Alt + F, D)	Disconnects from the current session with the host.
Recent Maps (Alt + F, R)	Provides a sub-menu that lists the most recently used map files. To open one of the map files listed in the sub-menu, select the file from the list.
Exit (Alt + F, E or Ctrl + Q)	Closes MapMaker. If the current map file has not been saved, a prompt appears to save the file before MapMaker exits.

Map Menu

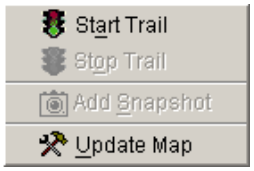


Figure 9. The Map Menu

To access the **Map** menu using the mouse, click **Map** on the menu bar. To use the keyboard, press **Alt** + **M** simultaneously to open the menu and then press the key that corresponds to the underlined letter of the command. The **Map** menu offers the following options:

Option	Command
Start Trail (Alt + M, A)	Starts recording a trail consisting of the screen snapshots and navigational steps used during the legacy application interaction.
Stop Trail (Alt + M, O)	Stops recording the trail.
Add Snapshot (Alt + M, S)	Takes a "snapshot" of the legacy application's current screen and adds the snapshot to the Trail Repository. **Note:** This option is only available in Character Mode.
Update Map (Alt + M, U)	Creates or updates the map. It is recommended that maps be updated frequently. In the event of an error, frequent updating helps track the problem. The following are times it would be a good idea to update: • After adding a tag • After disabling a field • After adding a new trail

Tools Menu

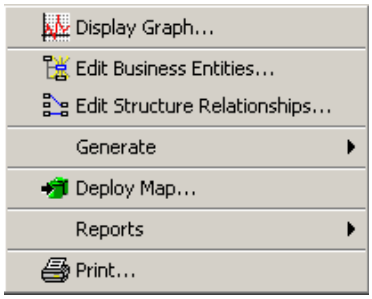


Figure 10. The Tools Menu

To access the **Tools** menu using the mouse, click **Tools** on the menu bar. To use the keyboard, press **Alt** + **T** simultaneously to open the menu and then press the key that corresponds to the underlined letter of the command.

The **Tools** menu offers the following options:

Option	Command
Display Graph (Alt + T, G)	Opens the **Map Schema** window, providing a graphical representation of the map file.
Edit Business Entities (Alt + I)	Opens the **Business Entity Editor**, used to define and edit all Data Types and Global Variables. For more information on Business Entities, see "Data Typing" on page 203.
Edit Structure Relationships (Alt + S)	Opens the **Structure Relationship Editor**, used to define and edit the relationships between the various Data Types and their components. For more information on Structure Relationships, see "Defining Data Structure Relationships" on page 249

Option	Command
Generate (Alt + T, E)	Opens a sub-menu used to generate java source code for the webMethods JI service associated with this map file. The sub-menu provides the following options: • **Generate All Code:** (Alt "+" T, E, A) Generates the service code and test clients. • **Generate Service Code:** (Alt "+" T, E, S) Generates the service code only. • **Generate Client Code:** (Alt "+" T, E, C) Generates test client code. • **Generate Definitions:** (Alt "+" T, E, W) Generates the definitions or description languages selected in the **Definitions** tab of the **Generate Code** dialog box. These options generate and compile the Java code and package the compiled code into JAR files. For more information about generating code, see "Generating the Service and Test Client" on page 398.
Deploy Map (Alt + T, D)	Opens a dialog box that allows the saved generated Jar file to be deployed. Once deployed, this file is available to the webMethods JI environment.
Reports (Alt + T, R)	Opens a sub-menu used to generate reports. The sub-menu provides the following option: • **Service Interface:** (**Alt** + **T**, **R**, **R**) Opens the **Report** dialog box for generating a service interface report. This provides details about the methods, method inputs, templates and data files associated with the service. **Note:** Output variables can also be identified. For more information about the **Reports** dialog box, see "Generating Reports" on page 111.

Option	Command
Compare	Opens the Screen Compare feature enabling you to compare one screen image with another, or to compare one screen image with all screens currently defined in a map. For more information, see "Comparing Screen Images" on page 171.

Print (Alt + T, P)	Opens the **Print** dialog box, used to specify the following print options:
	• Define the **Print Detail**:
	  • Set the **Selection** to one of the following:
	  - **Print Selected**: Prints a specific section of interest.
	  - **Print All**: Prints the entire display, consisting of the Tree View, the Presentation View and the Properties view.
	  • Set the **Type** to one of the following:
	  - **Scale**: Scales the map down in size, so it can be printed on a single page.
	  - Span: Maintains an accurate proportion of the image, extending the output across as many pages as needed.
	• Define the **Selection Color**: click **Change** to display the **Choose Selection Color** dialog box, and adjust the color settings as needed.

Help Menu

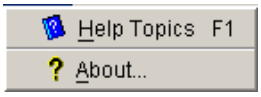


Figure 11. The Help Menu

To access the **Help** menu using the mouse, click **Help** on the menu bar. To use the keyboard, press **Alt** + **H** simultaneously to open the menu and then press the key that corresponds to the underlined letter of the command.

The **Help** menu offers the following options:

Option	Command
Help Topics (Alt + H, H or F1)	Opens a window containing help for MapMaker. **Note:** This launches Acrobat Reader with the webMethods JI User Guide. Acrobat Reader must be installed.
About (Alt + H, A)	Opens a window listing the version of MapMaker and other information.

Shortcut Menus

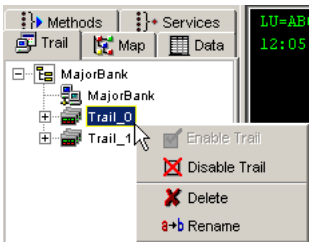


Figure 12. Example Shortcut Menu

Shortcut menus are available in many places throughout the MapMaker user interface. To open a shortcut menu, select and right-click a component in the Tree view, or right-click in the Presentation view.

Shortcut menus are context-sensitive. They present only those options that apply to the current situation. As a result, shortcut menus are not always available, and the options they present vary from situation to situation.

MapMaker Toolbar

The MapMaker tool bar contains shortcuts to a number of operations included in the pull-down menus. The following tool bar buttons are available:

New Map: Creates a new map file.

Open Map: Opens the **Open Map File** dialog box, used to open an existing map.

Save Map: Saves the current map.

Print: Opens the **Print** dialog box, used to specify various print settings.

Host Properties: Displays the **Hosts** dialog box, used to identify host connections.

Connect Host: Opens the **Connect to Host** dialog box, used to connect MapMaker to a legacy host.

Disconnect Host: Disconnects from the current host.

Add Snapshot: Takes a “snapshot” of the legacy application’s current screen and adds the snapshot to the Trail Repository.

Note: This button is only enabled in Character Mode.

Start Trail: Starts trail recording mode, which records the navigational steps used during interaction with the legacy application.

Stop Trail: Stops trail recording mode.

Update Map: Updates the map associated with a recorded trail.

Display Graph: Opens the **Map Schema** dialog box, which provides a graphical representation of the map file.

Edit Business Entities: Opens the **Business Entity Editor**, used to manage all External Business Entities, Internal Business Entities and Global Variables.

Edit Structure Relationships: Opens the **Structure Relationship Editor**, used to manage the relationships (one way or reversible) between the various Business Entities and their components.

XPath Evaluator: Opens the **XPath Evaluator**, used to verify the result of an XPath expression. For more information, see "Verifying the Result of an XPath Expression" on page 257.

Design Environment: Used from the **Debugger** interface to return to the MapMaker design environment.

Debugger: Opens the **Debugger**, used to debug methods. For more information, see "Debugging Methods" on page 363.

Generate All Code: Generates all code - service, client and WSDL.

Generate Service Code: Generates the service code only.

Generate Client Code: Generates the client code only.

Generate Definitions: Generates the definitions only.

Deploy Map: Deploys the current JI Service.

MapMaker Status Bar

The Status bar is located at the bottom of the MapMaker window. It provides additional information on the display, which varies as a function of the selected tab. For example:

- When working in the **Trail** tab, the Status bar provides host-related information (e.g. whether or not you are currently connected to a host, the host name etc.).
- When working in the **Methods** or **Services** tabs, the Status bar describes the first error detected regarding the object selected in the Tree view (or in the Presentation view, if you are in the **Methods** tab). For example, the Status bar may indicate that a particular step is not connected to the method, or that an object is not configured properly.

Naming Conventions

When naming items in MapMaker, Java naming conventions should be followed. This includes the following restrictions on names:

- No special characters. The following characters are not allowed: Left and right braces {} left and right brackets [ ], left and right parenthesis (), tilde (~), asterisk (*), plus sign (+), question mark (?), comma (,), hyphen (-), backslash (\) and forward slash (/).
- No spaces are allowed in any names.
- No numeric characters at the beginning of names.
- No reserved Java keywords, such as byte, double, package, etc. For more information about reserved Java keywords, see the Java documentation.
- No webMethods JI internally defined class names are allowed.

Note: All component names must be unique throughout the system, data and table templates, methods, services, and all other components in webMethods JI.

Configuring MapMaker Properties

Select **File** > **Properties** to open the MapMaker **Properties** dialog box. The **Properties** dialog box allows certain items to be configured within MapMaker to correspond with the system requirements. These include default locations for generated files, the location of the webMethods JI Resource Server, and other issues that are required for code generation.

The **Properties** dialog box consists of the following tabs:

- “General Tab” on page 97
- “Appearance Tab” on page 99
- “Default Directories Tab” on page 100
- “Java Tab” on page 102

- "Code Generation Tab" on page 104
- "Servers Tab" on page 105
- "Licensing Tab" on page 107

Note: Information in the **Properties** dialog box is carried forward from previous MapMaker installations. This information is stored in the mapmaker.cfg file, located in the c:\WinNT\Profiles\<username> directory in Windows. This file is binary and is not editable. This file can be overridden by command line option.

General Tab

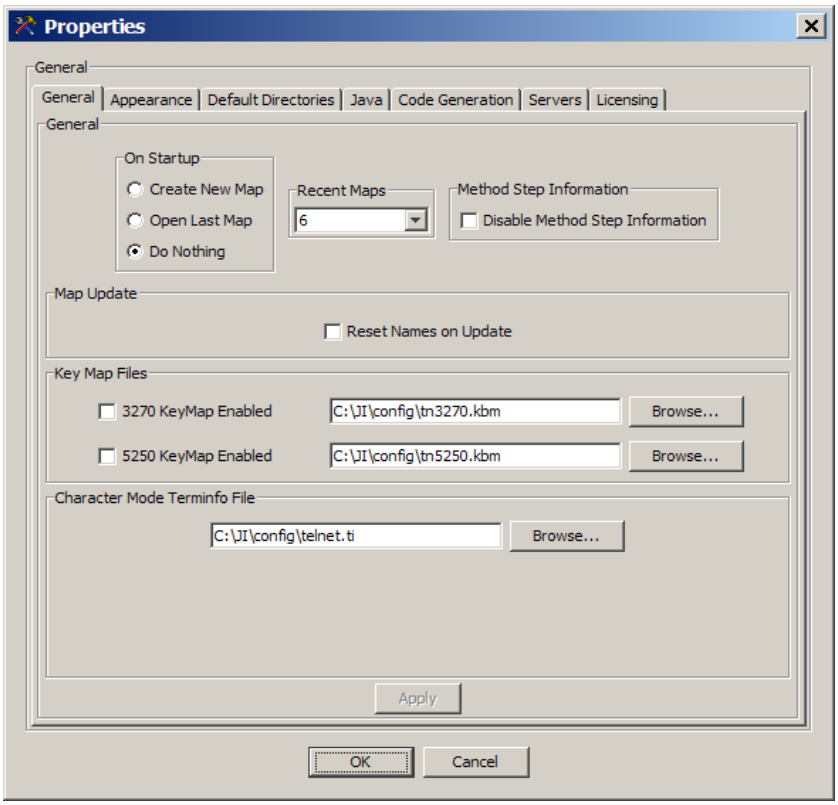


Figure 13. Properties Dialog Box > General Tab

The **General** tab provides the following options:

Option	Description
On Startup	Allows the user to configure how MapMaker opens map files when MapMaker is started. Options include: • **Create New Map:** Creates a new map on startup. • **Open Last Map:** Opens the most recently used map on startup. • **Do Nothing:** Does not open any map on startup.
Recent Maps	Determines the number of map files that are listed in the **Recent Maps** sub-menu.
Method Step Information	Choose whether to display or to **Disable Method Step Information**.
Reset Names on Update	When this option is selected and a map is updated, the names of all components are regenerated so that they are numbered in the correct order. **CAUTION:** Selecting this option resets any names given to individual screens or anything that has been renamed.
Key Map Files	MapMaker uses default keyboard mapping settings. This option allows the identification of customized keyboard mapping files. Sample keyboard mapping files are included with webMethods JI, named *tn3270.kbm* and *tn5250.kbm* and are located in *<JI_install_dir>/config*. For information about customizing these files, see Chapter 3 - "Keyboard Mapping" on page 99 of the *Supplemental Reference Guide*. To set custom keyboard mappings, set the checkbox to enable the custom settings and enter the path and filename for the keyboard mapping file. **Note:** The **Browse** button can be used locate the file.

Option	Description
Character Mode Terminfo File	MapMaker uses a default location for the term info translation file. To customize this location, enter a new location, or use the **Browse** button to find the file location. A custom terminal type can be configured by modifying the default configuration file. **Note:** The file is located in the <JI_install_dir>/config directory and is called telnet.ti.

Appearance Tab

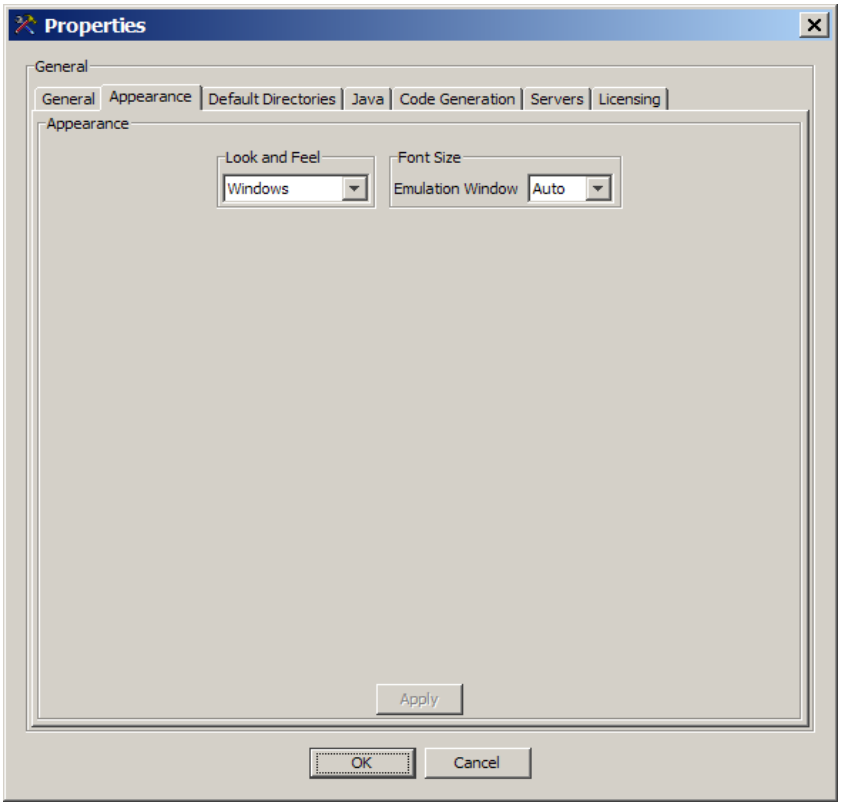


Figure 14. Properties Dialog Box > Appearance Tab

The **Appearance** tab provides the following options:

Option	Description
Look and Feel	Defines the style of the MapMaker graphical interface. Choose between **Metal**, **CDE/Motif** and **Windows**. **Note:** The screenshots in this manual use the **Windows** look and feel.
Font Size	The size of the font displayed in the **Presentation** view.

Default Directories Tab

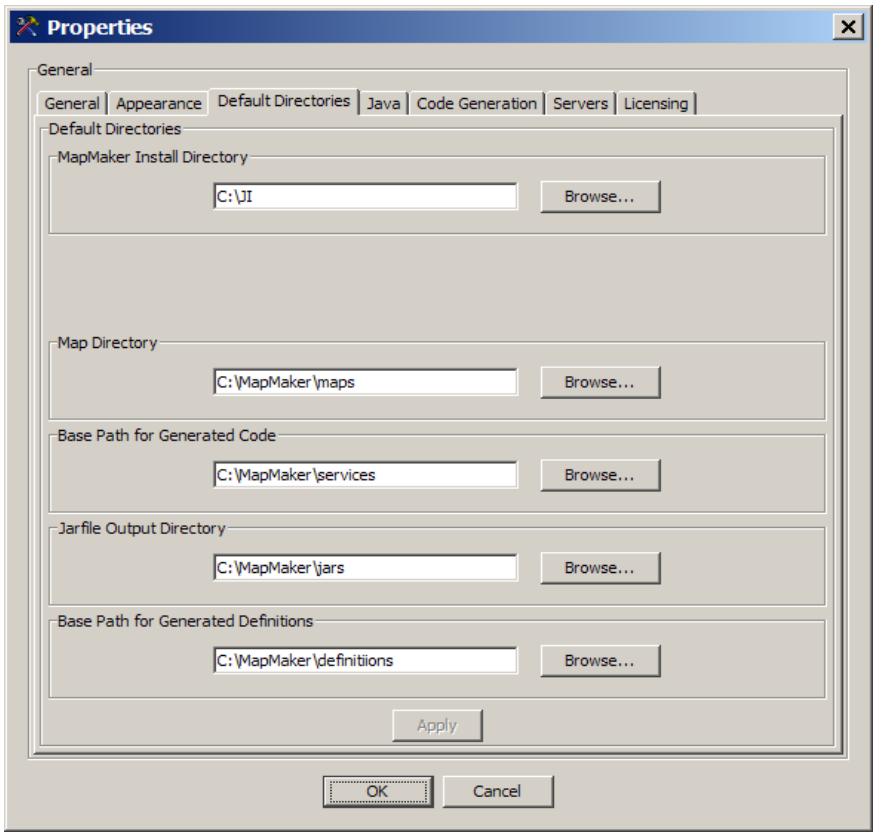


Figure 15. Properties Dialog Box > Default Directories Tab

The **Default Directories** tab provides the following options:

Option	Description
MapMaker Install Directory	The root directory into which MapMaker files were installed. Generally, this would be <*JI_install_dir*>, for example: *c:\JI* (Windows)
Map Directory	The default location for map files. This location is used as the default location when map files are saved. Enter the path (and drive letter, if applicable) to the location for the map files, or use the **Browse** button to select the appropriate path.
Base Path for Generated Code	The default location for service code that is output from MapMaker. This serves as the default location in the Generate Service Code dialog box. Enter the path (and drive letter, if applicable) to the location for the output files, or select the location by clicking the **Browse** button and selecting the appropriate path.
Jarfile Output Directory	The default location for the Jar files that are output from MapMaker. Enter the path (and drive letter, if applicable) to the location for the output Jar files, or use the **Browse** button to select the appropriate path.
Base Path for Generated Definitions	The default location for the definition files that are output from MapMaker. This serves as the default location in the **Generate Definitions** dialog box. Enter the path (and drive letter, if applicable) to the location for definition files, or select the location by clicking the **Browse** button and selecting the appropriate path.

Java Tab

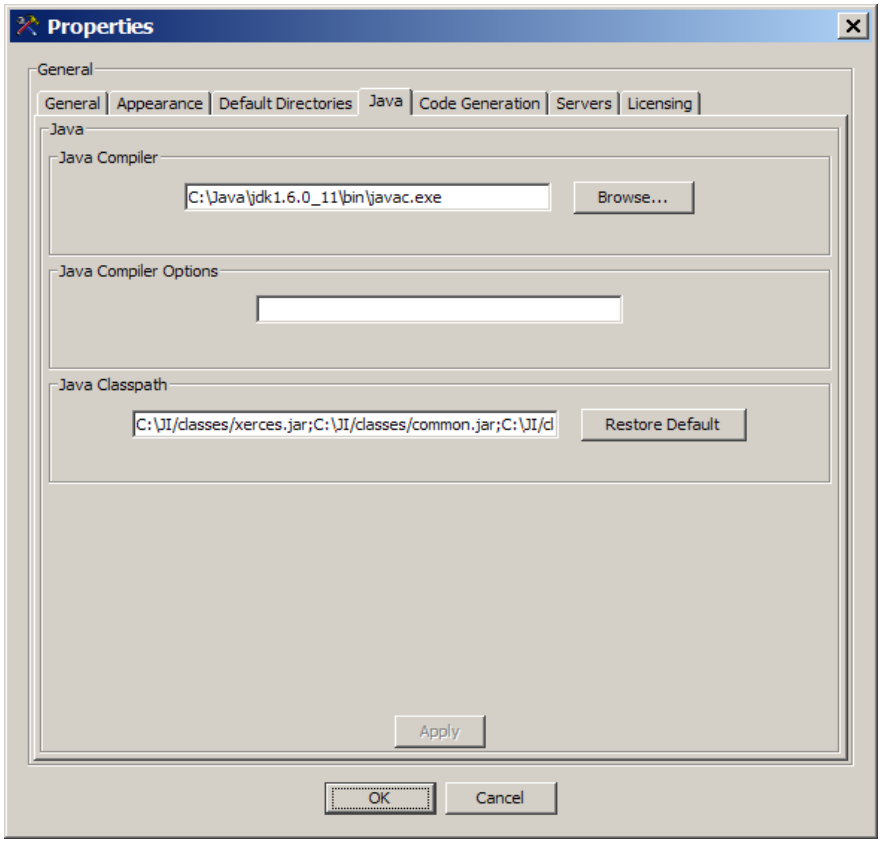


Figure 16. Properties Dialog Box > Java Tab

The **Java** tab provides the following options:

Option	Description
Java Compiler	The location of the system's Java compiler executable (*javac.exe* in Windows, or *javac* in UNIX). Enter the path (and drive letter, if applicable) to the location for the Java compiler, or use the **Browse** button to select the appropriate path. **Note:** The JDK or SDK must be installed to use the compiler.
Java Compiler Options	Enter command line options to pass to the compiler at compile time.

Option	Description
Java Classpath	The Java class path that is used by MapMaker to compile generated service code, custom classes, and/or test client code. The minimum required class path for this field should include the following: *<JI_install_dir>/classes/mapmaker.jar* where *<JI_install_dir>* represents the path (and drive letter, if appropriate) to the installation of webMethods JI.
Restore Default	Restores the default Java class path as designated by the webMethods JI installer.

Code Generation Tab

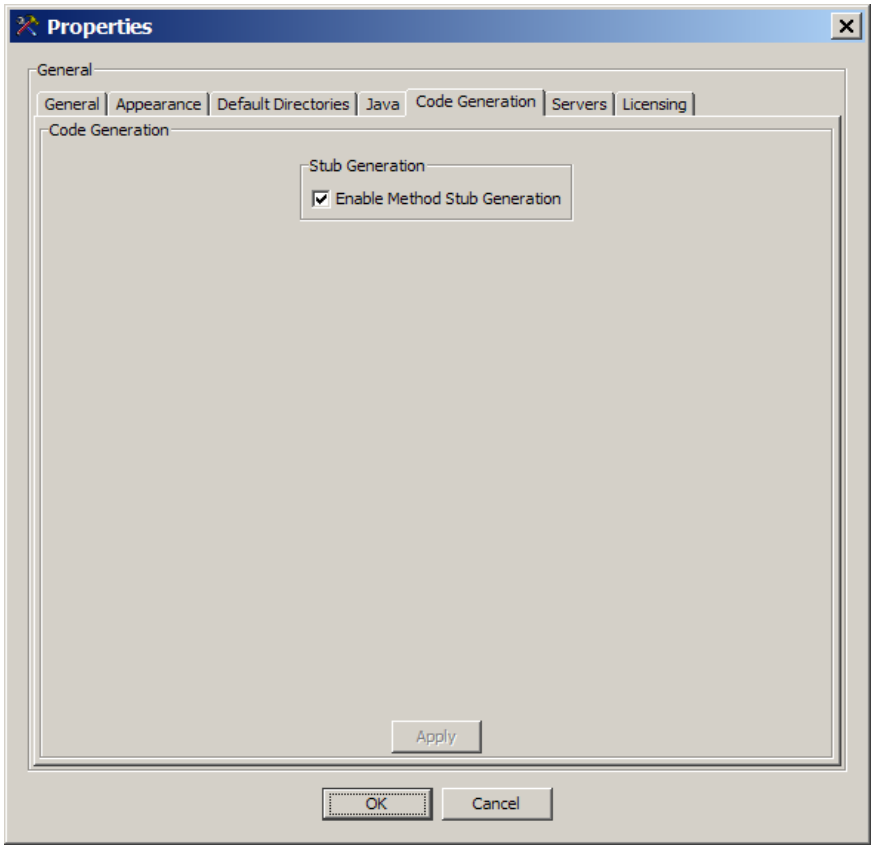


Figure 17. Properties Dialog Box > Code Generation Tab

The **Code Generation** tab provides the following options:

Option	Description
Enable Method Stub Generation	If this option is set and any Custom Classes have been included in the map, stub source files are created for the Custom Classes. Clear this option to prevent stub source files from being created.

Servers Tab

The **Servers** tab allows the determination of what mechanism MapMaker uses to locate the Resource Server. MapMaker can automatically deploy the generated service code into a Resource Database, but it must be able to find the Resource Server that serves the database. Identify the host and port of the Resource Server or instruct MapMaker to use multicasting technology to locate all Resource Servers running on servers within the subnet.

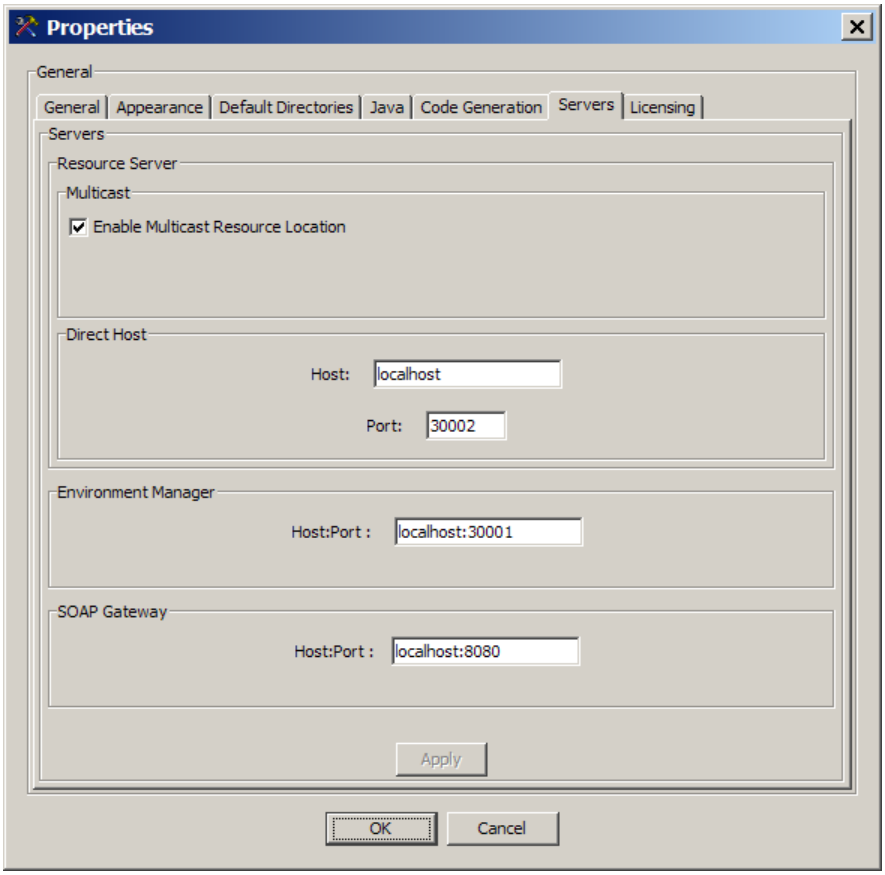


Figure 18. Properties Dialog Box > Servers Tab

The **Servers** tab provides the following options:

Option	Description
Multicast	Set the **Enable Multicast Resource Location** checkbox to instruct MapMaker to use the webMethods JI multicast resource location mechanism to locate the Resource Server. When you deploy a service with multicasting enabled, you are prompted to select the Resource Server from a list of all Resource Servers that were located on the local subnet. **Note:** If always deploying to the same Resource Server, clear this checkbox for faster deployment.
Direct Host	The name, or IP address and port number, of the Resource Server. When the user deploys a service without enabling multicasting, this host and port are used as the location of the Resource Servers. If the user deploys a service with multicasting enabled, the Resource Server listed here is listed along with any servers located via multicasting.
Environment Manager	The name, or IP address and port number, of the Environment Manager. This information serves as the default setting for the Environment Manager location that is included in generated client code.
SOAP Gateway	The Host name and Port of the servlet container in which the SOAP Gateway resides. Default is `localhost:8080`

Licensing Tab

The **Licensing** tab displays the MapMaker license and allows you to update it as needed.

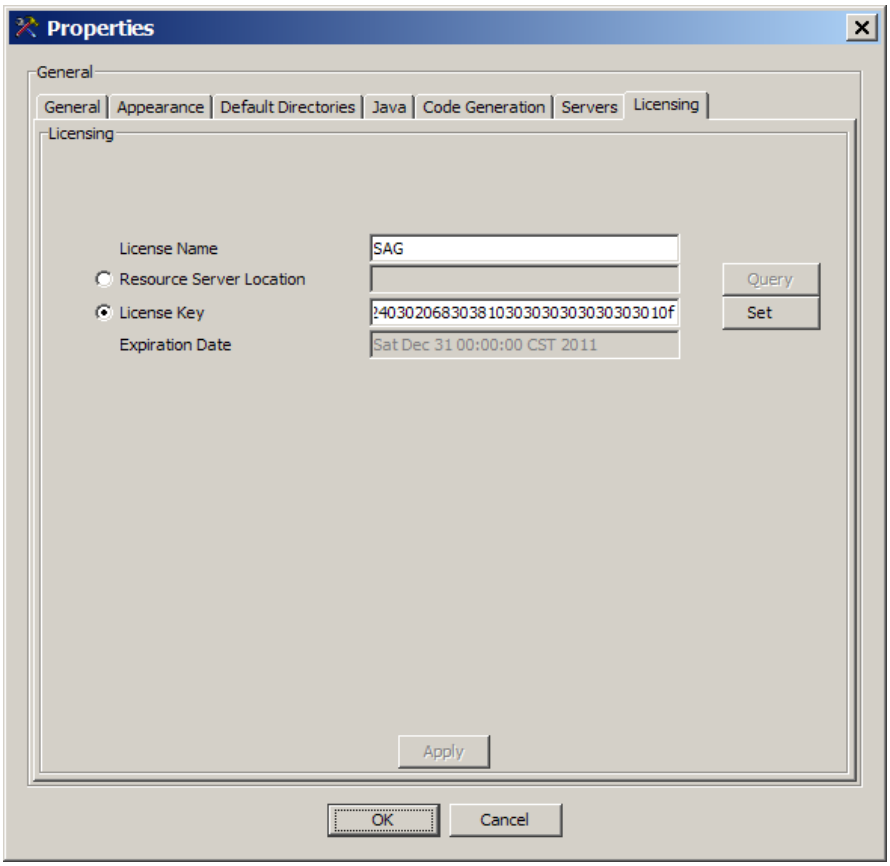


Figure 19. Properties Dialog Box > Licensing Tab

The **Licensing** tab provides the following options:

Option	Description
License Name	The name of the license. A license name is required whether the key is entered manually or obtained from the resource server.
Resource Server Location	This option is used to automatically locate and obtain a MapMaker license. Enter the host and port (e.g. `localhost:30002`) and click **Query**. This automatically completes the **License Key** and **Expiration Date** fields.

Option	Description
License Key	This option is used to manually enter a MapMaker license. Enter the license key and click **Set**. MapMaker checks the validity of the key and completes the **Expiration Date** field.
Expiration Date	The date and time on which the license expires. This date is automatically calculated based on either the **Resource Server Location** or the **License Key**.

Graph Window

After a map update has been completed, a graphical representation of the map can be viewed. This graphical representation shows the screens and actions involved in the map. To view the graphical representation, select **Tools** > **Display Graph**.

The Graph window, titled **Map Schema**, is displayed (Figure 20).

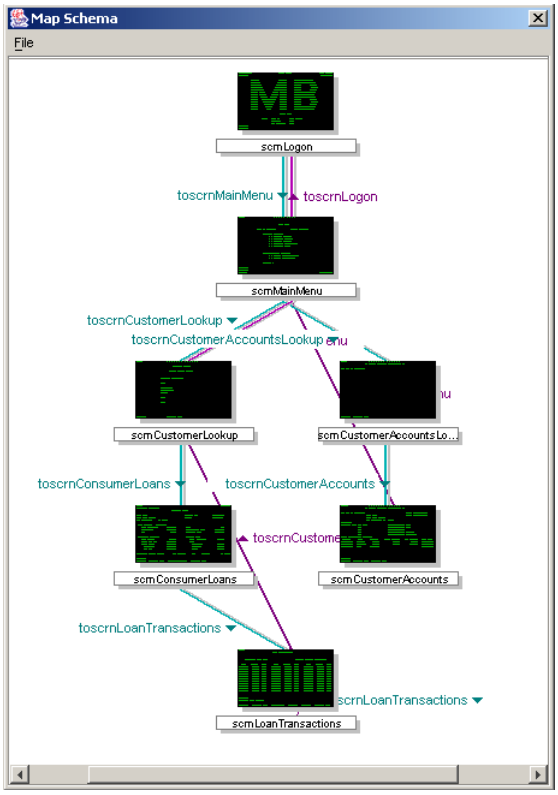


Figure 20. The Map Schema Window

Clicking on a screen image in the **Map Schema** window displays the screen in the Presentation view of the MapMaker user interface.

To print the graph select **File** > **Print** and choose one of the following print options:]

- **Span:** Prints the graph. Output may extend across multiple pages if necessary.
- **Scale:** Scales the map down in size to enable it to be printed on a single page.

Printing the Display

The **Print** dialog box can be launched from the following locations, by clicking the the **Print** tool bar button :

- The MapMaker display (in this case, the **Print** dialog box can also be launched by selecting **Tools** > **Print** from the menu)
- The **Business Entity Editor**
- The **Structure Relationship Editor**

The **Print** dialog box allows you to select the components of the display you wish to print, as well as the desired print type and color. Figure 21 shows an example **Print** dialog box, launched from the MapMaker display.

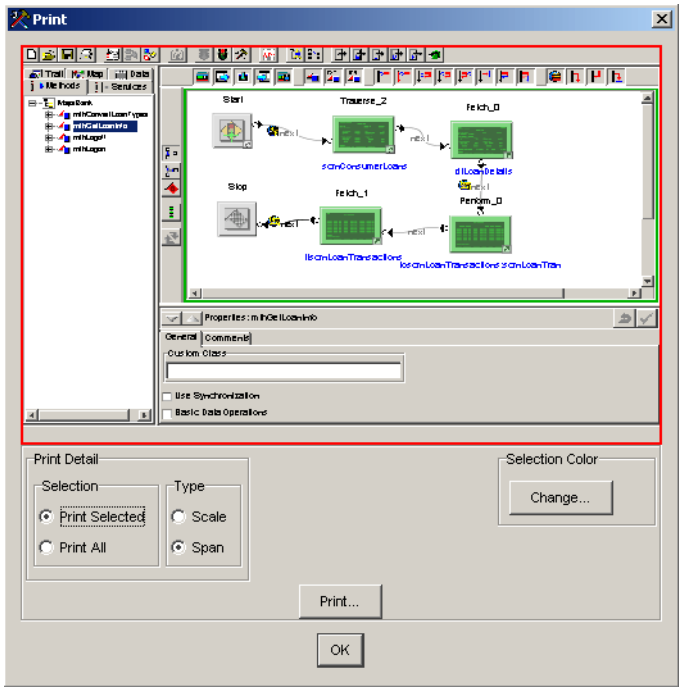


Figure 21. Print Dialog Box

1 Define the **Print Detail** in terms of **Selection**, by choosing one of the following:
   - **Print Selected**—allows you to focus on a specific section of interest. Position the cursor on the appropriate section and click several times, until you zoom in on the section of interest (for example, a single click on the selects both the Presentation view and the Properties view; a second click focuses on the Presentation view, the main tool bar and the Link Mode tool bar, and a third click limits the selection to the Presentation view alone).
   - **Print All** — includes the entire display, consisting of the Tree view, the Presentation view and the Properties view.
2 Define the **Print Detail** in terms of **Type**, by choosing one of the following:
   - **Scale**—scale the map down in size to enable it to be printed on a single page.
   - **Span**—maintain an accurate proportion of the image, extending the output across multiple pages if necessary.
3 In the **Selection Color** section, click **Change** to display the **Choose Selection Color** dialog box and adjust the color settings as needed.
4 Once you are done specifying the print properties, click **Print** to display the standard **Print** dialog box and specify the printer's **Name**, **Print range** and **Number of copies**.

Generating Reports

webMethods JI services can generate a service interface report, which provides information about the methods included in the service, the method inputs, templates, and data fields included with each method, and optionally any output parameters that the method creates. Service interface reports can be generated in text or HTML format. To generate a service interface report, select **Tools** > **Reports** > **Service Interface**. This opens the **Report** dialog box.

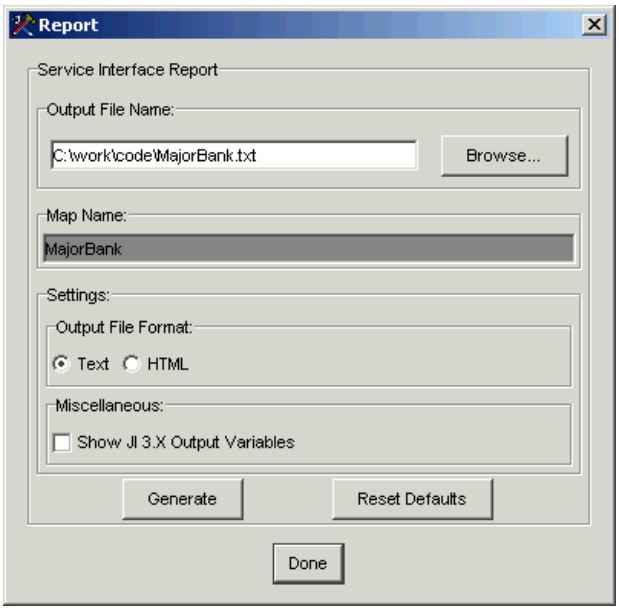


Figure 22. The Report Dialog Box

The **Report** dialog box provides the following options:

Option	Description
Output File Name	The name and optional path for the service interface report. The default value is the location where the code is generated to. See “Default Directories Tab” on page 100 for more information.
Map Name	The name of the current map from which the service interface is generated. This field is not editable.
Output File Format	Select either **Text** or **HTML** for the format of the service interface report.

Option	Description
Show Output Variables	If this checkbox is set, the methods' output variables are included in the service interface report.

After completing the information in **Report** dialog box, click the **Generate** button.

Custom Classes

Within MapMaker, Custom Classes can be defined for various components. Custom Classes are used to create custom code that work with the MapMaker generated service. When a Custom Class is defined, MapMaker creates a "stub" class file, including the constructor and any selected methods, when the service code is generated.

After the stub Java source file is generated, modify the Custom Class source file to add customized behavior. The source file is re-compiled (with the custom behavior included) whenever the Service is re-generated, and the class is included in the generated Jar file. The Custom Class source file is not overwritten during service generation, unless the **Overwrite Custom Code** checkbox is set in the **Generate Code** dialog box. The custom code is also stored in the map file. This allows the map to be moved to a different webMethods JI environment, the service generated and the custom class files recreated. For more information about Custom Classes, see Chapter 11 - "Customizing MapMaker-Generated Service Code" on page 427.

Defining Custom Classes

Custom Classes are created by right clicking on a component that supports custom classes (in either the Tree view or the Presentation view), and selecting **Add CustomClass** from the shortcut menu. The default name of the custom class is `Cust` plus the component name. For example, a custom class on the action `toMainMenu` would be named `CusttoMainMenu`. A custom class can be renamed by right-clicking on it and selecting **Rename** from the shortcut menu. The Custom Class settings are defined in the Properties view.

The following example describes how to add a custom class to a map:

1 In the **Map** tab, right click the map in the Tree view and select **Add CustomClass**.

   A new Custom Class node is added to the map's branch in the Tree.

   The Custom Class name consists of the prefix `Cust` and the name of the object to which it belongs, e.g. **CustMajorBank** (Figure 23).

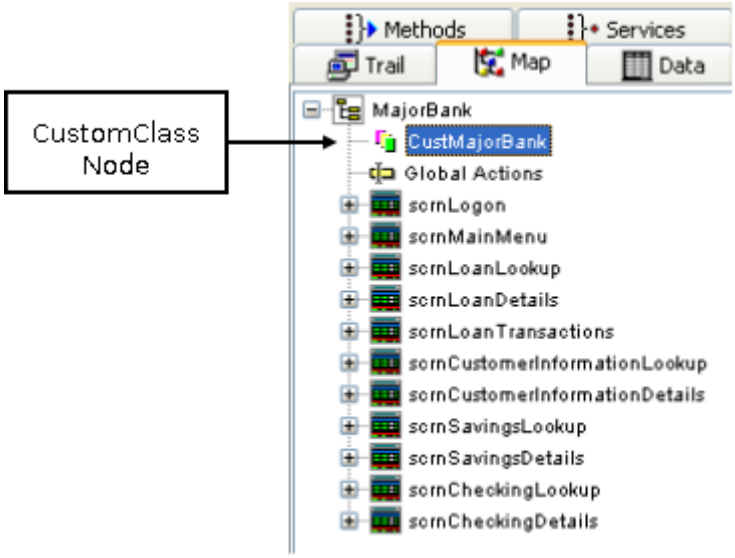


Figure 23. Custom Class Node

2 To define the Custom Class properties, select the Custom Class node in the Tree. The Custom Class properties are displayed in the Properties view. The Custom Class Properties view, allows the selection of methods to be included in the stub code that is generated when code is first generated by MapMaker.

Following is an example of the Custom Class Properties view for the Map class:

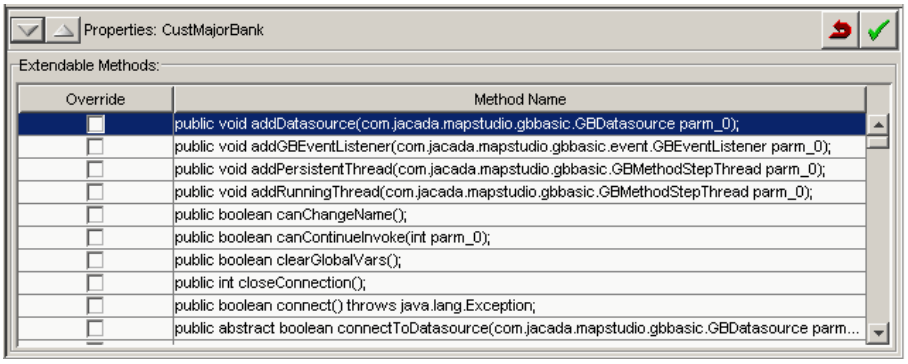


Figure 24. An Example of the Custom Class Properties for the CustMajorBank Class

Select all methods to include in the generated stub and click the **Apply** button. When service code is generated (see "Generating the Service and Test Client" on page 398), stub code for the selected methods is included in the Custom Class file.

Upon subsequent generations of code, the behavior varies depending on the setting of the **Overwrite Custom Code** option in the **Generate Code** dialog box. If this option is set, MapMaker overwrites any existing custom code, including modifications made, with new stub code. If this option is cleared, MapMaker compares the methods selected in the Custom Class Properties view to the methods that exist in the Custom Class source file.

The following describes MapMaker's behavior based on the list of selected methods:

- If there has been no change to the selected methods, then no change is made to the Custom Class source file.
- If additional methods have been selected in the **Custom Class Properties** dialog box, stub code for the new methods is inserted in the Custom Class source file.
- If methods have been cleared in the **Custom Class Properties** dialog box, any custom code for that method is removed from the Custom Class source file. The removed custom code is saved in the map file, and if the method is reselected and code is generated again, the previous code is reinserted into the Custom Class source file.

Note: In order to save the custom code that MapMaker removed from the Custom Class source file, save the map (**File** > **Save**) after generating code.

If an existing Custom Class source file is overwritten, the original source file is saved in an old directory subordinate to the directory where the class files are stored.

Compatibility Mode

webMethods JI features a Jacada Integrator 3.5 compatibility mode. In this mode, existing Jacada Integrator 3.5 features are emulated as much as possible, and many of the new webMethods JI 4.x features are unavailable. Both modes can be mixed within the same Map.

webMethods JI 4.5 provides Jacada Integrator 3.5 compatibility on several levels:

- For Map-level support, see "Map Convertor" on page 115.
- For Method-level support, see "Method Compatibility Mode" on page 115.

- For Input-level support, see “Input Compatibility Mode” on page 116.
- For Step-level support, see “Step Compatibility Mode” on page 117.

Map Convertor

Version 3.5 maps can be converted to version 4.x using the *ea_convertmap.exe* command line utility, located in the *<JI_install_dir>/bin* directory.

To perform the conversion, proceed as follows:

1. Save the 3.5 map to be converted in the *<JI_install_dir>/bin* directory.
2. From the *<JI_install_dir>/bin* directory, enter the following command:

```
ea_convertmap <Map Name>.map
```

The *bin* directory now contains two maps: the version 4.x map, and a backup version 3.5 map named *<Map Name>.map.old*.

Method Compatibility Mode

In the webMethods JI 4.5 version of MapMaker, the 3.5 compatibility mode is set per-method, using the **Basic Data Operations** checkbox, located in the **General** tab of the method’s Properties view (see Figure 25).

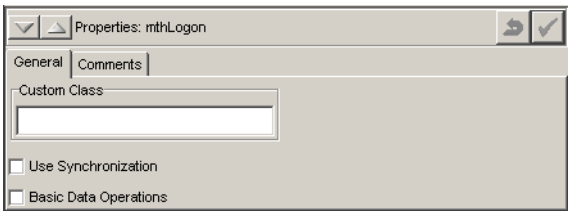


Figure 25. Method Properties > General Tab

By default, compatibility mode is disabled for all new methods created in MapMaker. Enabling this setting restores 3.5 compatible data operations and disables many of the advanced 4.x data modeling and mapping functionality. This setting is necessary in order to provide the greatest compatibility with Jacada Integrator 3.5-style MapMaker maps and/or clients.

When a version 3.5 map is converted to version 4.x, the 3.5 map's methods are converted into 3.5 compatibility mode methods in the converted 4.x map. For example, the **Basic Data Operations** checkbox is automatically set and the appropriate terminator steps (Stop, EndIf, Continue and EndThread) are automatically added to each method.

Method Variables cannot be added to 3.5 compatibility mode methods, that is, methods for which the **Basic Data Operations** checkbox is set. If desired, it is possible to convert a 3.5 compatibility method into a full version 4.x-style

method. To do this, clear the **Basic Data Operations** checkbox, define and set the necessary input and output data types, and add the desired data mapping definitions.

Input Compatibility Mode

Inputs are always associated with data sources. Inputs that are compatible with version 3.5 do not support complex data structures. Accordingly, when such an Input is defined, the **Input Type Selection** section of the **General** tab in the Input Properties view of version 3.5 compatible inputs (Figure 26) is disabled. Instead, the Input Properties include a **Basic Data Operations** section, which defines the input using a simple data source, i.e. a **Global Variable** whose type is **Internal Object**.

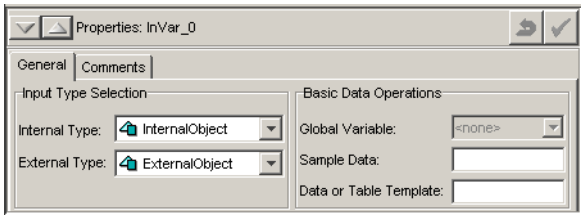


Figure 26. Input Properties > General Tab

Output Compatibility Mode

In version 3.5, Output objects were not commonly used. The returned data were collected and automatically returned to the user in a single return message, without being explicitly specified, filtered, transformed or operated upon.

Accordingly, the Output **Type Selection** section of the **General** tab in the Output Properties view of 3.5 compatible outputs (Figure 27) is disabled.

Instead, the Output Properties include a **Basic Data Operations** section, which defines the Output using a simple data source, i.e. a **Global Variable** whose type is Internal Object.

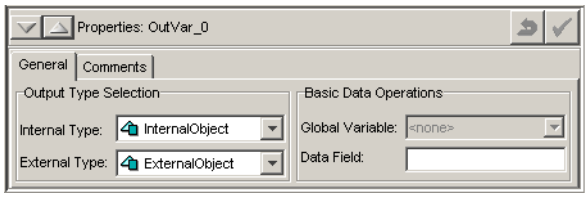


Figure 27. Output Properties > General Tab

Step Compatibility Mode

When a method is in Compatibility Mode, it determines the availability of the following steps' Compatibility-related properties:

- Set step (Figure 28): the **Output Data Assignment** option is enabled in the **General** tab.

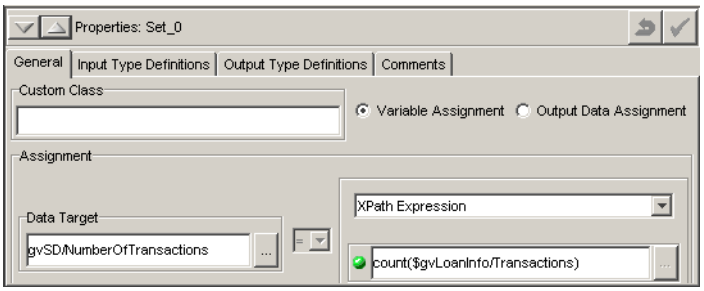


Figure 28. Set Step Properties > Enabled Output Data Assignment Property

- ExternalInvoke step (Figure 29):
  - The **Metadata** tab is disabled.
  - The **Share Global Variables** property is enabled.

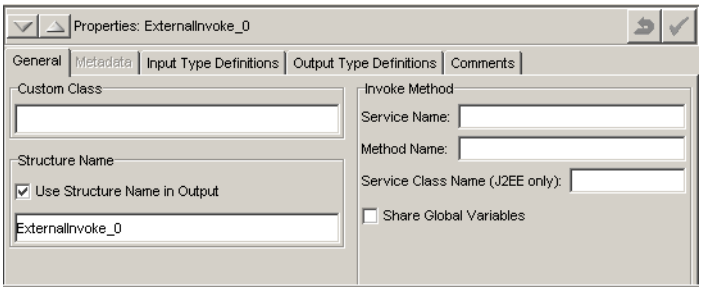


Figure 29. ExternalInvoke Properties > Disabled Metadata Tab

Batch Mode

MapMaker's command line batch mode can be used to script or automate tasks that would normally be performed interactively through the MapMaker GUI. Most of the tasks that can be performed in the GUI can also be performed via batch mode.

A batch mode task is invoked by specifying MapMaker's *-batch* command line option with a batch properties file and the map to operate on:

```
ea_mapmaker -batch <path_to_props_file> <path_to_map>
```

The absolute path to both the map and the properties file must be specified on the command line. MapMaper reads the properties file to determine which tasks should be performed, then opens the map and performs the task specified in the

properties file. Multiple tasks may be performed in a single batch run but each task will only be performed once, esentially allowing the user to script the entire load, update, generate and deploy process for a single map. For more information on the MapMaker command line options and the batch mode properties, see the MapMaker man page in the *webMethods JI Supplemental Reference Guide*.

MapMakerr's batch mode is not intended to replace the normal interactive development process: recording trails or developing services and methods. It will make the process of mass deployments less dependent on human interaction.

Exporting

The SDFX export facility allows a webMethods JI map or screens to be exported into Software AG's SDFX format and subsequently imported into other Software AG products.

Exporting Maps to SDFX

An entire map can be exported to an SDFX file by loading the map. The following steps outline how to export a map to an SDFX file:

1. If the map is not already open, select **File** > **Open** to open an existing map for export.
2. Select **FIle > Export**. The **Export from Map** dialog box is displayed (Figure 30):

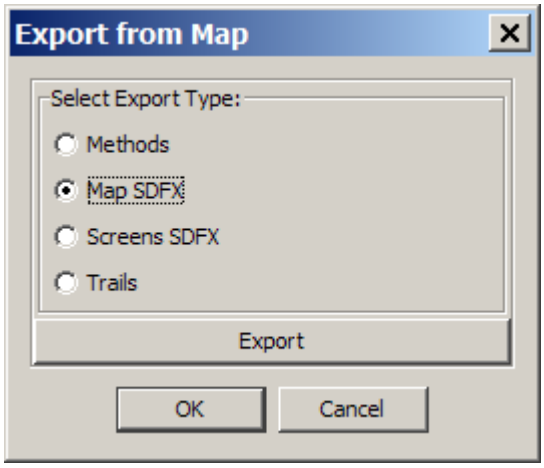


Figure 30. Export from Map dialog box

3. Select the **Map SDFX** radio button.
4. Click **Export.** The **Select File for Export** file browser is displayed. The default file name will be the name of the map with a .sdfx extension in the same directory the map was loaded from.

5 Once you are satisfied with export file name and location, click **Export**. The map will be exported and the **Finished Exporting** dialog box will be displayed (Figure 31):

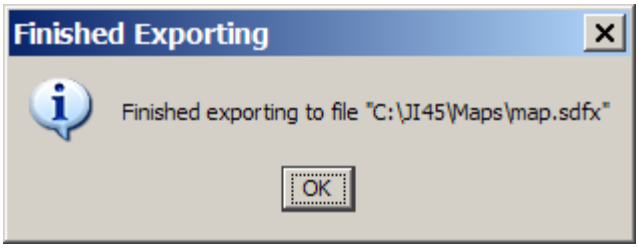


Figure 31. Finished Exporting dialog box

6 Click **OK.**

7 Click **OK.**

Exporting Screens to SDFX

One or more screens can be exported from a map to an SDFX file. The following steps outline how to export screens from a map to an SDFX file:

1 If the map is not already open, select **File** > **Open** to open an existing map for export.

2 Select **FIle > Export**. The **Export from Map** dialog box is displayed (Figure 30):

3 Select the Screens SDFX radio button.

4 Click **Export.** The **Export SDFX** dialog box is displayed (Figure 32):

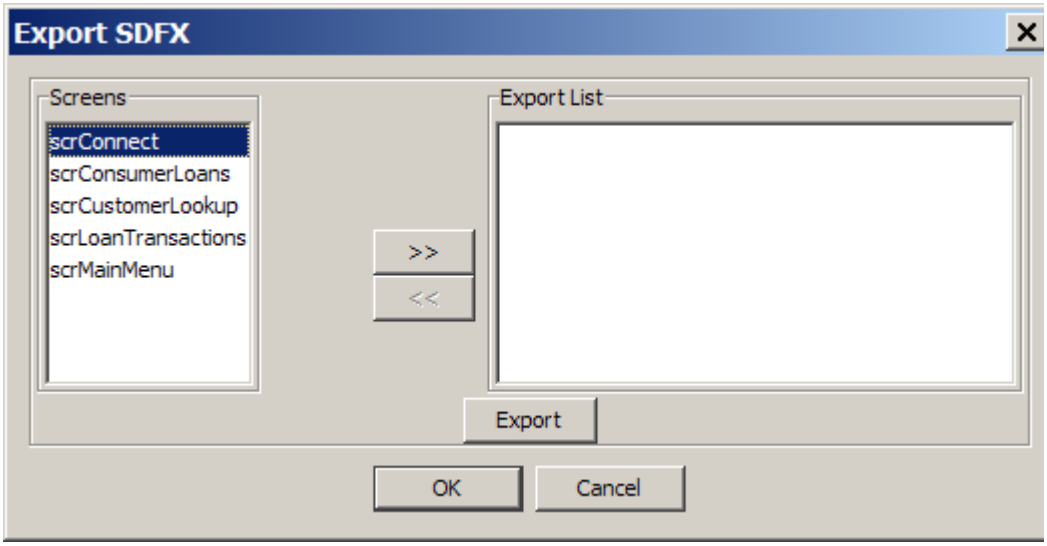


Figure 32. Export SDFX dialog box

5 Select the screens you wish to export from the **Screens** panel and click the **>>** button to move them to the **Export List** panel
6 Click **Export.** The **Select File for Export** file browser is displayed. The default file name will be the name of the map with a .sdfx extension in the same directory the map was loaded from.
7 Once you are satisfied with export file name and location, click **Export**. The map will be exported and the **Finished Exporting** dialog box will be displayed (Figure 33):

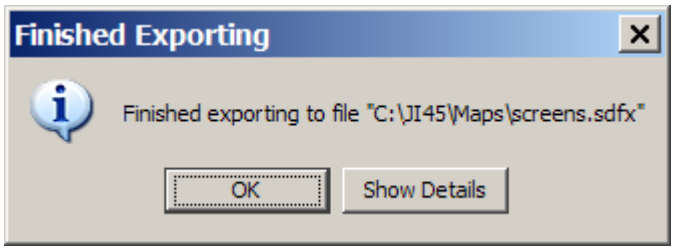


Figure 33. Finished Exporting dialog box

8 Click **OK**.
9 Click **OK**.
10 Click **OK**.

MapPlayer

MapPlayer is a tool included with MapMaker that allows generated maps for TN3270 and TN5250 to be played back. When playing maps, MapPlayer serves as a host simulator, and can be used by a terminal emulator (such as MapMaker) to interact with the map as it is played back.

Note: MapPlayer currently cannot play back maps that were generated using Character Mode.

Starting MapPlayer

MapPlayer can be started from the command line. Additionally, in Windows, MapPlayer can be started using shortcuts, if shortcuts were created during webMethods JI installation.

Command Line

MapPlayer can be started from the command line by executing the following command at the command line:

```
cd <JI_install_dir>\bin
.\ea_mapplayer
```

> **Note:** This executable uses a LAX file to specify the runtime behavior of the application. If certain changes occur to the system, it may be necessary to edit the ea_mapplayer.lax file. An example of this could be, using a different Java Runtime Environment (JRE) than the one selected during webMethods JI installation.

The MapPlayer Interface

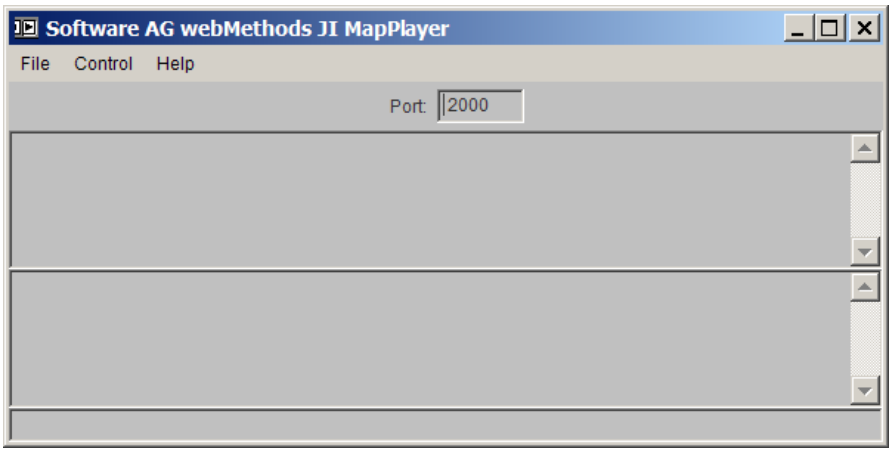


Figure 34. The MapPlayer Interface

MapPlayer Menus

The MapPlayer interface contains three menus, **File**, **Control**, and **Help**. The **File** menu is used to open map files and to exit MapPlayer. The **Control** menu is used to start or stop the open map file, and to determine if actions should be enforced. The **Help** menu provides information about MapPlayer.

Enforce Actions

If **Enforce Actions** is selected in the **Control** menu, MapPlayer requires that all data input by the user, including the information in all fields on the screen, along with the AID key, are identical to the data that was recorded when the map was created. Otherwise, MapPlayer will not be able to perform the appropriate action.

If **Enforce Actions** is not selected, MapPlayer determines an action match based only on matching the AID key that was sent. If there are multiple actions using the same AID key, the first AID key match in the action list is the one used.

Regardless of the **Enforce Actions** setting, if no action match is found, MapPlayer defaults to performing the first action listed for the screen.

MapPlayer Port Identification

The **Port** text box identifies the port on which MapPlayer listens for connections. Default is 2000.

MapPlayer Status Windows

There are two status windows in MapPlayer. The top status window identifies the connections to MapPlayer. The host name and IP address, as well as the port number of the terminal emulator, are identified. The lower status window represents any activity that MapPlayer has executed. The date, time, and type of activity is logged to this window.

Using MapPlayer to Play a Map in MapMaker

To play back a map in MapMaker, perform the following steps:

1. Open the map in MapPlayer. To do this, select **File** > **Open** and select the map from the dialog box. MapPlayer loads the map. This could take a few seconds depending on the map's size.
2. Start playing the map. In MapPlayer, select **Control** > **Start**.
3. In MapMaker, create a Host Profile that connects to MapPlayer by selecting **File** > **Hosts** to open the **Hosts** dialog box. Click **New** and enter the host name or IP address on which MapPlayer is running, and enter the MapPlayer port number. If MapPlayer is running on the same machine as MapMaker, `localhost` can be entered for the host name.
4. Connect to MapPlayer by selecting **File** > **Connect** (a new or existing map must be open).
5. In MapMaker, walk through the map. Because MapPlayer is playing a map, and is not a real TN Server, the session is not actually interactive. As a result,

perform the exact keyboard operations and AID keys that are required by the map.

6 When completed, select **Control** > **Stop** to stop playing the map.

Actions recorded with MapMaker and saved in the map are used by MapPlayer to determine when to traverse to the next screen and which screen and/or snapshot are accessed by the player. Actions can contain both text entered into text fields along with AID keys that advance to the next screen.

As a result, when playing maps in MapPlayer, text and AID keys should be entered exactly as they were entered when the map was originally recorded in MapMaker.

Note: When using MapPlayer, any **Initial Connect Screen** setting (set in "Map Properties" on page 147) is ignored and the first screen in the first trail is assumed to be the initial screen for connecting to the host.

To enable MapPlayer to play back a map, it cannot contain any changes that have not been updated. To play maps created using previous versions of MapMaker, load the map into the current version of MapMaker, update the map and save it, before playing it in MapPlayer.

There are some host actions that cannot be simulated by MapPlayer and thus not all maps can be successfully played in MapPlayer.

Chapter 4. Creating Maps in MapMaker

The Mapping Process

MapMaker is used to generate maps that are used to automatically generate interfaces to legacy applications. These interfaces can be used as webMethods JI services. MapMaker maps consist of Java source files that contain all the logic required to navigate and interact with legacy applications.

The mapping process consists of two phases: trail recording and mapping. During the trail recording phase, the MapMaker user interacts with legacy screens using the terminal emulation capabilities of MapMaker. This interaction mirrors a normal user session with the legacy application. This can be accomplished with a single trail or with multiple trails. If multiple trails are used, MapMaker combines them into a single map of the legacy application.

The mapping phase begins after the trails have been collected. During this phase MapMaker analyzes the trails according to rules which define the mapping process. Usually, no additional intervention is required. In some cases, however, developer productivity is greatly improved if screens and fields are given meaningful names. For those cases where the host application exhibits complex behavior or complex data layouts, additional mapping rules, such as screen tags may be specified to identify similar screens.

The steps required to map a legacy application and produce a webMethods JI service are as follows:

1. Define the host profile.
2. Connect to the host.
3. Create one or more trails by navigating through the legacy application.
4. Update the map file based on the trail information. Create tags and/or disable fields if necessary to customize the map and ensure that MapMaker properly distinguishes distinct screens, and groups similar screens together. Please note that the rules for screen recognition and screen combination do vary between Block Mode and Character Mode.
5. Specify the location of legacy data that needs to be retrieved and locations where input will be provided to the legacy application. This is done by adding data templates, table templates, and data fields.
6. Define methods and identify input and output variables for the method, as well as navigational steps that will be included in the method.
7. Define services that utilize the methods.
8. Generate and deploy the services.

The following sections detail these steps for using MapMaker.

Defining Communications Connections

To access external hosts from within MapMaker, connections must be identified using the **Hosts** dialog box. MapMaker supports two data stream modes for communication with legacy hosts. These data streams are Block Mode and Character Mode. Below is a brief explanation of these modes and the applications that use them.

Block Mode Data Streams

Block mode data streams are used by TN3270 and TN3270E (Models 2-5), TN5250 Standard (80 and 132 column), and partial support for TN5250E connections with the legacy host. This mode is called "Block Mode" because the legacy host sends the data in logical blocks (also called RUs). Block mode is referred to as TN3270/TN5250 throughout this chapter. In this mode, after a trail has been captured and the "Update Map" action is performed, each snapshot in the trail is scanned for fields. These fields are compiled into a list and used as a signature (by using the offsets and lengths) to differentiate the snapshots into a list of screens. All snapshots with no fields are grouped together under a single screen.

Note: The support for TN5250E exists to allow definition of session names.

NVT Mode Support

Network Virtual Terminal (NVT) mode support is provided when running a block mode (3270 or 5250) emulation to logon to a front end or TN3270/TN5250 server using simple ASCII character based protocol. All data received before being put into block mode (either by Binary/EOR or TN3270E) is considered to be NVT. Once the logon is completed, the server puts MapMaker into block mode (TN3270/TN3270E/TN5250). Only the NVT mode functionality necessary to support communications during session startup and with TN3270/TN5250 servers is provided.

As such the following restrictions apply:

- Global actions are not supported in NVT mode.
- MapPlayer does not play maps containing NVT data.
- Once MapMaker is put into block mode, it cannot return to NVT mode.
- TN3270E NVT-DATA type is not supported.

Keys Sent to Host

Only alphanumeric keys, the **Enter** key (carriage return), and the **Backspace** key, are sent to the host. All others (**PF1**, **Tab**, **Esc**, etc.) are ignored.

Character Processing in NVT Mode The following table shows how characters are processed by MapMaker while in NVT mode:

Character	Action
Carriage Return (0x0D)	Move the cursor to the left column.
Line Feed (0x0A)	Move the cursor to the next row. If the cursor is on the last row of the screen, the whole screen is scrolled up one line.
Backspace (0x08)	Moves the cursor left, one column, unless the cursor is already at the left margin.
Alphanumeric ASCII characters (0x20 through 0x7E)	The character is placed on the screen at the current cursor position and the cursor is moved ahead one position. If the cursor was at the maximum column position, the cursor is positioned to column one. If the cursor was at the maximum column and maximum row, the screen is scrolled up one line.
All other characters	Ignored

Snapshots A snapshot is taken only when the **Enter** key is pressed. This is not configurable, as it is in character mode. The snapshot contains one artificial field which contains the whole screen (as in character mode).

Building a Map

As in character mode, NVT snapshots end up as separate screens because the screen is compared character-by-character. In block mode, RUs are counted and when the correct number is reached, MapMaker determines that it may perform the next action. NVT mode does not have RUs, hence they are not available to determine when all the data has been received from the host.

Therefore, a static or scroll tag must be added to NVT screens so MapMaker knows when to send the Action Input data. Cursor Settle may also be used in NVT screens. The Action Input and Action Destination screens look similar to character mode maps, except that the Action Destination for an action transition

from an NVT screen to a block mode screen looks like a regular block mode action (i.e. “To <screen> after N”). Also, in the Action Input screen, the Action Key list is a subset of the keys that are allowed in character mode.

NVT to Block Mode Transition

When transitioning from NVT mode to block mode, MapMaker counts RUs the same as it would for a block mode-to-block mode transition, except that all carriage returns received from the host are counted as an RU.

The following example illustrates this:

The server requests MapMaker for a login and password, and the login has already been sent. <CR> = carriage return and <LF> = line feed.

1. The server sends MapMaker the string “`Password:`” in ASCII.
2. MapMaker sends the host “`guest<CR>`”.
3. The server sends MapMaker <CR><LF><CR><LF>.
4. The server negotiates block mode.
5. The host sends two block mode RUs.

The Action Destination screen would then show “`To Screen_2 after 4`". The number 4 is derived from the two <CR>s in Step 3 and the two RUs in Step 5. This is the number of RUs MapMaker waits for before assuming that it can send the data in the next action. After an NVT mode-to-block mode transition, you cannot return to NVT mode.

Character Mode Data Streams

Character Mode data streams are obtained by making a Telnet connection to the legacy host. This mode is called “Character Mode” because the legacy host sends the data character by character (there are no logically delimited blocks of data). Character mode is referred to as such throughout this chapter. Character mode emulations have no fields to use as a signature to differentiate the snapshot into a list of screens. For this reason, each character mode snapshot is given one artificial field that takes up the entire screen.

When the “Update Map” action is performed, instead of using the offset and length of fields as a signature, the entire snapshot screen is compared character by character and all matching snapshots are grouped together under a screen. However, this artificial field can be disabled to prevent the character by character comparison. By disabling the artificial field, adding a tag to the screen and performing the “Update Map” action, screens that are identical (such as any UNIX Shell prompt screen) can be combined.

Character mode data streams have support for the DEC VT100/220/320 “Special Graphics Set”, also known as the “VT100 Line Drawing Character Set”. These contain line drawing characters and other special characters. MapMaker and Java

Services do support Keypad Application Mode, however this support is not currently extended to the PATerm. Host downloading of user-defined keys is not supported, along with some additional DEC-specific functionality.

Note: MapPlayer does not support Character Mode maps.

To define host connections, select **File** > **Hosts**. The **Hosts** dialog box opens:

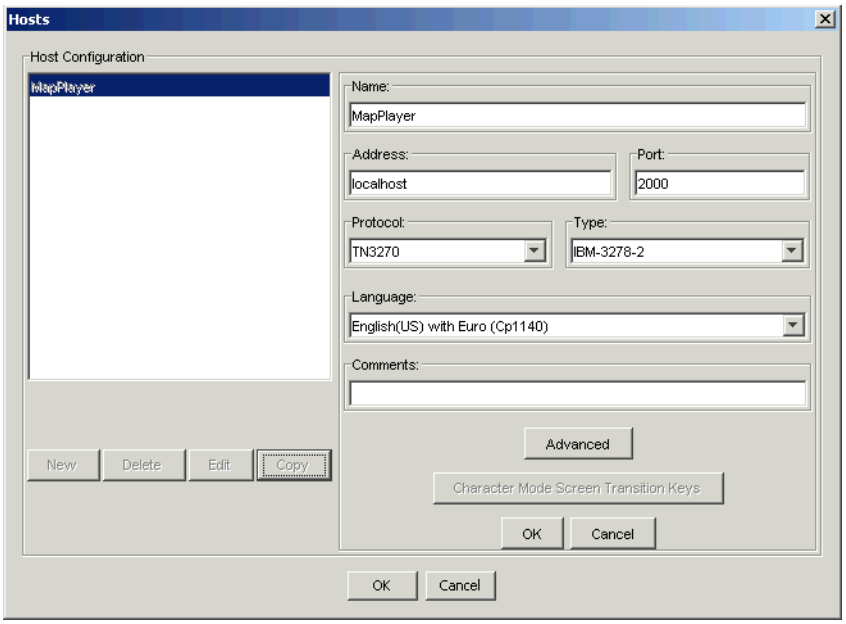


Figure 35. Hosts Dialog Box

The **Hosts** dialog box offers the following options:

Option	Description
Host Configuration	The left pane of the **Hosts** dialog box lists all host profiles that have been defined. The host profile list is managed using the following buttons: • **New:** Click the **New** button to create a new host profile. After creating a new profile, change the name, enter the address, and change any other information that is required. • **Delete:** Click the **Delete** button to delete an existing profile. • **Edit:** Click the **Edit** button to edit the host profile information for an existing profile. • **Copy:** Create a copy of the selected host configuration. Use this option to easily define a new host that has similar properties to an existing host, by modifying only a few details that are unique to the new host.
Name	The name of the profile.
Address	The IP address or DNS name of the legacy host.
Port	The port number of the host's TN3270, TN5250, or Telnet server. The default entry is 23.
Protocol	The Communications Protocol drop-down list is used to select the protocol type. This affects the model type selection options and how the Advanced screens are defined. Available options are: • **TN3270-Block Mode** • **TN5250-Block Mode** • **Telnet-Character Mode**

Option	Description
Type	Select the model type. Allowable options are: • For TN3270/TN3270E, select IBM-3278-2, IBM-3278-3, IBM-3278-4, or IBM-3278-5. • For TN5250, select IBM-3179-2, IBM-3477-FG, or IBM-5555-C01 • For Telnet, select DEC-VT100, DEC-VT220, DEC-VT320, or CUSTOM
Language	Select the national language used by the host. For information about customizing the national language conversion used by webMethods JI, see the *Supplemental Reference Guide*. **Note:** National language support is only defined for 3270 and 5250 emulations. This option is disabled for Character Mode, which uses ISO Latin-1.
Comments	Enter any comments related to the host profile.
Advanced	The **Advanced** button is used to configure parameters based upon the Protocol selected above. For more information see "Advanced Host Configuration" on page 131.
Character Mode Screen Transitions Keys	This opens a check box panel for selecting the keys to be used for taking snapshots. This is a Character Mode feature and is not available for TN3270 or TN5250.

Advanced Host Configuration

The **Advanced** button opens the **Advanced Hosts** dialog box, whose options vary as a function of the selected **Protocol**:

- For TN3270 emulation, the **Device Name** can be configured.
- For TN5250 emulation, the **User Name** can be configured.
- For Character Mode emulations, **Terminal Type String Information** can be configured.

TN3270 Configuration

This **Advanced** button displays this option when the selected **Protocol** is TN3270.

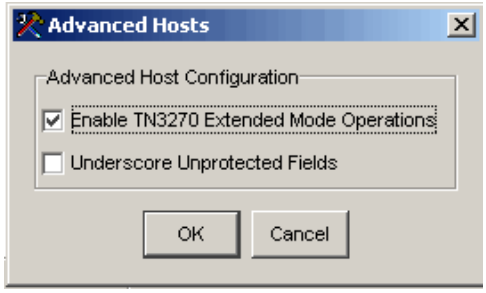


Figure 36. Advanced Hosts > TN3270 Configuration

The following options can be entered here:

Option	Description
Enable TN3270 Extended Mode Operations	Enable or disable TN3270E functionality.
Underscore Unprotected Fields	Set to underscore unprotected fields that the user can write in.

TN5250 User Name Configuration

This dialog box is used to access TN5250E User Name functionality.

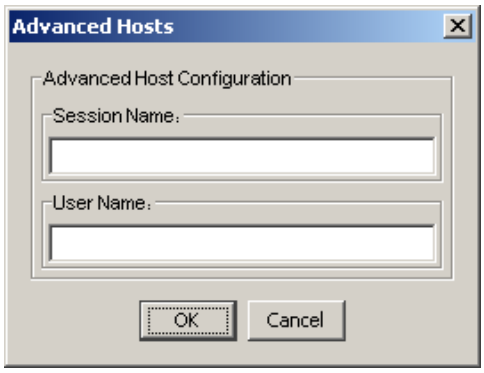


Figure 37. User Name Configuration

The following option can be entered here:

Option	Description
Session Name	The name of the session associated with this host.
User Name	The name of the user associated with this host.

Non-Custom Configuration for Character Mode

This dialog box is displayed when the **Protocol** is **Telnet**, and the **Type** is *NOT* **Custom**.

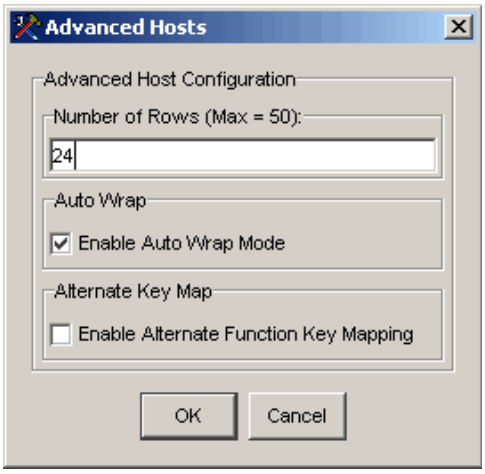


Figure 38. Non-custom Configuration for Character Mode

The following options can be entered here, if the non-custom configuration criteria have been met:

Option	Description
Number of Rows	The number of rows allowed for the emulation. Maximum value is 50.
Auto Wrap	Set **Enable Auto Wrap Mode** if you want the cursor to move to the first position of the next line as soon as the cursor reaches the end of a line.
Alternate Key Map	Set **Enable Alternate Function Key Mapping** in order to use your customized set of function keys.

Custom Configuration for Character Mode

This dialog box is displayed when the **Protocol** is **Telnet**, and the **Type** is **Custom**.

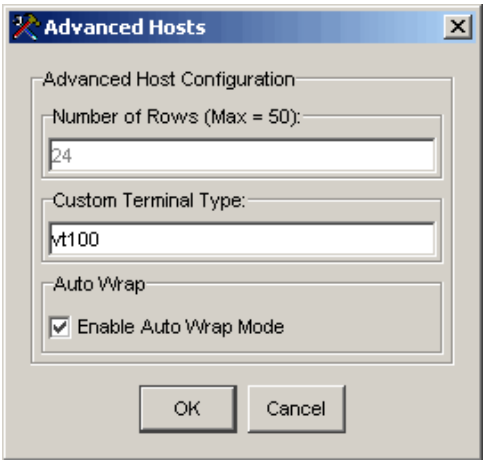


Figure 39. Custom Configuration for Character Mode

The following options can be entered here, if the custom configuration criteria have been met:

Option	Description
Number of Rows	The number of rows allowed for the emulation is set here. The value is 50.
Custom Terminal Type	Enter the type of custom terminal being used. **Note:** This string must correspond to information in the telnet.ti file.
Auto Wrap	Set **Enable Auto Wrap Mode** if you want the cursor to move to the first position of the next line as soon as the cursor reaches the end of a line.

Character Mode Screen Transition Keys

Screen transitions are what cause a snapshot to be taken. In TN3270/TN5250 mode, the standard IBM AID keys are used as a trigger. In character mode, there are no AID keys. Because of this, keys that trigger a snapshot to be taken, are defined by a configurable list. This list is configured in the **Character Mode Screen Transition Keys** option in the **Host** dialog box.

This button opens a dialog box that contains checkboxes for selecting the keys to trigger a snapshot. The default settings, as shown below, can be restored at any time by simply selecting the **Restore Defaults** button.

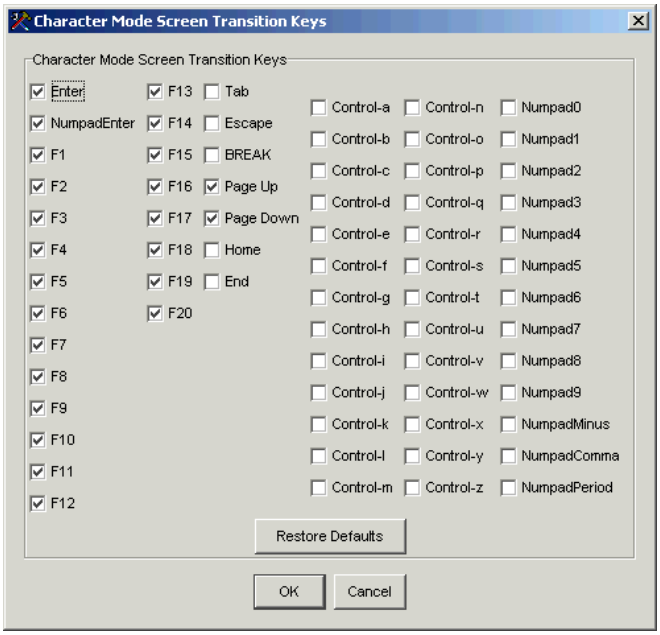


Figure 40. Character Mode Screen Transition Keys

After adding or editing a host profile, click the **OK** button to save the changes, or click the **Cancel** button to cancel the changes.

Connecting to the Host

After the host profile is defined, connect to the host using one of two different methods:

- Start a trail first, which automatically prompts for a host connection.
- Connect to the host and then start a trail at the appropriate time.

When creating the initial trail, the login screen and initial navigational information is needed in the trail. It is recommended that the first trail be started prior to connecting to the host.

Subsequent trails may not require the login screen. For example, trails can be created that traverse from various screens back to the main menu. In this case, connect to the host first, navigate to the correct screen, and then start trail recording, perform the navigational steps, and stop trail recording upon return to the main menu.

Note: For this release, when recording Character Mode trails and using scroll tags, the login sequence should be included in subsequent trails to allow MapMaker to perform the proper analysis.

These steps outline how to connect to the host to record the initial trail:

1. If the map file is not already open, select **File** > **New** to create a new map file.
2. Select **Map** > **Start Trail** or click the **Start Trail** button ( ) to start the trail recording mode. The **Connect to Host** dialog box is displayed (Figure 41):

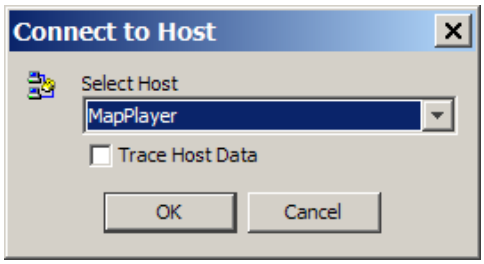


Figure 41. Connect to Host Dialog Box

Note: To connect to the host without starting trail recording mode, select **File** > **Connect**. This also opens the **Connect to Host** dialog box.

3. Select the host from the **Select Host** box.
4. If you wish to record the entire data stream for debugging purposes, set the **Trace Host Data** check box.
5. Click **OK**.
   The initial host screen is displayed in the Presentation view (Figure 42).

Note: If the **Trace Host Data** checkbox is set, the trace of the interaction with the legacy host is logged into the *mapmaker_stdout.txt* file, in *<JI_install_dir>/logs*.
In order to select the host from the **Connect to Host** dialog box, the host connection must be defined prior to trail recording. For more information about configuring host connections, see "Connecting to the Host" on page 135.

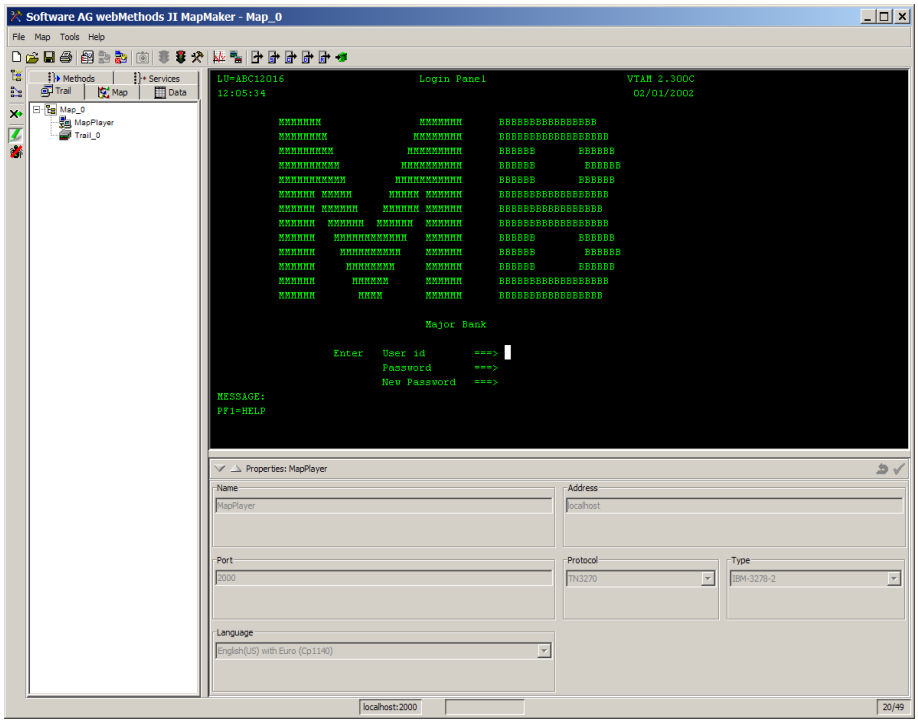


Figure 42. An Initial Host Screen

In trail recording mode, the host screen is interactive, functioning exactly like a terminal emulator. Screens and navigational information, such as keyboard input, are recorded during the navigation.

Trail Recording

A trail is a linear path of all screens encountered while navigating through a host application while in trail recording mode. Each trail is added to the trail repository for the current map, and MapMaker uses these trails to build a comprehensive map of the host application. Trails contain the following information:

- A snapshot image of each screen in the trail, including any formatted fields.
- The keystrokes and AID key(s) typed into each screen.

The main goal of the trail recording phase is to expose MapMaker to all the host screens that are required for the webMethods JI service to interact with the legacy application. Because MapMaker is essentially learning how the host application is controlled, it is important to perform all actions that are required to make the host application perform appropriately at runtime.

Using the Euro Symbol

MapMaker supports the use of the Euro currency symbol in both the Presentation view and in the text fields in MapMaker dialog boxes.

- **Using the Euro currency symbol in Presentation view.** In order to allow the Euro symbol to be displayed in the Presentation view, webMethods JI replaces the international currency symbol with the Euro symbol. This symbol can be entered using the typical keyboard sequence for windows.
- **Using the Euro currency symbol in MapMaker dialog boxes.** The Euro symbol can be inserted into text fields in MapMaker dialog boxes using whatever method the system normally requires for entering the Euro symbol. For example, if the keyboard has a Euro character, it may be input normally. Alternatively, the system may allow the Euro character to be entered using the **Alt** key plus a numeric sequence. See the specific system documentation for more information.

The Trail Tab

The **Trail** tab lists the host name or IP address, followed by all trails recorded for the current map. Expanding a trail shows the screenshots it includes. Similarly, expanding a screenshot shows the fields it includes.

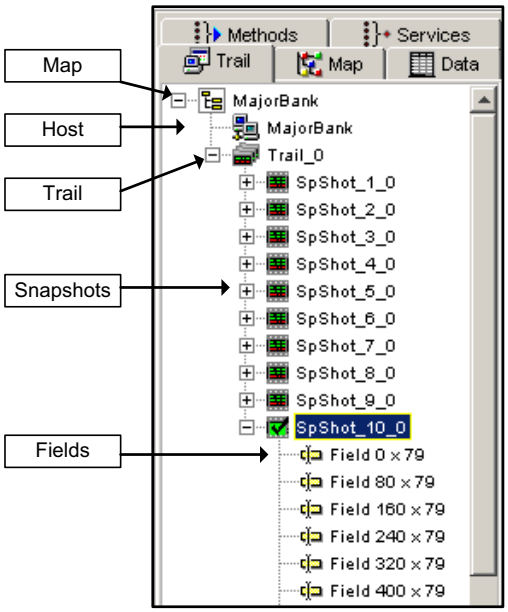


Figure 43. The Trail Tab

The following sections describe the nodes comprising the **Trail** tab.

Host Node

The Host node displays the name of the legacy host. Selecting the Host node indicates whether there is currently an active session on the legacy system:

- If you are connected to the legacy system, the Presentation view displays the current screen in the session and trail recording can continue.
- If you are disconnected from the legacy system, the Presentation view is empty.

Another indication of whether or not the session is active is provided by the toolbar:

- If you are connected to the legacy system, the **Connect Host** button is disabled and the **Disconnect Host** button is enabled ( ).
- If you are disconnected to the legacy system, the **Connect Host** button is enabled and the **Disconnect Host** button is enabled ( ).

> **Note:** Screens can be viewed or disabled within trails while still connected to the host. To return to the live host connection after viewing a screen select the Host node in the Tree view.

To reconnect to the host listed in the **Trail** tree, right-click on the Host node and choose **Connect Host** from the shortcut menu.

To reconnect using a different host and/or port, use the **Connect** button ( ) or select **File** > **Connect**, in order to choose the appropriate host from the **Connect to Host** dialog box.

If the host session is lost during trail recording, the trail is automatically stopped.

Trail Nodes

The Trail nodes list the trails that have been included in the current map. Each Trail node can be expanded to display the screens it includes.

Snapshot Nodes

The Snapshot nodes list the screen snapshots included in each trail. Each snapshot component can be expanded to view the fields it includes.

Selecting a snapshot in the Tree displays the corresponding screen in the Presentation view.

Field Nodes

The Field nodes list the fields that MapMaker automatically detected in the screen. Fields are used by MapMaker to identify the screen. For more information about screen recognition, see "Customizing MapMaker's Screen Recognition" on page 157.

The default field name (which cannot be changed) consists of the field's absolute position on the screen and of the number or characters it contains (for example, the first field in the screen containing ten characters would be named "field 0 x 10").

Selecting a field in the Tree displays that field's position in the Presentation view.

Renaming Trails and Screen Snapshots

After a trail is recorded, the trail can be renamed: right-click on the trail name, select **Rename** from the shortcut menu and enter the new name of the trail. Screen snapshots can be renamed using the same method.

Alternatively, you can rename an object by selecting it and pressing the **PF2** key (**F2** on the keyboard). The object name becomes available for editing, and you can specify the new name and press the **Enter** key to apply the change.

> **Note:** Fields cannot be renamed.

Using Multiple Trails

Trails can begin and end at any point in the host application, and multiple trails can be added to the trail repository. If a trail that was added to the map is incorrect or not used by the webMethods JI service, this trail can be deleted without affecting any other trail. The number of trails can vary according to specific needs. This number has no impact on the runtime performance of the application.

Adding Trails

All current trail recording must be halted prior to starting a new trail. To exit a current trail, select **Map** > **Stop Trail** or click the **Stop Trail** button ( ).

To add a new trail:

1. Connect to the host and navigate through the legacy application to the screen where the new trail starts.

2 Select **Map** > **Start Trail** or click the **Start Trail** button ( ).
3 A new trail is started and added to the trail repository. Begin navigating through the screens to record the trail.

Note: When using Character Mode, a new trail should be started from login to properly capture scrolling data (see "Scroll Tag Configuration" on page 161).

Deleting Trails

To delete a trail, right-click on it and select **Delete** from the shortcut menu.

Importing Trails

You may import a trail from another map. This provides the opportunity for several developers to work simultaneously on the same project.

To import a trail:

1 Select **File** > **Import**. A message box opens, warning that you should create a backup of your map before importing.

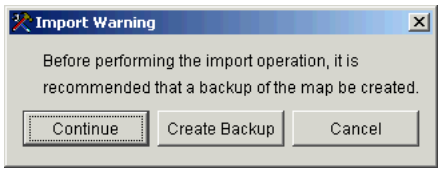


Figure 44. Import Warning

2 Choose one of the following:
- **Create Backup** to create a backup of the current map. The **Save As** dialog box appears. Save the backup map with a different name. Note the creation of a backup file in the designated directory, and the file name in the MapMaker title bar does not change. The backup file is just a "snapshot" of this file before any modification occurs.
- **Continue** to proceed without creating a backup.

The **Select File for Import** dialog box opens (Figure 45):

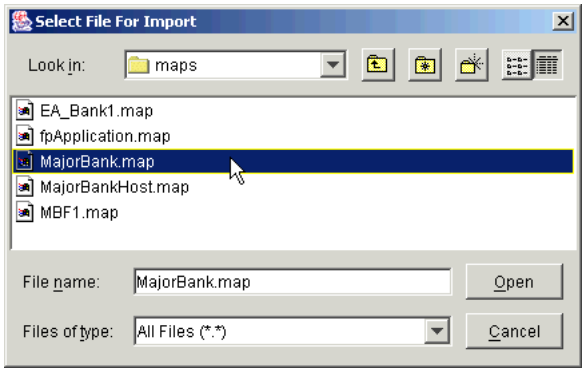


Figure 45. The Select File For Import Dialog Box

3 Select the Map file from which to import a trail, and click **Open**. The **Update Completed** message appears (Figure 46):

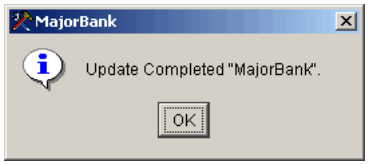


Figure 46. The Update Completed Message

4 Click **OK.** The **Import from:** dialog box opens (Figure 47):

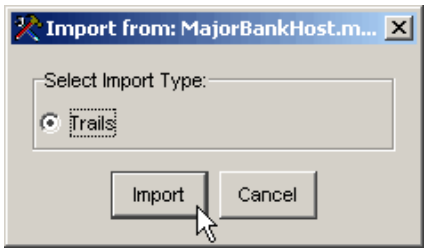


Figure 47. The Import from: Dialog Box

5 Select **Trails** and click **Import**. The **Import Trails** dialog box opens (Figure 48):

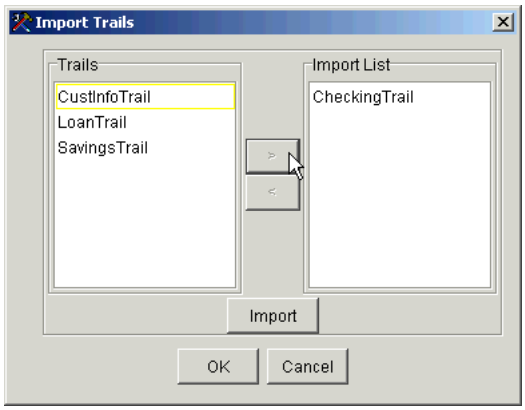


Figure 48. The Import Trails Dialog Box

6 Move the desired trail(s) from the **Trails** list to the **Import List** list. For each trail, select the trail in the **Trails** list and click the right-pointing arrow. The selected trail now appears in the **Import List** list. Repeat this process for all trails to be imported.

> **Note:** You can click **Import** after each time you transfer a trail to the **Import List**. This imports one trail at a time. Another option is to move all the trails to the **Import List** and then import. This imports all the trails at once.

7 Click **Import.** The following dialog box is displayed (Figure 49):

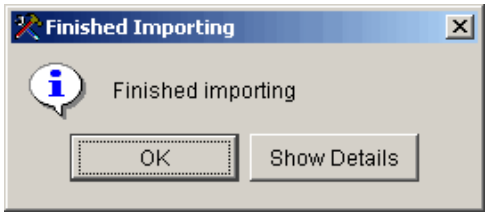


Figure 49. Finished Importing Dialog Box - Trails

The **Import Details** dialog box displays information on the importation as it was handled by MapMaker. The information contained in the details log may be saved as a text file by using the **Save To File** button in the **Import Details** dialog box. It is recommended that the lead developer maintain a history of all map import activities by saving the details file for each import action. A sample **Import Details** dialog box is shown below (Figure 50).

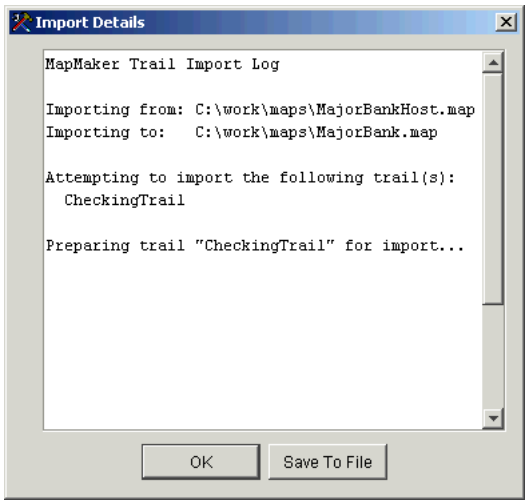


Figure 50. Import Details Dialog Box

8 When the desired trail(s) have been imported, click **OK** to close the **Finished Importing** dialog box.

> **Note:** The **Finished Importing** dialog box is displayed after **every** import operation. When importing trails individually, that is one at a time, the dialog box appears after each import operation.

9 Click **OK** to close the **Import Trails** dialog box.

10 Click **Done** in the **Import from**: dialog box. The **Update Completed** message box is displayed.

The trails are now imported to the current map.

Enabling and Disabling Trails and Screen Snapshots

Occasionally trails are recorded, or include screen snapshots, that should not be included in the map. These can be disabled by right-clicking the component in the Trail tab and selecting **Disable** from the shortcut menu. This instructs MapMaker to ignore this trail or screen when updating the map. If the trail or screen later becomes necessary again, right-click on the component and select **Enable**.

Stopping Trail Recording

After trail recording is complete, select **Map** > **Stop Recording** or click the **Stop Trail** button .

Chapter 5. Customizing the Map

After completing the trail recording, the next step is to assemble the trails into a logical map structure. In this step, MapMaker analyzes the trails to determine the true structure of the host application.

This section describes:

- "The Map Tab" on page 145
- "Map Properties" on page 147
- "Global Variables" on page 150
- "Global Actions" on page 153
- "Screen Properties" on page 154
- "Customizing MapMaker's Screen Recognition" on page 157
- "Using Floating Tags to Identify the Location of Floating Data Fields" on page 164
- "Actions" on page 164
- "Comparing Screen Images" on page 171

The Map Tab

The **Map** tab of the Tree view displays the screens and Global Actions associated with the map. Subordinate to each screen component are Fields, Tags, Actions, and Snapshots.

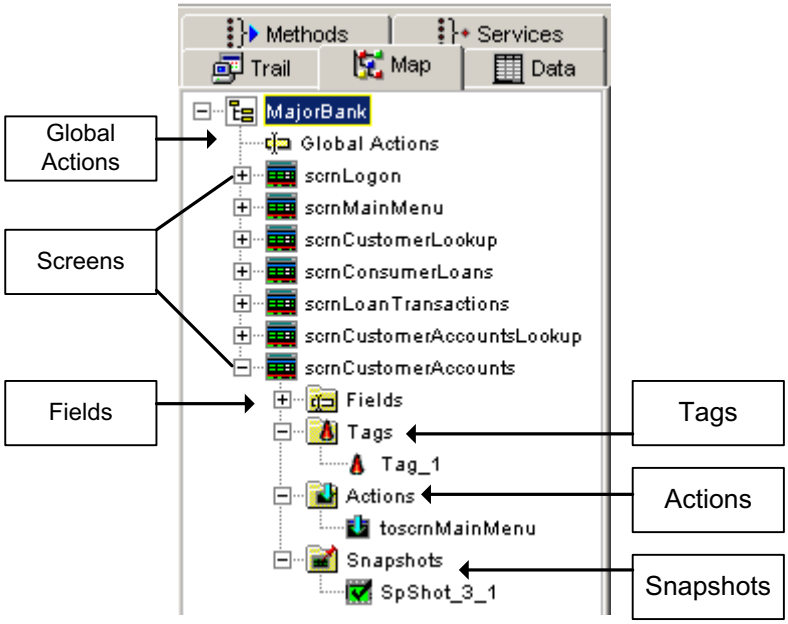


Figure 51. The Map Tab

The **Map** tab offers the following options:

Component	Description
Global Actions	Global actions are used to define a default AID key or other entries that should be performed when there has been an error and the legacy host is in an unknown state, or to complete other actions such as paging actions for Table Templates. For information about adding global actions, see “Global Actions” on page 153.
Screen	The name of the screen. To change the name, right-click on the screen name, select **Rename** from the shortcut menu and enter the new name.
Fields	Lists the fields that MapMaker automatically detected in the screen. Fields are used by MapMaker to identify the screen. For more information about screen recognition, see “Customizing MapMaker’s Screen Recognition” on page 157.

Component	Description
Tags	Tags are used to distinguish between screens that MapMaker unexpectedly identified as identical. For more information about screen recognition, see "Customizing MapMaker's Screen Recognition" on page 157.
Actions	Actions are the keystrokes performed to traverse between screens. For more information about actions, see "Actions" on page 164.
Snapshots	Lists the snapshots included with the screen component. Often, two or more screens in a trail repository are interpreted by MapMaker to be identical. All identical screens are listed together, and only those screens that are unique have their own Screen components in the map. For more information about screen recognition, see "Customizing MapMaker's Screen Recognition" on page 157.

Map Properties

To view or manage the Map-related settings, select the **Map** node (the root) of the Tree. The Map properties are displayed in the Properties view (Figure 52).

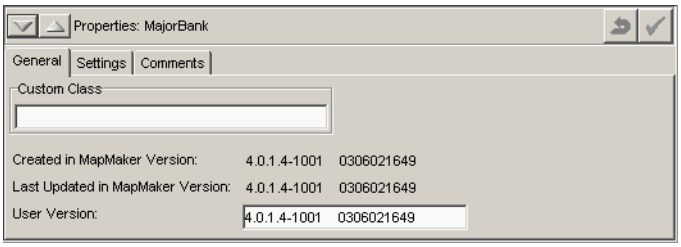


Figure 52. Map Properties View > General Tab

The **General** tab of the Map Properties view offers the following options:

Option	Description
Custom Class	The custom class that should be used with the map. For more information about custom classes, see "Custom Classes" on page 112. **Note:** If a custom class is created for a map, the name of the custom map class, rather than the name of the map as defined in MapMaker, should be used in the Configuration Manager when identifying the class name of the map during service configuration.
Created in MapMaker Version	The version of MapMaker in which the map was originally created.
Last Updated in MapMaker Version	The version of MapMaker that was used the last time an **Update Map** operation was performed.
User Version	An optional field, allowing you to assign a symbolic name for this version of the map. The default is identical to **Created in MapMaker Version**.

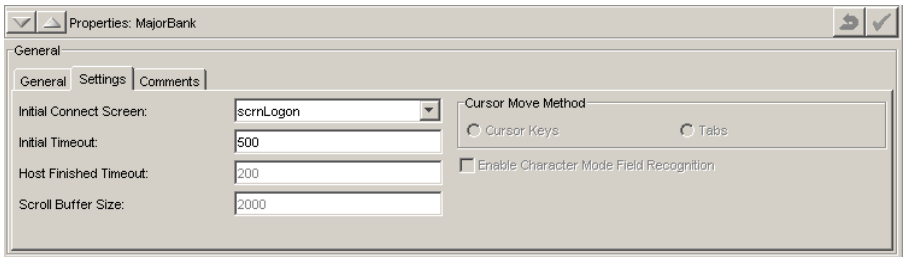


Figure 53. Map Properties View > Settings Tab

The **Settings** tab of the Map Properties view (Figure 53) offers the following options:

Option	Description
Initial Connect Screen	The screen that MapMaker should use as the initial screen to connect to the legacy host. Normally, MapMaker assumes that the first screen listed in the map is also the initial screen (the first screen displayed when connecting to the host). However, if multiple trails are recorded, it is possible that the first trail does not contain the initial connection screen. Rather, a subsequent trail is used to connect to the host. If this is the case, the first screen in the map is not the initial connection screen. This option allows the selection of the initial connection screen from the drop-down list. Because this option defaults to the first screen in the map, only select an initial connection screen if it is not the first screen.
Initial Timeout	The amount of time mapmaker waits (in milliseconds) after sending data to the host before starting to check for cursor settle or a scroll tag. **Note:** This option is only available for Telnet applications. Default: 500 milliseconds.
Host Finished Timeout	The amount of time mapmaker waits (in milliseconds) before assuming that the host has stopped sending data. **Note:** This option is only available for Telnet applications. Default: 200 milliseconds.
Scroll Buffer Size	The size of the scroll buffer (see "Scroll Tag Configuration" on page 161). **Note:** This option is only available for Telnet applications. Default value is 2000 characters.

Option	Description
Cursor Move Method	Indicates whether the host expects **Cursor Keys** or **Tabs** as the method for moving the cursor between fields. **Note:** This option is only available for Telnet applications.
Enable character Mode Field Recognition	Determines whether or not screens are matched based on visual attributes (e.g. color). **Note:** This option is only available for Telnet applications.

Global Variables

During trail recording, MapMaker saves all the information that was typed into fields (Block Mode) or sent to the host (Character Mode), and incorporates this information into actions included with the map. By default, MapMaker constructs an action using all the keystrokes entered into fields or sent to the host during trail recording, as well as the AID key (TN3270/TN5250 only) pressed to complete the operation.

Rather than storing the data typed into fields during trail recording in the action, a Global Variable may be used. For example, the text representing the user ID and password can be replaced with variables. These fields are not hard-coded in the webMethods JI service, but instead are passed as variables from the client application. This is accomplished by adding Global Variables to the map, and assigning those variables to action components in the map. For information about assigning Global Variables to actions, see “Modifying Actions” on page 165.

Later, methods can make use of these Global Variables and actions to pass variable parameters from the client to the legacy screen. For more information about methods, see “Managing Methods” on page 277.

Variables can also be added on a method level, meaning that they apply only to the specific method in which they are defined. For more information, see “Method Variables” on page 282.

Adding Global Variables to the Map

To add a Global Variable to the map:

1 Select **Tools** > **Edit Business Entities** from the menu, or click on the toolbar.

The **Business Entity Editor** opens, displaying the **Global Variables** tab (Figure 54).

Note: Global Variables can also be added from the **Edit Variables** dialog box. For more information, see "Method Variables" on page 282.

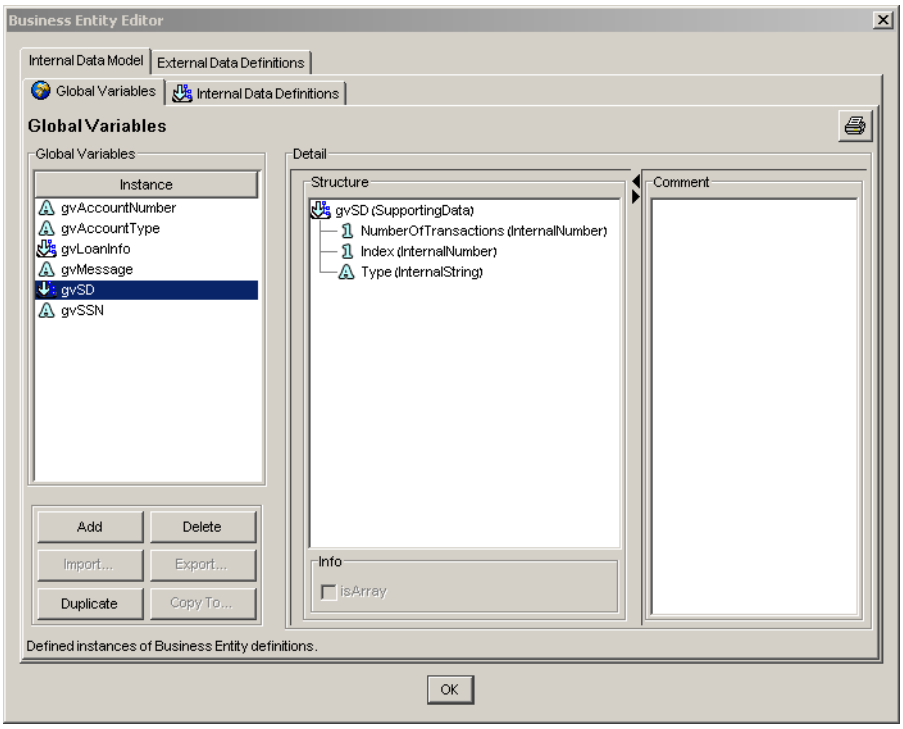


Figure 54. Business Entity Editor Dialog Box > Global Variables Tab

The **Global Variables** tab provides the following options:

Option	Description
Add	Click to add a new Global Variable.
Delete	Click to delete a Global Variable selected from the list.
Duplicate	Click to duplicate the selected Global Variable. The duplicated Global Variable receives a new name.

Option	Description
Default Value	Here you can specify a default value for the Global Variable. The default value is assigned to the Global Variable in a method, in case no value is provided by the method. **Note:** The default value must conform to one of the following formats of the data type in question: 1) The input format(s) 2) The default format (if the data type has no input format). For example, a non-structured global variable whose type is InternalDate has no input formats, so its default value is validated using the InternalDate's default format. If the value does not conform to the relevant format, a warning message is displayed.

2 Click **Add**.

The **Choose a Type** dialog box opens (Figure 55).

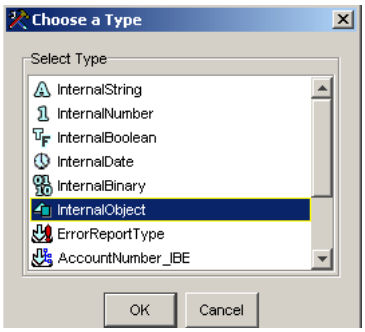


Figure 55. The Choose a Type Dialog Box

3 From the **Choose a Type** dialog box, choose the data type to be assigned to the Global Variable. Data types consist of webMethods JI primitive data types (e.g. `InternalString`) and defined Internal Business Entities. For more information on data types, see Chapter 7 - "Data Modeling" on page 201.

4 Click **OK**. The new Global Variable is displayed in the list.

5 Rename the Global Variable by right-clicking on it and selecting **Rename**.

6 Click **OK** to close the **Business Entity Editor**.

Global Actions

During regular interaction with the host application, some user activity can cause the application to move into a state that is unknown or unexpected by webMethods JI. Many host applications have an AID key that returns the user to a known place in the application. For example, an application may use the **PF3** key to always return the user to the main menu. If this is the case, a global action can be assigned to be used as a default mechanism that always returns the application from an unknown state to a known state.

> **Note:** An error message indication "To null after *** Action Destination" means the destination screen no longer exists. The global action should be deleted.

Using Global Actions with Table Templates

Global actions can also be used to determine the behavior of Table Templates. In order for Table Templates to repeat over multiple screens, they must have a "page down" action that allows them to get to the next screen. Global actions can be used to perform that "page down" action. This is useful when a table does not repeat over multiple screens during trail recording, but it does repeat over multiple screens at run-time. A global action using the AID key can be created and applied to advance to the next screen of tabular data. For more information about configuring Table Templates that repeat over multiple screens, see "Table Template Properties" on page 191.

Adding Global Actions to the Map

To add a global action to the map, proceed as follows:

1. Select the **Global Action** node in the Map Tree, right-click and select **Add Global Action**. The **Choose a Destination** dialog box is displayed, listing all the screens (Figure 56).

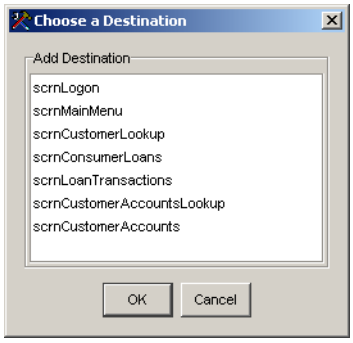


Figure 56. Choose a Destination Dialog Box

2 Select the screen to be used as the destination of this Global Action, and click **OK**.

A new action is added to the Global Actions node of the Map Tree. The new action can be renamed by right-clicking it, selecting **Rename** from the shortcut menu and entering a new name.

When the action is selected in the Tree, the action properties are displayed in the **Properties** view. For a description of this view, see "Modifying Actions" on page 165.

Screen Properties

To configure information related to a particular screen, select it in the Tree. The Screen properties are displayed in the Properties view (Figure 57).

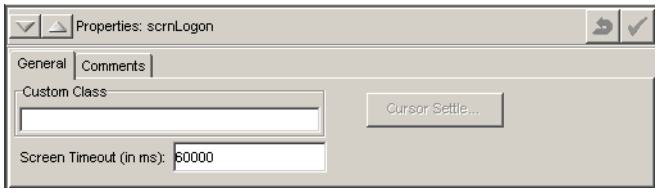


Figure 57. Screen Properties > General Tab

The **General** tab of the Screen Properties view offers the following options:

Option	Description
Custom Class	The custom class that should be used with the screen. For more information about custom classes, see "Custom Classes" on page 112.

Option	Description
Screen Timeout (in ms)	The amount of time, in milliseconds, that the service code waits for the screen to be recognized before it assumes that the screen traversal has failed.
Cursor Settle	This feature allows waiting for the cursor to settle in a particular position on the screen. Clicking this button opens the **Cursor Settle** dialog box (see "Cursor Settle" on page 155). **Note:** This feature is only available when using Character Mode.

Cursor Settle

The **Cursor Settle** dialog box (Figure 58) contains options for controlling the cursor settle position.

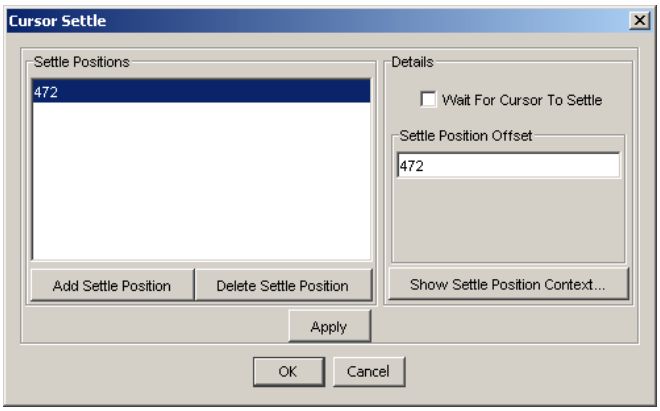


Figure 58. The Cursor Settle Dialog Box

The **Cursor Settle** dialog box offers the following options:

Option	Description
Settle Positions	This is a list of one or more offsets on the screen to which the cursor must be positioned before continuing. **Add Settle Position:** Click this button to add a new cursor settle position. **Delete Settle Position:** Select the position to delete and then click this button to delete it.
Wait for Cursor to Settle	Prevents the system from proceeding to anything else before the cursor settles into the preset position. **Note:** This box must be set to enable cursor position checking (based on the positions defined in the **Settle Positions** list). The default is cleared.
Settle Position Offset	Enter the offset in this field.
Show Settle Position Context	This button opens the **Settle Position Context** dialog box.

Settle Position Context

This screen allows the user to check if the settle position offset is correct by showing where the offset would be in relation to the characters on the selected snapshot.

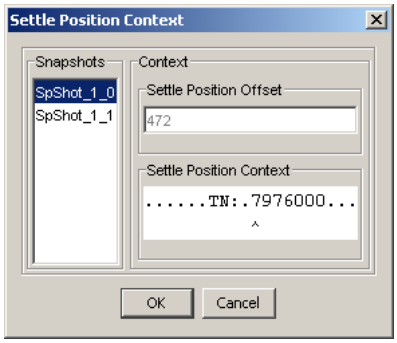


Figure 59. The Settle Position Context Dialog Box

The **Settle Position Context** dialog box offers the following options:

Option	Description
Snapshots	The list of snapshots in this screen.
Settle Position Offset	The position of the cursor offset.
Settle Position Context	This shows the 10 characters before and after the settle position. Use this to ensure that the settle position is in the right location. **Note:** Right-clicking on a position in the Presentation view displays a shortcut menu that shows the offset location of the mouse pointer. This is useful in determining the settle offset.

Note: Cursor Settle is not available for TN3270 or TN5250.

Customizing MapMaker's Screen Recognition

The rules for screen recognition and screen combination vary between Block Mode and Character Mode.

In Block Mode, MapMaker automatically attempts to combine screens. This is due to the additional screen information available in the TN3270 and TN5250 protocols. In contrast, for Character Mode, MapMaker keeps each snapshot in a separate screen until the user modifies the screen characteristics in order to direct MapMaker to combine the affected screens.

Block Mode

MapMaker analyzes the screen snapshots included in the trail repository to determine their general layout based on the formatted fields that make up the screens. When MapMaker determines that two snapshots are identical, it groups the snapshots together in the same screen in the map.

MapMaker uses the position of formatted fields, regardless of the content of the fields, to determine if screen snapshots are identical. As a result, some screens with identical fields may be incorrectly identified as identical, and screens that contain no formatted fields will be considered identical. In these cases, it is

necessary to tell MapMaker how to distinguish between screens that are different even though MapMaker interpreted them as identical. This is done using tags that identify a selected area on the screen that contains unique information. Unlike fields, which use only their position and size to determine a screen match, tags use alphanumeric characters contained in the selected area to determine the screen identity. Assign as many tags as required to uniquely identify a screen.

On the other hand, MapMaker may identify two snapshots of the same screen as separate screens, due to a difference in field layout. To ensure that both screens are identified as the same screen, you must disable all fields and assign a tag which is in common to both screens. Once the fields have been disabled, screen recognition is conducted only according to tags, and the two screens are united when the map is updated.

For more information see “Assigning a Tag” on page 159.

Character Mode

In Character Mode, there are no fields, therefore, the operation of MapMaker is different than for TN3270/TN5250. Initially, each screen has one artificial field that contains the whole screen. If this field is left enabled, MapMaker does a character by character comparison of screens and joins together identical snapshots under one screen. The artificial field can be disabled and scroll tags (see “Scroll Tag Configuration” on page 161) or regular tags (see “Assigning a Tag” on page 159) may be added to allow combining or splitting snapshots into screens.

Tags

Situations where tags are required:

- Unformatted screens always require tags so that MapMaker can distinguish between them.
- Often, the field layout of screens is identical but the information they contain is different (such as, when only certain actions may be desired as available depending on the data displayed in a field on the screen).

  “Tag” this data so that MapMaker differentiates between the screens based on the value displayed in the field. This forces MapMaker to limit the actions available based on the displayed text.
- Screens may serve similar purposes but might vary slightly depending on the fields in the screen. To instruct MapMaker to recognize these screens as the same, disable fields used in the screen recognition process. See “Customizing Fields Used for Screen Recognition” on page 163, for instructions.
- Scroll tags should be used in Character Mode to recognize screens that contain text that may vary in location (such as the UNIX logon/password and the UNIX Shell Prompt).

After assigning tags or disabling fields, update the map by selecting **Map** > **Update**.

Block Mode Field Attributes

Because of the way that MapMaker determines the size and location of fields for TN3270, and if the clients will not be logging into the host using the same extended attributes that were used when the map was recorded (i.e., TN3270 or TN3270E), create tags for each screen and disable all fields for screen recognition. This instructs MapMaker to ignore field formatting and recognize screens based on only the tags.

Character Mode Runtime Screen Recognition

Character Mode terminal emulations do not have fields like the TN3270 and TN5250 emulations. Therefore, at runtime, tags and Cursor Settle Position are the only way to recognize a screen.

Screens are matched in the following order:

1. Cursor Settle Position
2. Scroll Tags
3. Normal Tags

Assigning a Tag

When MapMaker has identified two or more screens as identical, they are grouped together under the Snapshots node of the screen in question.

To instruct MapMaker to distinguish between the identical screens, proceed as follows:

1. Use the mouse to click and drag a box around a unique area on the screen displayed in the Presentation view (e.g. the screen's serial number). The selected area is surrounded by a box. To deselect, click anywhere else in the Presentation view.
2. Right-click anywhere in the Presentation view and select **Add Tag** from the shortcut menu. A tag is added to the Tree view. Its properties are displayed in the Properties view (Figure 60).

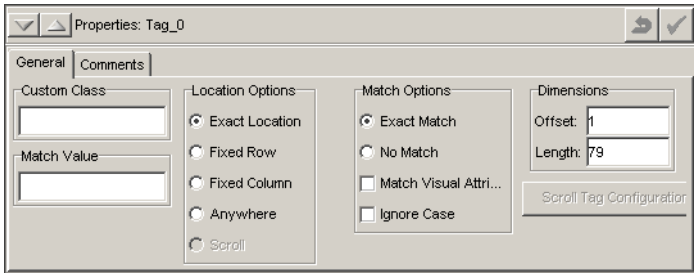


Figure 60. Tag Properties > General Tab

The **General** tab of the Tag Properties view offers the following options:

Option	Description
Custom Class	The custom class that should be used with this tag. For more information about custom classes, see "Custom Classes" on page 112
Match Value	The text string of the tag selected in the Presentation view.
Location Options	Defines how the tag "floats". Because the selected tag may be in different places depending on how the screen is formatted, this option allows you to create a "floating" tag that is used for screen identification. • **Exact Location:** The tag does not float. • **Fixed Row:** The tag floats but must appear in the same row. • **Fixed Column:** The tag floats but must appear in the same column. • **Anywhere:** The floating tag could be anywhere on the screen. • **Scroll:** This activates the **Scroll Tag Configuration** button and removes the text from the **Match Value** field. The tag configured with this button, matches the last N (raw) characters received from the host. A normal tag matches characters at a particular location in the Presentation view. **Note:** Scroll is only available for Character Mode

Option	Description
Match Options	• **Exact Match:** The tag must be on the screen, in the same position, and in the same case, in order for the screen to be recognized. • **No Match:** The tag cannot be present on the screen in order for the screen to be recognized. • **Match Visual Attributes**: Set to determine a screen matches this tag based on attributes such as color or highlighting. • **Ignore Case:** It is enough that the tag is spelled the same, and it does not need to have the same case for the screen to be recognized.
Dimensions	• **Offset:** The screen offset for the tag. • **Length:** The length of the tag.
Scroll Tag Configuration	This Character Mode option opens the **Scroll Tag Match Value** dialog box. Configuration is set here for one or multiple scroll tags. (See Scroll Tag Match Value below.) **Note:** Only one scroll tag is allowed per screen, although each scroll tag may have multiple matching strings.

Scroll Tag Configuration

Character Mode scroll tags are configured to match the last N raw characters (including escape sequences) received from the host. A typical use for the feature is to check for the “login:” prompt from a UNIX system. The tag strings are not checked until the host has stopped sending data for the configured amount of time (see Host Finished Timeout in “Map Properties” on page 147).

Non-alphanumeric characters may be required in scroll tags. Since scroll tags match raw characters received from the host, the following method is used to enter non-alphanumeric characters:

Entering Non-alphanumeric Scroll Tags	
Escape	\e
Control Characters	^<control char>

Entering Non-alphanumeric Scroll Tags

Hex number	\xhh
Octal number	\ooo
Backslash	\\

For example, to enter the VT100 escape sequence for positioning the cursor to row 2, column 10, enter:

```
\e[2;10H
```

Scroll Buffer

A Scroll Buffer is used to hold the last characters received from the host. Along with standard characters, escape sequences and control characters are also held.

An API method allows custom code to retrieve the scroll buffer. It is:

```
public Stringbuffer getScrollBuffer()
```

Scroll Tag Match Value

The **Scroll Tag Match Values** dialog box is opened using the **Scroll Tag Configuration** button in the Tag Properties view. This dialog box allows the configuration of scroll tag match strings.

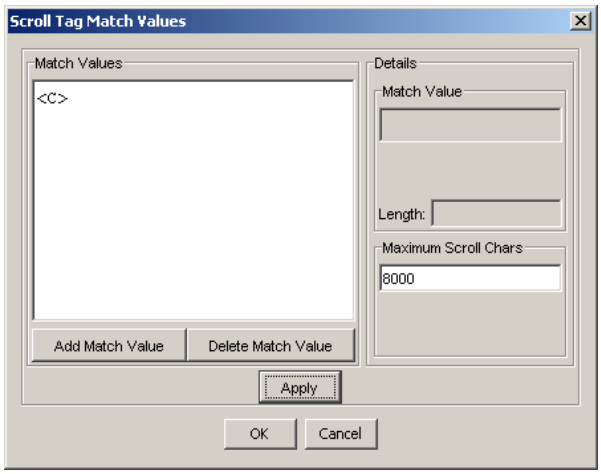


Figure 61. The Scroll Tag Match Values Dialog Box

The **Scroll Tag Match Values** dialog box provides the following options:

Option	Description
Match Values	A list of one or more scroll tag match values.
Match Value	Allows editing the selected scroll tag match value.
Maximum Scroll Chars	The maximum number of characters that can be received from the host, before it is assumed that the tag did not match. This is a "fail-safe" value to prevent MapMaker from getting stuck waiting for a tag if the host starts sending a continuous stream of data.

After identifying a tag for the screen, select **Map** > **Update Map** to update the map. Depending on the situation, screens may be added to, or removed from, the map.

Note: Assigning or unassigning a tag may cause screens to be removed from the map. As a result, any components tied to that screen, such as data templates, tags, or method steps, may need to be corrected or recreated.

Customizing Fields Used for Screen Recognition

Another method for customizing screen identification is to "disable" some of the fields used for matching. This is done to reduce the matchset, so that two or more screens with subtle differences are recognized as the same screen. A good example of such a case is a set of screens where the main purpose of the screens is the same and it is accomplished using a similar set of fields, but portions of the screens do not match. By default, all fields are used for screen matching, and are listed in the Tree view under the Fields node.

To disable all fields, right-click the Fields node and select **Disable All Fields** from the shortcut menu. Observe that an "X" is displayed next to the screen's fields in the tree, indicating that they are disabled (Figure 62).

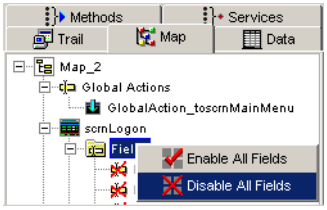


Figure 62. Disabled Fields

To re-enable all fields, right-click **Fields** and select **Enable All Fields** from the shortcut menu.

To disable a single field, select it and select **Disable Field** from the shortcut menu. Observe that an "X" is displayed next to the field in the tree, indicating that it is disabled. To re-enable a field, right-click it and select **Enable Field**.

Snapshots can be enabled or disabled in a similar way.

> **Note:** Enabling or disabling fields or snapshots may cause screens to be removed from the map. As a result, any components tied to that screen, such as data templates or method steps, may need to be corrected.

Using Floating Tags to Identify the Location of Floating Data Fields

Another purpose for tags is to tie them to data fields, which allows "floating data fields" to be created by anchoring the data field to a floating tag.

For example, the location of a data field may vary depending on other data that may be displayed on the host screen. Generally, the data field has a label that floats with the field. To create a "floating data field", map the field's label as a tag. Identify the tag as "floating" using the **Location Options** in the **General** tab of the Tag Properties view.

Then, when creating the data field, use the **Relative To Tag** option (in the **General** tab of the data field Properties view) to anchor the data field to a floating tag. For more information about creating data fields and setting their properties, see "Adding Data Fields to a Data Template" on page 186.

Actions

Block Mode Only

During Block Mode trail recording, MapMaker saves all the information typed into the fields and incorporates this information into Actions included with the map.

By default, MapMaker constructs an action using all the keystrokes entered into fields during trail recording as well as the AID key pressed to complete the operation.

Character Mode Only

During Character Mode trail recording, MapMaker saves all keystrokes sent to the host and incorporates this information into Actions included with the map. These keystrokes are separated into text (alphanumeric characters) and Action key types.

Both Modes

Often, rather than storing the data typed into fields during trail recording in the action, this data is replaced with a variable. For example, the text representing the user ID and password can be replaced with variables. These fields are no longer hard-coded in the webMethods JI service, but instead are passed as variables from the client application. This is accomplished by adding Global Variables to the map, and then assigning those variables to Action items in the map.

Modifying Actions

To modify an Action, select it in the Tree view. The action's settings are available for viewing and editing in the Properties view.

> **Note:** When you first display the Action's Properties view, the **General** tab is displayed by default. Once you navigate through additional tabs, the last tab selected is remembered and shown the next time the Action's Properties view is displayed. The selected tab is remembered until MapMaker is restarted.

Action General Tab

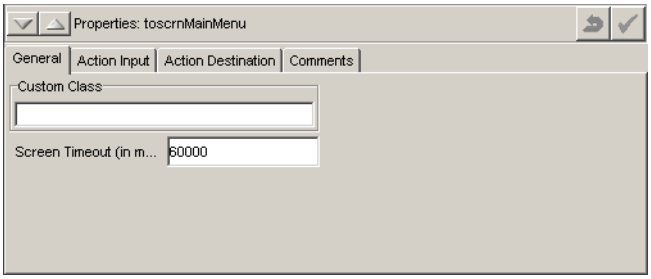


Figure 63. Action Properties > General Tab

The **General** tab of the Action Properties view offers the following options:

Option	Description
Custom Class	The custom class that should be used to extend this action. For more information about custom classes, see "Custom Classes" on page 112. **Note:** If a custom class is specified here, a stub may be generated (depending on the settings). This allows the action class to be overridden with custom code.
Action Timeout	The amount of time, in milliseconds, that the service code waits for the action to complete before it assumes that the action has failed.

Click the **Apply** button ( ) to apply any changes to the action properties.

Action Input Tab

The **Action Input** tab is used to identify the static text and variables as well as the AID key that serves as the input for the action.

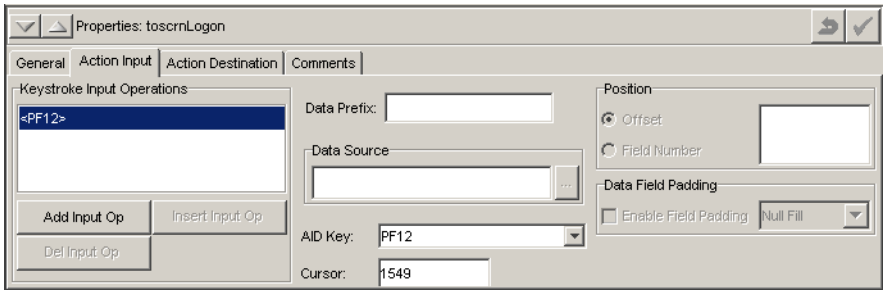


Figure 64. Action Properties View > Action Input Tab

The **Action Input** tab offers the following options:

Option	Description
Keystroke Input Operations	Lists the input operations associated with this action. The following items are listed: **TN3270/TN5250:** **<AidKey>:** Click on this entry to modify the AID key. **At N: <text>:** Shows the position and text that was typed into the screen. **Character Mode**: Alphanumeric characters are shown as "Text:<characters>". Action keys are shown as "Action Key:<keyname>".
Keystroke Input Operation (continued)	**All Communication Protocols:** Three buttons appear at the bottom of this list. These buttons control the following Input Operation options: • **Add Input Op**: Adds an input operation. • **Insert Input Op**: Adds an input operation before the currently selected operation. • **Delete Input Op**: Deletes an input operation. **When using Character Mode:** If either the **Add Input Op** or **Insert Input Op** button is clicked, a shortcut menu is shown. This allows the selection of either a **Text** or an **Action Key** operation. • **Text:** When selected, an empty text operation is inserted. To fill it in, select the operation, type the characters into the **Data Prefix** field and press the **Apply** button. • **Action Key:** When selected, an empty action key is inserted. To fill it in, select the operation, and select the appropriate key from the **Action Key** list.

Option	Description
Data Prefix	If data was entered into this screen during trail recording, it is listed in this field. If a variable is assigned to this action, any data in this field is prepended to the data input by the variable. If a Data Source is assigned to this action, this data prefix is prepended to the Data Source's value.
Data Source	Specify the name of the Data Source (a Global Variable or a Global Variable component) you wish to use as input. Specify the data source using one of the following techniques: • Type the data source's name. Or: • Click the ellipsis button to display the **Choose a Data Source** dialog box, and then select the appropriate data source from the list.
AID Key: Block Mode Or: Action Key: Character Mode	Represents the AID key (TN3270/TN5250) or Action key (Character Mode) that is pressed. A drop-down list allows you to select the desired key.
Cursor	Indicates the position of the cursor after the data was input.

Option	Description
Position	Represents the position on the legacy screen where the data will be input. This can be identified by the position (offset) or the field number. • **Offset:** Offset is determined using the following format: For an 80 column screen, the offset is (column # - 1) + 80 x (row # -1). For example, the first column of the second row is position 80: (1 - 1) + 80 x (2-1). • **Field Number:** This option should be used if the field "floats", that is, if the position of the field might change depending on what other data is displayed on the screen. The field number is based on the number of writable fields on the screen; if the field is the second writable field on the screen, the field number will be `2`. **Note:** This option is only available for Block Mode.
Data Field Padding	**Enable Field Padding**: This option controls padding of the input fields sent to the host. When set, the input field is padded with the selected fill character (**Null Fill** or **Space Fill**). • For Block Mode, this box is enabled with **Null Fill**. • For Character Mode, the default (and only) setting is **Space Fill**.

Click the **Apply** button ( ) to apply any changes to the action properties.

Action Destination Tab

The **Action Destination** tab provides options that specify how the service created by MapMaker determines that the action is complete and proceeds to the next instruction. If the action has multiple destinations, each destination has a separate set of properties. Select the destination screen in order to modify screen matching properties for each screen.

Note: These modifications cannot be made when using Character Mode.

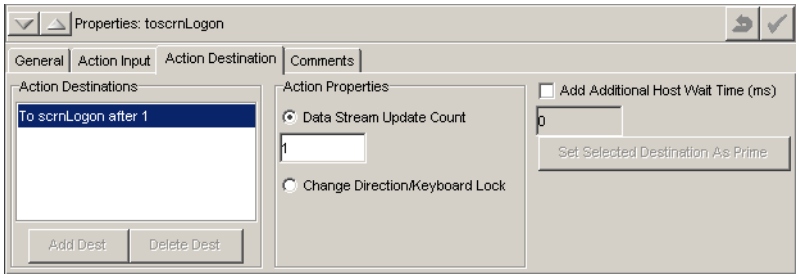


Figure 65. Action Properties View > Action Destination Tab

The **Action Destination** tab offers the following options:

Option	Description
Action Destinations	Lists the screen or possible screens that are navigated to after the action is complete. • **Add Dest**: Adds a destination to the list. After this button is clicked, select a destination from the provided dialog box. • **Delete Dest**: Deletes a destination from the list.
Data Stream Update Count	Identifies the number of data stream updates that are required from the host prior to the service attempting to verify that the current screen matches the action's destination screen. The default is the number of updates which occurred during trail recording. If this option is set to `0` (zero), the service checks after each data stream update to determine if the destination screen matches, waiting up to the timeout value for a destination match to occur. This is frequently needed for VM/ESA applications, which often send data in a varying number of bundles. **Note:** This option is only available for Block Mode.
Change Direction/ Keyboard Lock	When this option is selected, a change direction or keyboard lock is used to determine if the action is complete. **Note:** This option is only available for Block Mode.

Option	Description
Add Additional Host Wait Time (ms)	When this checkbox is set, you may specify how long the host should wait before the action has timed out.
Set Selected Destination as Prime	To identify one of the destination screens as the primary destination for this action, select the destination screen in the **Action Destinations** list and click the **Set Selected Destination As Prime** button. Because legacy applications sometimes have multiple routes to get to the same screen, this option allows the determination of which screen is the preferred route.

Click the **Apply** button ( ) to implement any changes to the action properties.

Comparing Screen Images

The Screen Compare feature enables you to compare one screen image with another in order to pinpoint the differences between screens in terms of fields, tags, attributes and text. In addition, the Screen Compare feature enables you to compare one screen image with all screens currently defined in a map.

Note: The screen size, in characters, must be the same in both screen images in order to perform the compare operation.

Screen images are classified into the following image types:

Screen	A legacy host screen that has been recorded and defined in MapMaker, as displayed in the **Map** tab.
Snapshot	An instance of a screen, as displayed in the **Trail** and **Map** tabs.
Live host	The current screen of the legacy host session to which MapMaker is connected.

When selecting screen images for comparison in the **Screen Compare** dialog box, the first image type selected determines the image types to which it can be compared, as shown in the following table:

Image type selected	Options available for comparison
Snapshot	• Snapshot • Live Host • Screen • All Screens
Live Host	• Snapshot • Screen • All Screens
Screen	Screen

This section describes:

- "Comparing One Screen Image with Another" on page 172.
- "Comparing One Screen Image with All Screens in a Map" on page 179.

Comparing One Screen Image with Another

One situation in which you need to compare one screen with another is when a map has been defined for a legacy application, and that legacy application is subsequently updated. The Screen Compare feature enables you to compare individual screens in a map with their updated counterparts in the legacy host, and reports each difference in fields, tags, attributes and text.

Another reason for comparing one screen image with another is to fine-tune the tagging and screen recognition process. For information about tagging and screen recognition, see "Customizing MapMaker's Screen Recognition" on page 157.

To compare one screen image with another:

1 Select **Tools** > **Compare**, or click the **Compare** button.
The **Screen Compare** dialog box is displayed:

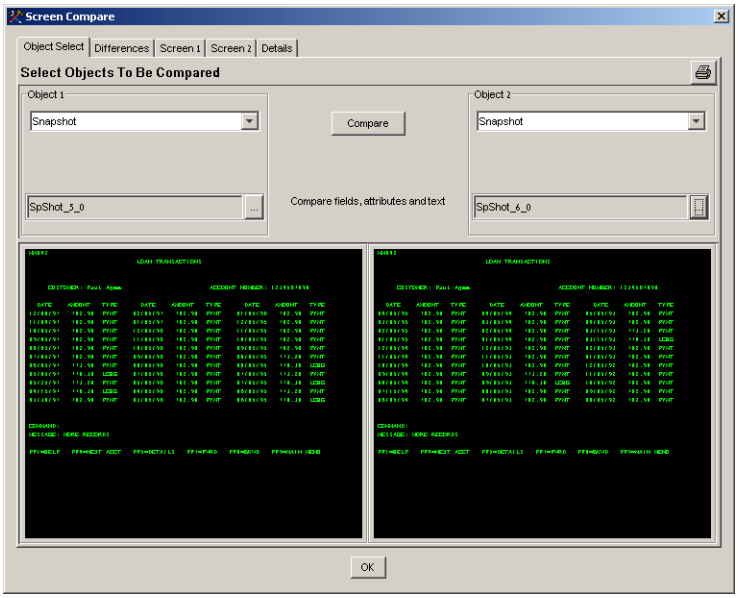


Figure 66. The Screen Compare Dialog Box

2 In the **Object 1** area, select the image type of the first screen in the comparison from the drop-down list.

3 Click the ellipsis button in the lower part of the **Object 1** area.

> **Note:** The image type selected in Step 2 determines the options displayed when clicking the ellipsis button. If you selected **Live Host**, the ellipsis button is disabled, the current legacy screen is displayed in the left of the **Comparison** pane, and you should proceed to Step 6.

If you selected **Snapshot** or **Screen**, the **Choose a Snapshot/Screen** dialog box is displayed, listing the available snapshot or screen names.

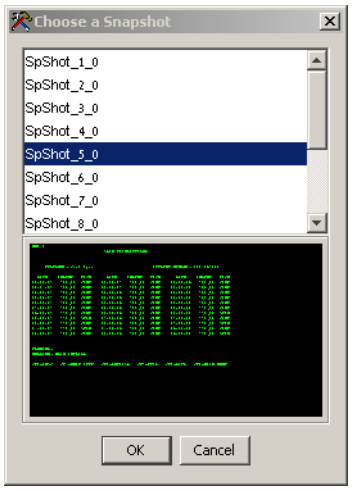


Figure 67. The Choose a Snapshot Dialog Box

4 Select the name of the snapshot or screen to be compared.

A preview of the selected image is displayed in the lower part of the **Choose a Snapshot/Screen** dialog box.

5 When you have selected the required snapshot or screen, click **OK**.

The selected image is displayed in the left side of the **Comparison** pane in the **Screen Compare** dialog box.

6 In the **Object 2** area, select the image type of the second screen in the comparison from the drop-down list.

7 Click the ellipsis button in the lower part of the **Object 2** area.

Note: The image type selected in Step 6 determines the options displayed when clicking the ellipsis button. If you selected **Live Host**, the ellipsis button is disabled, the current legacy screen is displayed in the right of the **Comparison** pane and you should proceed to Step 10. The **All Screens** option is described on page 179.

If you selected **Snapshot** or **Screen**, the **Choose a Snapshot/Screen** dialog box is displayed, listing the available snapshot or screen names, as shown in Figure 67.

8 Select the name of the snapshot or screen to be compared.

A preview of the selected image is displayed in the lower part of the **Choose a Snapshot/Screen** dialog box.

9 When you have selected the required snapshot or screen, click **OK**.

The selected image is displayed in the right of the **Comparison** pane in the **Screen Compare** dialog box.

10 Click **Compare**.

The **Differences** tab is displayed, containing the results of the compare operation:

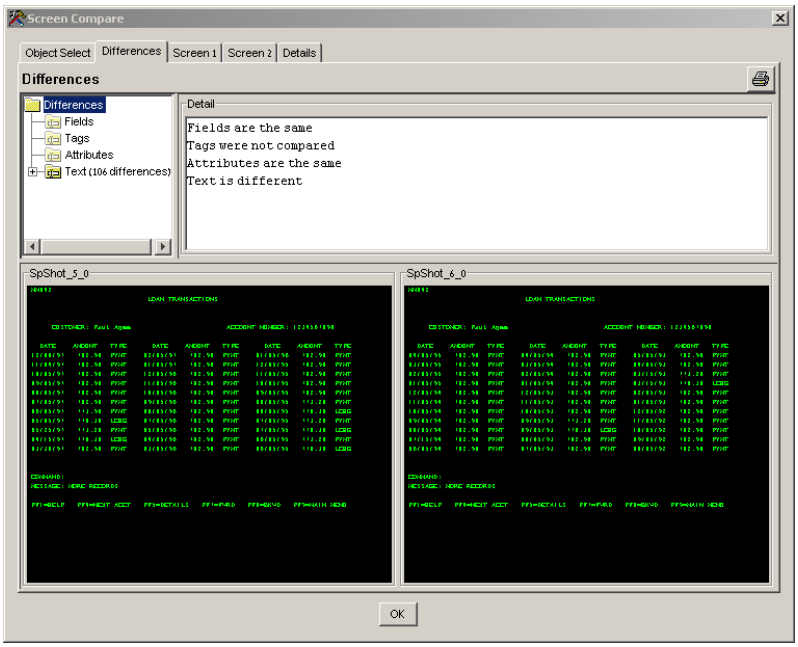


Figure 68. The Screen Compare Differences Tab

Viewing Results in The Differences Tab

The **Differences** tab is displayed when you compare one screen image with another, and contains a tree comprised of fields, tags, attributes and text differences. It also contains a **Detail** pane and a **Comparison** pane, containing textual and graphical descriptions of each difference.

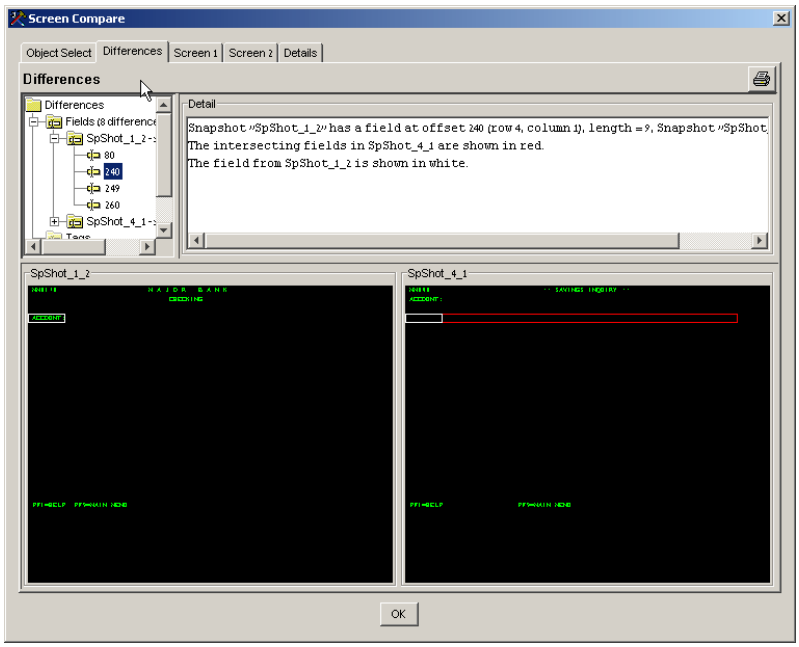


Figure 69. The Screen Compare Differences Tab Reporting a Field Difference Between Two Snapshots

The upper part of the **Differences** tab contains a Tree view on the left and a **Detail** pane on the right.

Tree view	Contains the **Differences** tree, where each node represents comparison types:
Fields	Contains the differences in field length and location.
Tags	Contains the text differences in tags.
Attributes	Contains the differences in field attributes, such as color, blink, underscore.
Text	Contains the differences in text.

The number of differences found for each difference type is displayed in parenthesis in the Tree view.

Selecting a node in the **Differences** tree outlines its corresponding differences in the **Comparison** pane, and displays a description of the outlines in the **Detail** pane.

Detail pane	Displays a summary of the differences, or a description of the various ways in which differences are outlined in the **Comparison** pane, according to the node selected in the **Differences** tree.

The lower part of the **Differences** tab contains the **Comparison** pane, showing the results of the two screen images that were compared. Elements of these screen images are outlined in red or white according to the node selected in the **Differences** tree. For more information about outlines, see "Screen Compare Outlines" on page 177.

Comparison Combinations

The combination of screen image types in a comparison determines which items are compared.

Snapshot to Snapshot comparisons compare:

- The fields in both snapshots.
- The attributes in both snapshots.
- The text in both snapshots. Characters with undisplayable values ('\0') are converted to a space character before comparison.

Snapshot to Screen comparisons compare:

- The fields in the screen to the fields in the snapshot. Disabled fields in the screen are ignored.
- The tags in the screen.

Screen to Screen comparisons compare:

- The fields in both screens. Disabled fields are ignored.
- The tags in both screens.

Note: A Live host screen image is treated as a Snapshot.

Screen Compare Outlines

The **Comparison** pane uses red and white outlines to accentuate each difference between the two screens in the comparison:

- A white outline surrounds an element that exists in one screen and its "would be" location in the other screen.
- A red outline surrounds an element that overlaps a white-outlined element.

Viewing Results in The Screen 1 and Screen 2 Tabs

The **Screen 1** and **Screen 2** tabs display larger versions of the screen images in the comparison, including any elements that are outlined according to the node selected in the **Differences** tab tree.

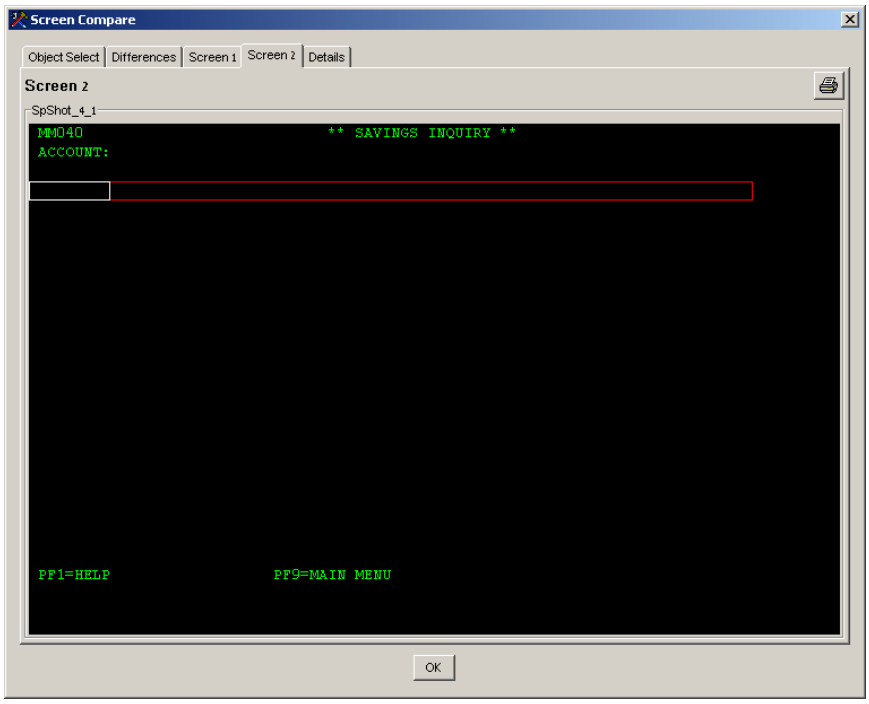


Figure 70. The Screen Compare Screen 2 Tab

Figure 70 shows an enlarged version of screen image 2, in which:

- An existing field from screen image 1, which does not occur in screen image 2, is outlined in white.
- A field that occurs in screen image 2 but overlaps the field from screen image 1 is outlined in red.

Viewing Results in The Details Tab

After comparing one screen image with another, select the **Details** tab to view a detailed textual description of the comparison results. This description lists the individual differences in fields, tags, attributes and text, as illustrated in Figure 71.

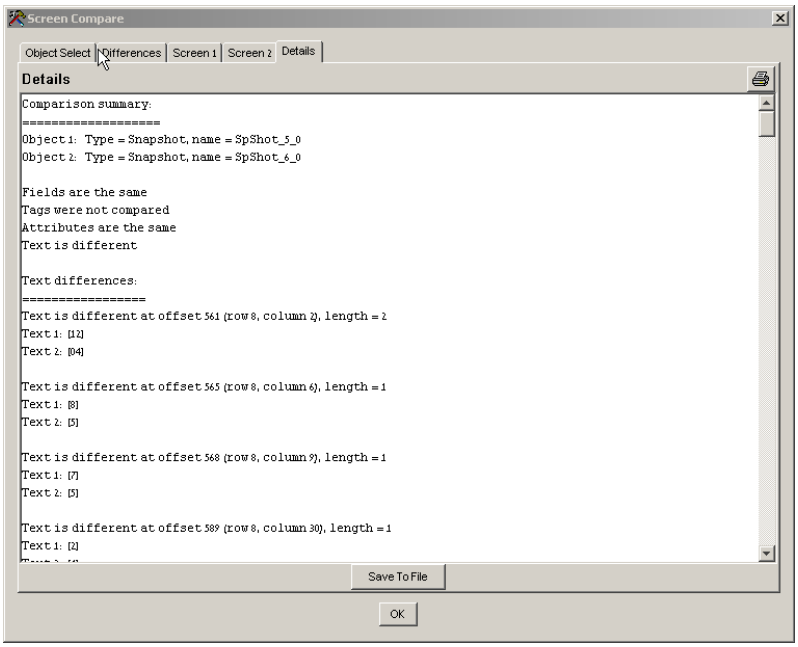


Figure 71. The Screen Compare Details Tab Listing Comparison Results

Comparing One Screen Image with All Screens in a Map

The Screen Compare feature enables you to compare one screen image with all screens currently defined in a map. This is useful for determining whether or not a legacy host screen is currently included in a map, or for checking MapMaker's name for a screen that you come across in the legacy host, for future reference.

To compare one screen image with all screens in a map:

1. Follow steps 1-5 in "Comparing One Screen Image with Another" on page 172.
2. In the **Object 2** area, select **All Screens** from the drop-down list.

   The ellipsis button is disabled and the right-hand **Comparison** pane goes blank.
3. Click **Compare**.

   The **Details** tab is displayed, stating whether or not the selected screen image matches a screen currently included in the map:

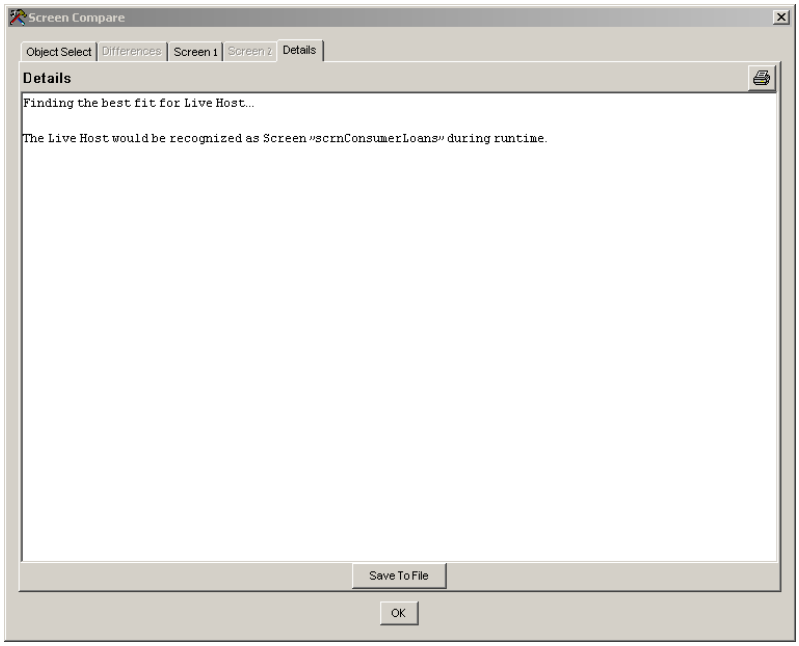


Figure 72. The Screen Compare Details Tab Stating that the Selected Snapshot has been Recognized

Chapter 6. Data

The **Data** tab enables you to define the data to be obtained from the legacy screens contained in the map. Data is retrieved from legacy screens by defining data templates or table templates, which contain data fields defined on a specific screen.

This section describes:

- "The Data Tab" on page 181
- "Data Templates" on page 183
- "Data Fields" on page 186
- "Table Templates" on page 190

The Data Tab

The **Data** tab of the Tree view (Figure 73) defines specific data fields of interest for each screen in the map. The selected data fields can contain one of the following:

- Input data - information provided by the client for input to the legacy screen, such as the user ID, password or bank account number.
- Output data - information required by the client for retrieval from the legacy screen, e.g. the balance of an account. Field properties, such as text color and style, can also be retrieved as output data.

These input and output data fields are organized in one of the following templates:

- Data templates - define individual fields.
- Table templates - define repeating or tabular data.

Note: Both data and table templates and the data fields they include may contain Custom Classes. For information on Custom Classes, see "Custom Classes" on page 112.

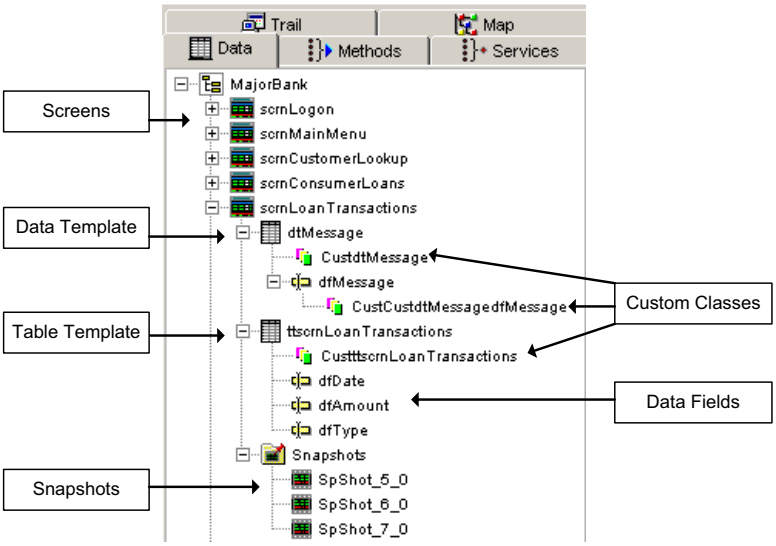


Figure 73. The Data Tab

The **Data** tab consists of the following options:

Component	Description
Screen	The name of the screen.
Data Template	Represents the data template for the screen. A data template is used to identify fields on the legacy screen that can be accessed programmatically.
Data Template, Table Template or Field Custom Class	The custom class that is generated to override this data template, table template or field. For more information about custom classes, see “Custom Classes” on page 112.
Data Field	Lists the data fields included in the data template or table template
Table Template	Represents a table template included in the screen. A table template is similar to a data template, except that it is used to group “repeating” fields (fields in a table on the legacy screen).

Component	Description
Snapshots	Lists all snapshots identified as belonging to this screen.

Data Templates

Data Templates map the host application screen elements to programmatically accessible objects. A data template is associated with a specific host screen, and serves as a general container for data fields that are defined after the data template is created. Data template fields are defined by selecting the field in the Presentation view and then adding the field to the data template.

Note: Multiple data templates may be defined for each screen.

Adding Data Templates

To add a data template, select the relevant screen in the Tree view, right-click and select **Add Data Template** from the shortcut menu.

A data template branch is added to the screen. When you select this data template in the Tree view, the data fields you define in the Presentation view are added to the selected data template.

Managing Data Templates

To add a custom class, rename or delete a data template, right-click the template in the Tree view and choose the appropriate command from the shortcut menu.

To configure the data template settings, select it in the Tree and edit its properties in the Properties view (Figure 74).

Data Template Properties > General Tab

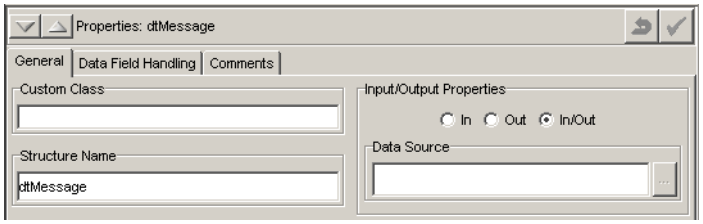


Figure 74. Data Template Properties > General Tab

The **General** tab of the data template Properties view provides the following options:

Option	Description
Custom Class	The custom class that is generated to override this data template. For more information about custom classes, see “Custom Classes” on page 112.
Structure Name	The key value inserted into the return map. The default structure name is the same as the template name. **Note:** Changing the structure name for the data template in the Properties view does not rename the data template in the Tree view, nor does it rename the external data type in the Business Entity Editor.
Input/Output Properties	Specify whether the data fields included in this data template are: • **In**: Data provided by the client. This option is available only if you have Global Variables defined. • **Out**: Data sent to the client • **In/Out**: Data that are both provided by the client and sent to the client (default).

Option	Description
Data Source	Identifies the Global Variable or Global Variable component that is updated by the value of this field. This option allows the retrieval of the data template values, and puts those values into a data source for use as an input value for another action. For example, a service might search for account numbers based on the customer's name and branch number. A subsequent screen looks up account information based upon the account number. This option allows the retrieval of the account number by creating a data template for the account number field. The account number is then input into the subsequent screen by tying the data template to a data source. That data source is then tied to the appropriate action that inputs the account number.

Data Template Properties > Data Field Handling Tab

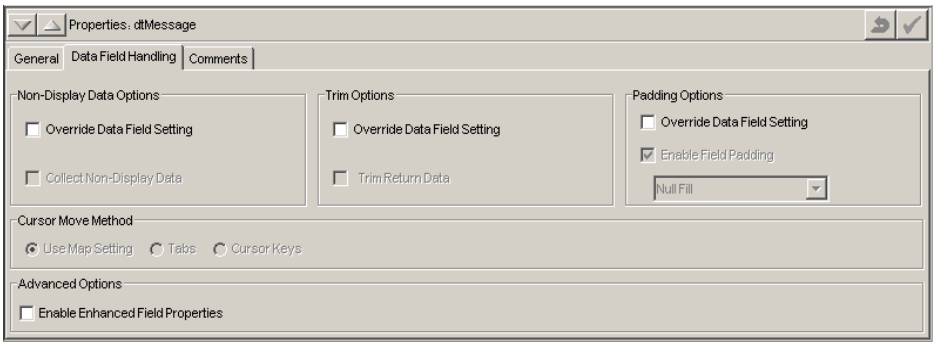


Figure 75. Data Template Properties > Data Field Handling Tab

The **Data Field Handling** tab of the data template Properties view (Figure 75) provides the following options

Option	Description
Non-Display Data Options	• **Override Data Field Setting**: • **Collect Non-Display Data**: When set, any non-display data collected is returned in readable form. When cleared, non-display fields are returned as spaces. Default is enabled.

Option	Description
Trim Options	• **Override Data Field Setting**: • **Trim Return Data**: When set, the leading and trailing white space is removed from the returned data. Default is disabled.
Padding Options	• **Override Data Field Setting**: • **Enable Field Padding**: This option controls padding of the input fields sent to the host. When set, the input field is padded with the selected fill character (**Null Fill** or **Space Fill**). - For Block Mode, this box is enabled with **Null Fill**. - For Character Mode, the default (and only) setting is **Space Fill**.
Cursor Move Method	Specify whether the cursor is moved using the **User Map Setting**, **Tabs** or **Cursor Keys** (character mode only).
Advanced Options	**Enable Enhanced Field Properties**: Enables the retrieval of properties from data fields defined in the legacy screen. These properties are **Name**, **Value**, **Length**, **Color** and **Style**. **Note:** This feature is enabled only when there are no data fields in the selected data template. To use this feature, set the **Enable Enhanced Field Properties** checkbox immediately after creating the data template.

Data Fields

Adding Data Fields to a Data Template

To add fields to a data template, proceed as follows:

1. Select the data template in the Tree view.
2. Select the field in the Presentation view, using one of the following methods:
   - Fields that were recognized by MapMaker as formatted fields can be selected automatically, by double-clicking on the field.

- Alternatively, position the cursor at the beginning of the field, click and drag the mouse to frame the field.

3 Right-click and select **Add DataField** from the shortcut menu (Figure 76).

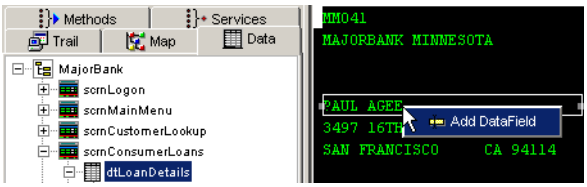


Figure 76. Adding Data Fields to Data Templates

The new data field is added to the data template branch that is selected in the Tree view.

Resizing Data Fields

By default, the length of a new data field is the maximum number of characters dedicated to it on the host screen. To resize the data field, proceed as follows:

1 Position the mouse cursor over the right or left side of the field.

2 When the cursor changes to a double arrow, click and drag to resize the field.

Managing Data Fields

To manage a data field, right-click it in the Tree view and choose one of the following options from the shortcut menu:

Option	Description
Add CustomClass	The Data Field's Properties view displays all Extendable Methods. Set the Methods that are to override this data field. For more information about custom classes, see "Custom Classes" on page 112.
Delete	Deletes the data field.
Rename	Renames the data field.

Data Field Properties

To configure the data field settings, select the data field in the Tree view. The data field properties are displayed in the Properties view.

Data Field Properties > General Tab

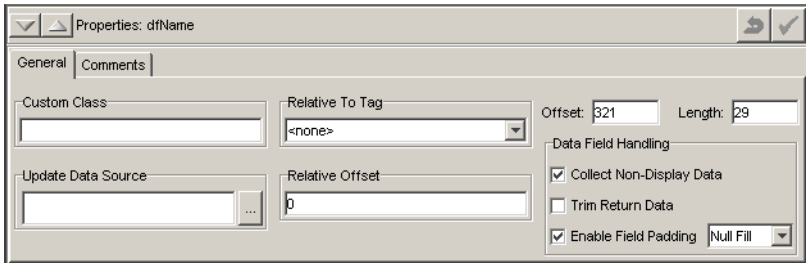


Figure 77. Data Field Properties > General Tab

The **General** tab of the data field Properties view offers the following options:

Option	Description
Custom Class	The custom class that is generated to override this data field. For more information about custom classes, see “Custom Classes” on page 112.
Update Data Source	Identifies the data source that is updated by the value of this field. This option allows the retrieval of the data field contents (value) and stores it into a Global Variable or Global Variable structure element for use by another component.
Relative To Tag	Identifies the tag this field is anchored to. Floating fields can be created by anchoring the field to a floating tag. For more information, see “Using Floating Tags to Identify the Location of Floating Fields” on page 189.
Relative Offset	For floating fields, this option represents the total number of characters that the floating field is offset from the tag to which it is anchored. This number should be calculated from the beginning of the tag to the beginning of the field.
Offset	The position of the field on the screen.

Option	Description
Length	The length, in characters, of the data field.
Data Field Handling	**Data Field Handling** features the following selections: • **Collect Non-Display Data**. When set, any non-display data collected is returned in readable form. When cleared, non-display fields are returned as spaces. Default is enabled. • **Trim Return Data**. When set, the leading and trailing white space is removed from the returned data. Default is disabled. • **Enable Field Padding**. This option controls padding of the input fields sent to the host. When set, the input field is padded with the selected fill character (**Null Fill** or **Space Fill**). - For Block Mode, this box is enabled with **Null Fill**. - For Character Mode, the default (and only) setting is **Space Fill**.

Using Floating Tags to Identify the Location of Floating Fields

Data fields can be anchored to “floating tags”. This allows the retrieval of data from fields whose position on the legacy screen might change depending on what other data is displayed on the screen.

For example, the location of a field might vary depending on the length of data in previous fields, or depending on the number of records in a table that is above the “floating” data field. Generally, the field has a label that floats with it.

To create a floating field:

1. Map the field’s label as a tag.
2. Identify the tag as floating, using the **Location Options** in the Tag Properties view. For more information, see “Assigning a Tag” on page 159.
3. After creating the data field, use the **Relative To Tag** option in the Data Field Properties view to anchor the field to a floating tag.

Table Templates

Table templates are primarily used to retrieve repeating tabular data that is contained on the legacy screen. Table templates are similar to data templates, except they do not encompass the entire screen.

After the table template is defined, individual repeating fields are mapped by selecting the field in the Presentation view and then adding the field to the table template.

Creating a Table Template

To create a table template, select the screen component in the Tree view, and position the mouse cursor at the corner of a table on the screen in the Presentation view. Click and drag the cursor to select the table. The table is enclosed in a red box (Figure 78).

After selecting the table, right-click within the red box and select **Add Table Template**.

> **Note:** A template has to be drawn carefully, if multiple columns of data are being retrieved with a template, it has to be properly positioned as it moves across the screen. The template's metrics are used to position subsequent positions of the template as it moves across the screen.

Figure 78. Creating a Table Template

After selecting a table template, the size of the selection box can be changed by positioning the mouse cursor over the resizing tools in the upper left or lower right corner of the red selection box. The cursor changes to a double arrow. Click and drag to resize the field.

The table template's name can be changed by right-clicking on the table template name in the Tree view, selecting **Rename** from the shortcut menu, and entering the new name. The name can also be changed in the Table Template Properties view.

Table Template Properties

To set the properties of the table template, select it in the Tree view and configure its settings in the Properties view.

Table Template Properties > General Tab

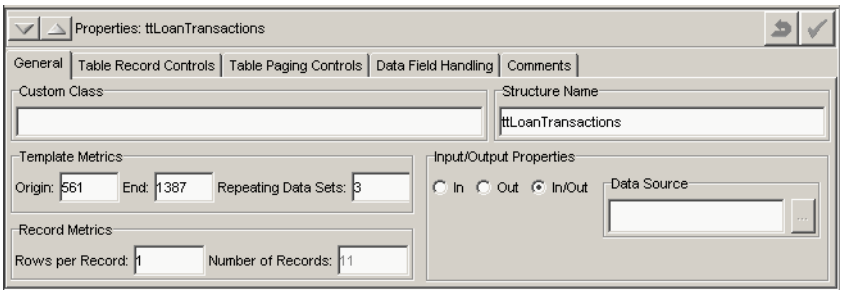


Figure 79. Table Template Properties > General Tab

The **General** tab of the Table Template Properties view (Figure 79) provides the following options:

Option	Description
Custom Class	The name of the custom class that should be used with this table template. For more information about custom classes, see "Custom Classes" on page 112.
Structure Name	The name that is inserted into the return message to the client. The default is the name of the data template.

Option	Description
Template Metrics	Describes the overall dimensions of the table template. The following items are identified: • **Origin**: The position (offset) of the beginning of the table. • **End**: The position (offset) of the end of the table. • **Repeating Data Sets**: Indicates the number of times that the table template repeats across the screen. Many tables on legacy screens repeat in multiple columns across the screen. For example, the table shown on page 190 has three columns: Date, Amount, and Type. The records for this table repeat across the screen in addition to repeating down the screen. Rather than defining a new table template for the next set of records, this option can be used to repeat the same table template across the screen.
Record Metrics	Identifies information about the records in the table. • **Rows per Record:** The number of rows used to display each record in the table. This value should be changed to identify the number of rows included in each record. • **Number of Records:** The total number of rows included in the column. The number is automatically calculated and cannot be changed.
Input/Output Properties	Identifies whether the table information can be input or output (or both). • **In:** These templates allow data to be written to the screen from a Global Variable. • **Out**: This is the default setting. These templates are "read-only". Data can only be output from the template. • **In/Out**: These templates can receive and output data. (Default) **Note:** If there is a Global Variable defined for the template, right-click on the template. The Properties view contains the above choices. For more information on Global Variables, see "Global Variables" on page 150.

Option	Description
Data Source	This property contains the Global Variable that should be used with this template. For more information on Global Variables, see “Global Variables” on page 150.

Table Template Properties > Table Record Controls Tab

The **Table Record Controls** tab has options to specify search criteria for a table or how to identify the end of data based on record content.

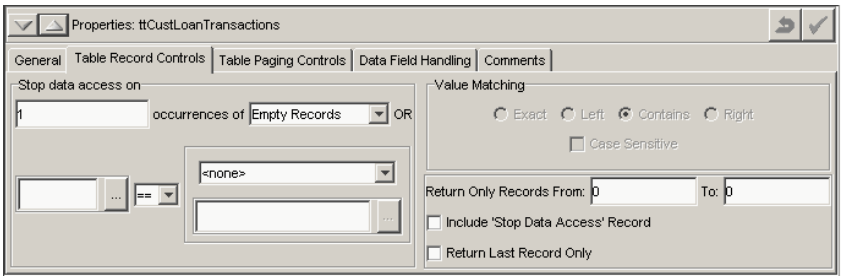


Figure 80. The Table Record Control Tab

The **Table Record Controls** tab (Figure 80) provides the following options:

Option	Description

Stop data access on	Determines how webMethods JI should stop retrieving records. Configure this service to stop in one of the following ways:
	• Stop after one or more occurrences of either **Empty Records** or **Repeated Records**,
	Or
	• Stop when two data sources are matched:
	  • Specify a data source by typing its name or clicking ... to select it from the **Choose a Data Source** dialog box.
	  • Choose between **= =** (equal to) and **! =** (not equal to) as the match criteria in the center box.
	  • Select the type of element the data source is to be matched against from the drop-down list. Choose between Global Variable, Internal String, Internal Number and Internal Boolean.
	  • Specify the selected element's name by typing it or by clicking ... to select it from the **Choose a Data Source** dialog box.

Option	Description
Value Matching	The value of the selected Data Source should match the text entered in the text box. The match between the specified Data Sources should be one of the following: • **Exact**: The text must match every character in the Data Source. • **Left**: The text must match the first (left-most) characters in the Data Source. • **Contains**: The Data Source must contain the text, but the • text can appear anywhere within the Data Source. • **Right**: The text must match the last (right-most) characters in the Data Source. • **Case Sensitive**: The case of the characters in the selected Data Source must match the case of the text in the Data Source Value.
Return Only Records From	Enter the range of record numbers to be included in the table.
Include 'Stop Data Access' Record	When set, causes the Stop record to also be collected. The Stop record is the record that matches the "Stop data access on" condition.
Return Last Record Only	When set, causes only the last record fetched to be returned as part of the table data. Default value is cleared. When searching for a single specific record, both this option and **Include 'Stop Data Access' Record** should be set.

Table Template Properties > Table Paging Controls Tab

The **Table Paging Controls** tab has options to determine the behavior of tables that repeat over multiple screens on the legacy application:

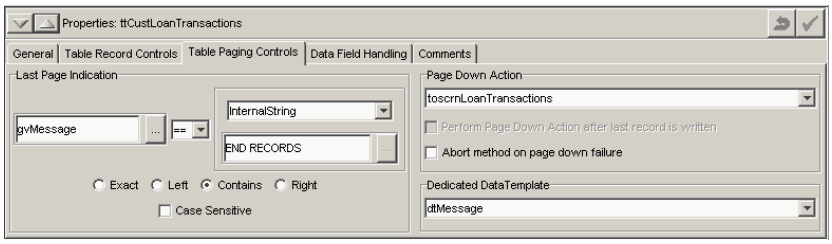


Figure 81. The Table Paging Controls Tab

The **Table Paging Controls** tab provides the following options:

Option	Description
Last Page Indication	If the specified conditions are met, the template considers this the last page of the table, and no further "page down" actions are performed. Set the last page indication as follows:T **1** Specify a Global Variable by typing its name, or clicking ... to select the Global Variable from the **Choose a Data Source** dialog box. **2** Choose between `==` (equal to) and `!=` (not equal to) as the match criteria in the center box. **3** Select the type of element the data source is to be matched against from the drop-down list. Choose between Global Variable, Internal String, Internal Number and Internal Boolean. **4** Specify the selected element's name by typing it or by clicking ... to select it from the **Choose a Data Source** dialog box.

Option	Description
Last Page Indication (continued)	The match between the specified Data Sources should be one of the following: • **Exact**: The text must match every character in the Data Source. • **Left**: The text must match the first (left-most) characters in the Data Source. • **Contains**: The Data Source must contain the text, but the text can appear anywhere within the Data Source. • **Right**: The text must match the last (right-most) characters in the Data Source. • **Case Sensitive**: The case of the characters in the selected Data Source must match the case of the text in the Data Source Value.
Page Down Action	Determines whether the service should "scroll" down to the next page of records. • If this is the only screen that contains records, select **<none>** from the drop-down list. • If there are additional screens that contain records, select the appropriate action for moving to these screens. For more information on how to control scrolling, see "Assigning Global Variables to Control Scrolling" on page 199.
Perform Page Down Action after last record is written	This selection allows the user to specify on a template write, whether a page down action should be performed after the last record has been written. This applies only to input or input/output tables, and only affects the action after the last record has been written.
Abort method on page down failure	Set to terminate the method in case the page down operation fails.
Dedicated Data Template	Specifies the data template that contains the data field for fetching the last page indicator on the screen.

Table Template Properties > Data Field Handling Tab

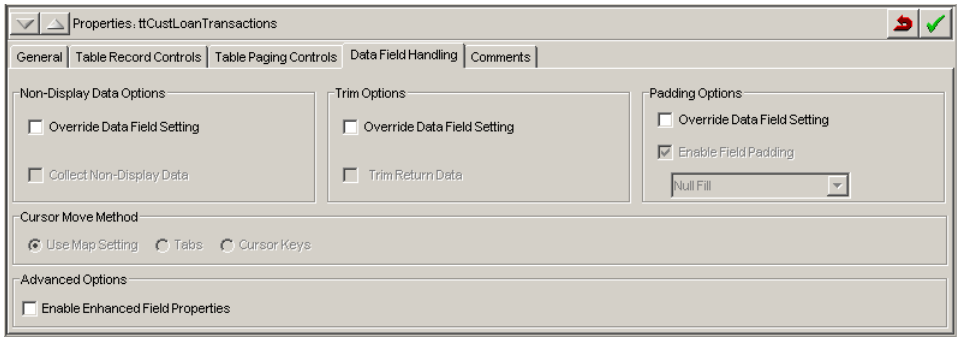


Figure 82. Table Template Properties > Data Field Handling Tab

The **Data Field Handling** tab provides the following options:

Option	Description
Non-Display Data Options	• **Override Data Field Setting**: • **Collect Non-Display Data**: When set, any non-display data collected is returned in readable form. When cleared, non-display fields are returned as spaces. Default is enabled.
Trim Options	• **Override Data Field Setting**: • **Trim Return Data**: When set, the leading and trailing white space is removed from the returned data. Default is disabled.
Padding Options	• **Override Data Field Setting**: • **Enable Field Padding**: This option controls padding of the input fields sent to the host. When set, the input field is padded with the selected fill character (**Null Fill** or **Space Fill**). - For Block Mode, this box is enabled with **Null Fill**. - For Character Mode, the default (and only) setting is **Space Fill**.
Cursor Move Method	Specify whether the cursor should move by using the **Map Setting, Tabs** or **Cursor Keys**.

Option	Description
Advanced Options	**Enable Enhanced Field Properties**: Retrieves the properties of data fields defined in the legacy screen. These properties are **Name**, **Value**, **Length**, **Color** and **Style**. **Note:** This feature is enabled only when there are no data fields in the selected table template. To use this feature, set the **Enable Enhanced Field Properties** checkbox immediately after creating the table template.

Assigning Global Variables to Control Scrolling

Host applications often contain multiple pages of data in table format, where the screen layout and title are the same for successive pages. There may be a message in a message status area that indicates there are additional records or that the screen is the last page of records. Alternatively, the message may be contained in one of the fields in the table template.

Once a table template has been added, it must be decided how scrolling through the pages is controlled within the map. When the message area contains a message denoting subsequent records of data, use a Global Variable to control scrolling.

Depending on the behavior of the legacy application, the message may be contained in one of the fields in the table template, in which case no additional data template is required because the value of the field on the table template can be used to populate the Global Variable created below.

1. The first step is to create a data template that captures the data contained in the message area. Click on the **Data** tab, then select the screen containing the table template. Right-click and select **Add Data Template** from the shortcut menu.
2. Select the new data template and double-click on the message indicating that the table continues to the next screen. Right-click and select **Add Data Field** from the shortcut menu.
3. The next step is to create a Global Variable. This variable extracts the data from the field for the message that indicates whether the table data is complete. The value of the variable is then used to determine the scrolling behavior of the table template.

   To create the variable, click the **Edit Business Entities** button ( ) to open the Business Entity Editor. Click on the **Internal Data Definitions** tab and select the **Global Variables** tab. In the **Global Variables** tab, click **Add**, select **InternalObject** as the data type for the Global Variable, and click **OK**. Select the

new variable, right-click and select **Rename**. Enter a new name for the Global Variable. Click **OK** in the Business Entity Editor to exit.

4 Next, we need to assign the new data field to the new Global Variable, so that the variable is populated with the data from the new data field.
Select the data field, and in the Properties view, specify the Global Variable to update in the **Update Global Variable** property by either typing the name of the Global Variable or by clicking the ellipsis button (...).

5 Next, configure the behavior of the table template to recognize the value of the new variable, so that when this variable contains the appropriate message, the table template will understands that it has to page down to extract more data from subsequent pages of information.

For instructions on providing a last page indication, see "Table Template Properties > Table Paging Controls Tab" on page 196.

Adding Fields to Table Templates

Fields are added to table templates by selecting the table template component in the Tree view and then selecting the field in the Presentation view. Fields that were recognized by MapMaker as formatted fields can be selected automatically, by double-clicking on the field. Fields can also be selected by positioning the mouse cursor at the beginning of the field, clicking, and dragging the mouse to select the field. After the field is selected, right-click and select **Add Data Field** from the shortcut menu.

After selecting a field using either of these methods, the size of the selection box can be changed as follows:

1 Position the mouse cursor over the right or left side of the field.

2 When the cursor changes to a double arrow, click and drag to resize the field.

After creating a data field, the field's name can be changed by right-clicking on the field name in the Tree view, selecting **Rename** from the shortcut menu, and entering the new name. To configure the field's settings, select it in the Tree. The field properties are then displayed in the Properties view (see "Data Field Properties > General Tab" on page 188).

Only the fields in the first logical record need to be defined. If a logical record spans more than one row then fields across all rows of the logical record should be defined if needed.

Chapter 7. Data Modeling

This chapter describes the following webMethods JI data modeling features and capabilities:

- Configuring the method input/output using a user-defined data structure that is required for a specific client (for example, defining the method input using XML data when working with a JClient3 or XML Gateway client).
- Supporting a variety of conventional data types, rather than only a String data type. The supported data types are String, Number, Boolean, Date, Binary and Object.
- Defining new data types. These may be structures that consist of other data types, either conventional or newly-defined. Moreover, a data type can be defined as an array of other data type objects.

  This is accomplished using MapMaker's **Business Entity Editor**.
- Defining Global Variables of various types, including newly defined data types. A Global Variable may, therefore, also be a structure consisting of various objects. A Global Variable of a certain data type allocates memory for the data that is directed through that specific data type.

  Global Variables are defined in MapMaker's **Business Entity Editor**.
- Exporting and importing data structures to and from external sources, such as XSD files.

  This is performed in MapMaker's **Business Entity Editor**.
- Defining the relationship between one data type and another.

  This is accomplished using MapMaker's **Structure Relationship Editor**, which supports the use of XPath Syntax. Using XPath, you can further manipulate the relationship between two corresponding objects.
- Mapping data from one data type to another. By mapping data, you are directing the flow of data within the chronological flow of a method. Even though the relationship between different data types is already defined, it is necessary to "tell" your method to take the data that arrives through one data type and, using Variables, transport it to another data type. Data mapping is necessary before a method's first step and after the method's last step.
  Data Mapping is defined using MapMaker's **Data Mapping Editor**.

Data Modeling is made possible through the use of the following MapMaker interface elements:

- The **Business Entity Editor** - allows you to add, edit and delete data types and global variables.
- The **Structure Relationship Editor** - allows you to select two of the existing data structures, and define the relationships between their elements.

- The **Data Mapping Editor** - displays information that is specific to a particular link between two specific method steps. It allows you to:
  - Map data from a **Source** data type to **Variable** data types.
  - Map data from **Variable** data types to a **Target** data type (if the step is a consumer of data).
  - Map data from a **Source** variable to a **Target** variable.

Assumptions

This document assumes that you are familiar with the *MajorBank.map* file provided with the webMethods JI Tutorial. For the examples presented throughout this document, we have created a map that uses four screens from the *MajorBank* map. The map's graphic schema is shown in Figure 83.

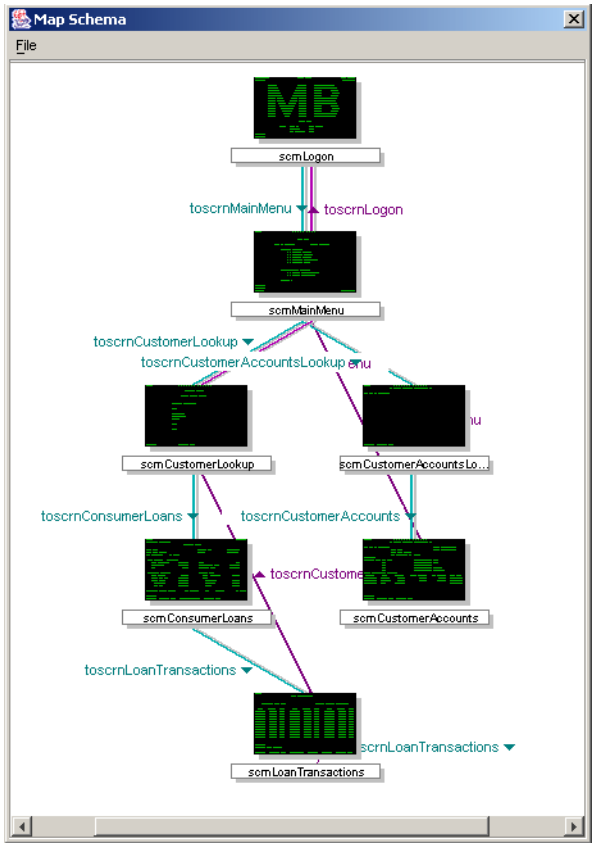


Figure 83. The MajorBank Host Application

The *MajorBank* map uses the following screens:

- **scrnLogon**: Used for logging into the host application.
- **scrnMainMenu**: The host application's main menu, allowing you to go to any of its screens.

- **scrnCustomerLookup**: Used to provide the account number you wish to query. Providing a correct account number takes you to the **Consumer Loans** screen (**scrnConsumerLoans**).
- **scrnConsumerLoans**: Displays information on the account whose number you entered in the previous screen (**scrnCustomerLookup**).
- **scrnLoanTransactions**: Displays date, amount and type information on each loan transaction performed by the owner of this account.
- **scrnCustomerAccountsLookup**: Used to provide the social security number you wish to query. Providing a correct social security number takes you to the **Customer Accounts** screen (**scrnCustomerAccounts**).
- **scrnCustomerAccounts**: Displays retail customer information for the given social security number.

Data Typing

A data type, also known as a Business Entity, is essentially a data structure used to define how data is transferred through the runtime. A data type may represent a single data object, or a structure consisting of various data objects. During runtime, every "crossroad" the data goes through is represented by data types.

There are two kinds of data types:

- **Internal Business Entities** (IBEs)—deal with the data that are used internally within the webMethods JI service.
- **External Business Entities** (XBEs)—deal with the data that are sent to or received from an entity that is external to the webMethods JI service, such as a legacy screen, a client or an "external" method step.

XBEs and IBEs are linked as follows:

Data is transferred from an XBE to a corresponding IBE in the following cases:

- When data is retrieved from a legacy screen.
- When data is received from a client.
- When data is received from an "external" method step.

Data is transferred from an IBE to a corresponding XBE in the following cases:

- When data is written into a legacy screen.
- When data is sent to a client.
- When data is sent to an "external" method step.

The term “Data Typing” refers to the definition and maintenance of data types. In webMethods JI, data typing is performed using MapMaker’s Business Entity Editor (for information on the Business Entity Editor, see “The Business Entity Editor” on page 210).

Internal Business Entities

Internal Business Entities (IBEs) represent user-defined data structures, which are internal to webMethods JI. IBEs can be thought of as the common data object or working objects within webMethods JI. An external business entity (XBE) must always be converted (copied) into an IBE for method steps to work against; and likewise, an IBE must always be converted (copied) to an XBE when interacting with an external data source.

IBEs (also referred to as internal types) are represented in MapMaker by the internal structure icon: . Each IBE structure may include any of the following internal elements:

- InternalString
- InternalNumber
- InternalBoolean
- InternalDate
- InternalBinary
- InternalObject

In addition, an IBE structure may include other IBE structures.

Figure 84 shows an example IBE named **LoanInfo** as it appears in the **Internal Data Definitions** tab of MapMaker’s Business Entity Editor.

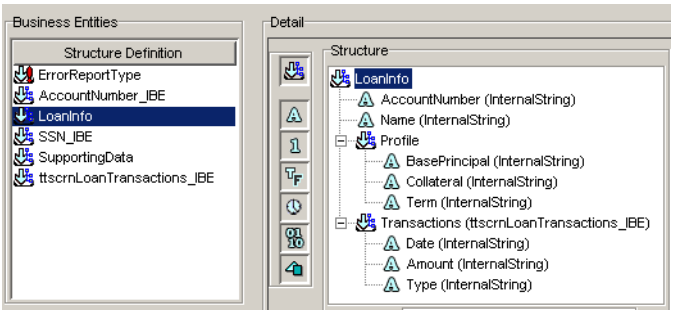


Figure 84. An IBE Data Structure

This IBE structure consists of two InternalString elements (**AccountNumber** and **Name**) and two InternalStructure elements (**Profile** and **Transactions**), which in turn contain their own InternalString elements.

ErrorReport IBE Type

In addition to the user-defined IBEs, there is a system-level IBE type known as ErrorReportType. This data type is always available on onFail links and is automatically populated when an error occurs within a method.

ErrorReportType is the only IBE that is automatically included in the **Internal Data Definitions** tab of MapMaker's Business Entity Editor. To access the ErrorReportType IBE from an onFail link, double click the onFail link to launch the Data Mapping Editor. The **ErrorReport** IBE is listed in the **Standard** tab's **Source** column.

Figure 85 illustrates the structure of an **ErrorReport** IBE.

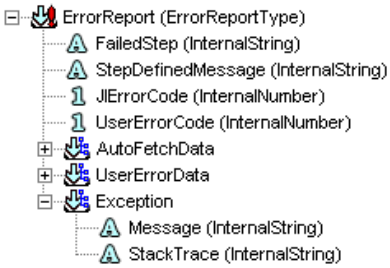


Figure 85. ErrorReport IBE Structure

The following table describes each element in the ErrorReportType structure.

Element Name	Data Type & Icon	Description
ErrorReport	ErrorReportType	The error report IBE structure.
FailedStep	InternalString	The name of the step that failed.
StepDefinedMessage	InternalString	A message defined by the step that failed, describing the error.
JIErrorCode	InternalNumber	Default error code that is predefined within JI.
UserErrorCode	InternalNumber	Allows you to create your own error code, by adding an integer value element to the ErrorReport IBE. This element can be used to return the error code to a client, an XML document, etc.

Element Name	Data Type & Icon	Description
AutoFetchData	InternalStructure	Any information that has already been fetched before the error occurred (e.g. all table data that a Fetch step managed to retrieve before failing).
UserErrorData	InternalStructure	A generic structure that you can populate with your own data (e.g. an error code, error string, etc.).
Exception	InternalStructure	Contains 2 strings: • A technical, java-level message based on the failure. • The Java stack trace for the exception.

Global Variables

Global Variables are used for storing and transferring data in a webMethods JI service. A Global Variable can be an instance of any the following types:

- Conventional data types — including String, Number, Boolean, Date, Binary and Object.
- IBE types — including all user-defined internal structures available, as well as the default ErrorReportType.

Figure 86 shows an example Global Variable named **gvLoanInfo** whose type is the **LoanInfo** IBE (shown in Figure 84), as it appears in the Business Entity Editor's **Global Variables** tab.

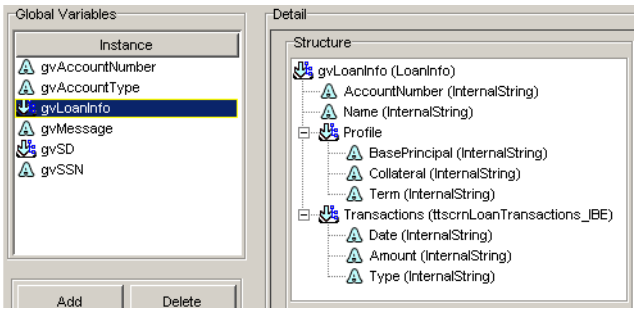


Figure 86. An IBE Type Global Variable

> **Note:** A Global Variable's structure is identical to the structure of its IBE type.

Variables can also be added on a method level, meaning that they apply only to the specific method in which they are defined. For more information, see "Method Variables" on page 282.

External Business Entities

An External Business Entity (XBE) is used to send or receive information to or from an external source, such as a client, a legacy screen or an "external" method step.

This section describes the three XBE types:

- "XML/XSD" on page 207
- "Map-List-Map" on page 208
- "Legacy Screen" on page 209

XML/XSD

An XSD Business Entity describes an XML document. The structure may contain elements and sub-elements.

An XSD Business Entity is used for:

- Receiving data from an XML client.
- Sending/receiving data from "external" steps.
- Sending data to an XML client.

Following is an example of an XSD Business Entity named **LoanInformation** as it appears in MapMaker's Business Entity Editor (Figure 87).

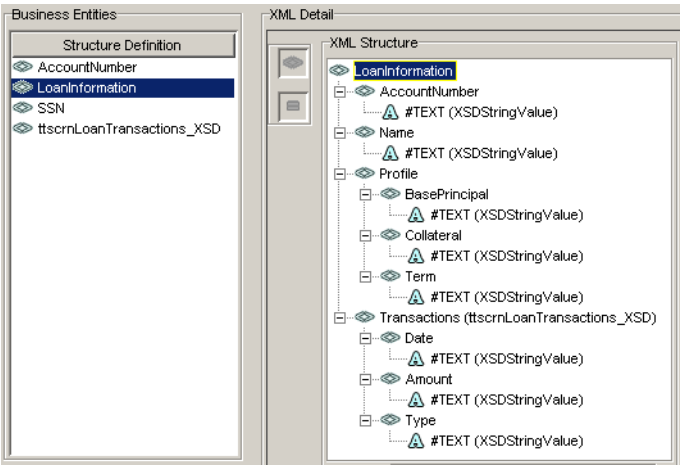


Figure 87. An Example XSD XBE Data Structure

This XSD Business Entity would be represented in XML as follows:

```
<LoanInformation>
    <AccountNumber></AccountNumber>
    <Name></Name>
    <Profile>
        <BasePrincipal></BasePrincipal>
        <Collateral></Collateral>
        <Term></Term>
    </Profile>
    <Transactions>
        <Date></Date>
        <Amount></Amount>
        <Type></Type>
    </Transactions>
</LoanInformation>"
```

The XSD structure in Figure 87 contains various elements, which are all marked by the symbol (including the root element, **LoanInformation**). Each of the sub-elements contains a text node whose type is XSDStringValue, marked by the symbol.

Map-List-Map

An MLM Business Entity describes a data structure used by webMethods JI's programmatic API.

An MLM Business Entity is used for:

- Receiving data from a Java client, a C client or a VB client

- Sending/receiving data to/from “external” steps
- Sending data to a Java client, a C client or a VB client

Figure 88 provides an example of an MLM Business Entity named **LoanInformation_MLM** as it appears in MapMaker’s Business Entity Editor:

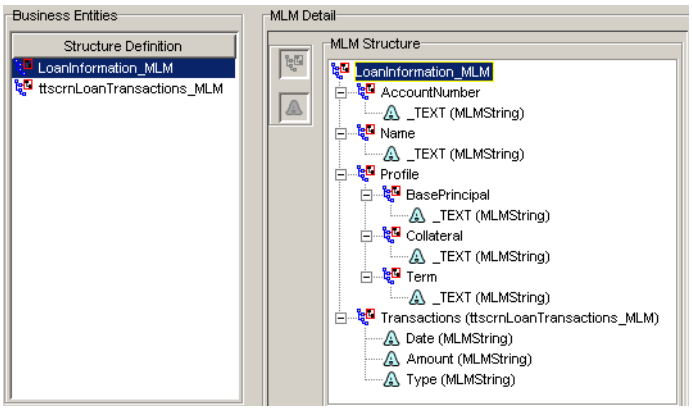


Figure 88. An MLM XBE Data Structure

The MLM above contains a root structure and various sub-structures that are all marked by the symbol. Each of these structures contains one or more data objects whose type is **MLMString**, marked by the symbol.

Legacy Screen

A Legacy Screen Business Entity describes a data template or a table template in a host screen. Data templates and table templates are created in MapMaker’s **Data** tab. These are automatically represented as Legacy Screen business entities in MapMaker’s Business Entity Editor.

Legacy Business Entities are for reference only, and cannot be edited in the Business Entity Editor. All changes must be made directly in the data template or table template itself, using the **Data** tab. These changes are automatically reflected in the corresponding Legacy Business Entities.

A Legacy Screen Business Entity is used for:

- Retrieving data from a host screen
- Writing data into a host screen

Enhanced Field Properties

When retrieving data from a host screen, a Legacy Screen Business Entity not only retrieves the value of a data field, but can also retrieve the properties used in the data field. These properties are:

- **Name**: The name of the field.

- **Length**: The length of the field, as defined by the MapMaker user.
- **Color**: The color of the field.
- **Style**: The style of the field. This property may contain several values, such as, Blink, Reverse Video, Underscore or Non Display.

> **Note:** This feature is enabled only when there are no data fields in the selected data/table template. To use this feature, set the **Enable Enhanced Field Properties** checkbox immediately after creating the data/table template. For more information, see "Data Template Properties > Data Field Handling Tab" on page 185 and "Table Template Properties > Data Field Handling Tab" on page 198.

Figure 89 provides an example of a Legacy Screen Business Entity named **dtLoanDetails** as it appears in MapMaker's Business Entity Editor:

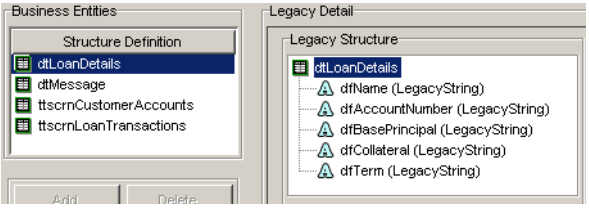


Figure 89. Legacy Screen XBE Data Structure

The structure above contains various data objects of LegacyString type. Each of these is marked by the symbol. The Legacy structure itself is marked by the symbol.

The Business Entity Editor

The Business Entity Editor consists of two upper tabs, each enabling different lower tabs:

- The **Internal Data Model** tab enables the following lower tabs:
  - **Internal Data Definitions** (see "The Internal Data Definitions Tab" on page 242)
  - **Global Variables** (see "The Global Variables Tab" on page 243).
- The **External Business Entities** tab enables the following lower tabs:
  - **XML/XSD** (see "The XML/XSD Tab" on page 244)
  - **MLM** (see "The MLM Tab" on page 246)
  - **Legacy Screen** (see "The Legacy Screen Tab" on page 247)

These tabs share the same structure, outlined in Figure 90.

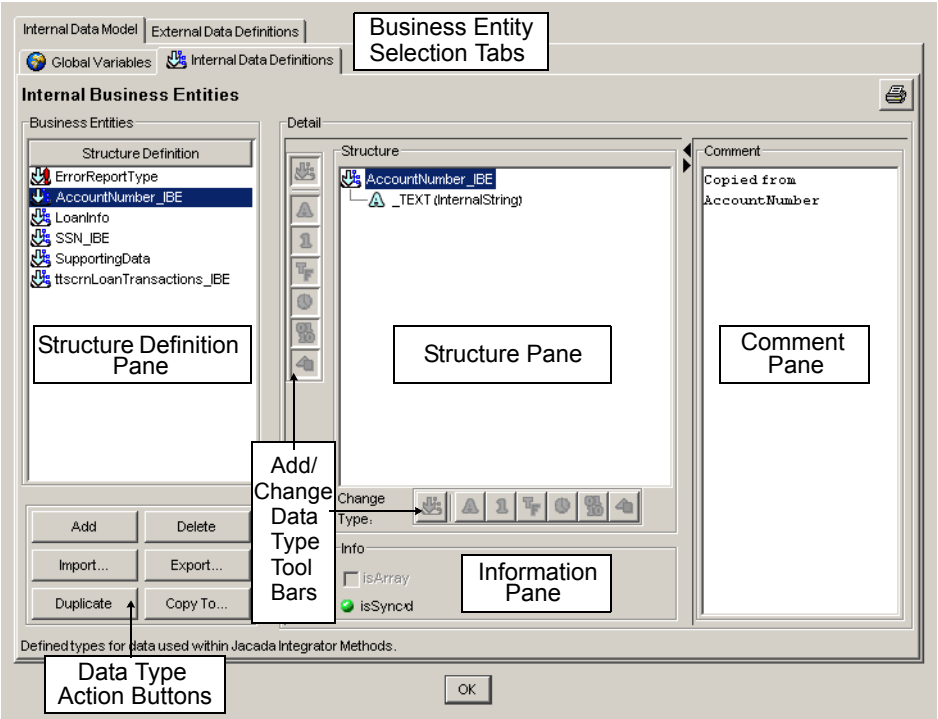


Figure 90. The Business Entity Editor

The Business Entity Editor provides the following options:

Option	Description
Business Entity selection tabs	Click to display the BE type of interest.
Structure Definition pane	Lists all public structure definitions of the selected BE type. These public structure definitions can be used as elements of other structures. Structure definitions are managed using the following tools: • The data type action buttons. • A shortcut menu, allowing you to either rename or delete the selected structure definition.

Option	Description
	Selecting a structure definition in the **Structure Definition** pane displays its hierarchical structure in the **Structure** pane, allowing you to manage its root and elements.
Structure pane	Shows the hierarchical structure of the public structure definition selected in the **Structure Definition** pane. This structure may include both public and private data types. The structures and their elements are managed using the following tools: • The add/change type tool bar buttons • The data type info properties • Shortcut menus (see "The Business Entity Editor Shortcut Menus" on page 215.)
Data Type add buttons (vertical tool bar) & change buttons (horizontal tool bar) • InternalStructure • InternalString • InternalNumber • InternalBoolean • InternalDate • InternalBinary • InternalObject	The **Details** pane contains a vertical toolbar and a horizontal toolbar used to manipulate the element selected in the **Structure** pane: • The vertical toolbar allows you - to add elements of various data - types to the selected element. • The horizontal tool bar allows you to - change a selected element from - one data type to another. **Note:** These tool bars are available in *the* **Internal Data Definitions**, **XML/XSD** *and* **Map-List-Map (MLM)** tabs only. The buttons available in each toolbar vary as a function of the data type tab you are displaying.

Option	Description
• MLMStructure • MLMString • XSDElement • XSDAttribute • XSDStringValue • XSDDecimalValue • XSDFloatValue • XSDDoubleValue • XSDBooleanValue • XSDDateTimeValue • XSDBase64Binary Value	The **Structure** pane supports the following drag and drop functionality: - You can drag new types from the tool bar to the appropriate location in the selected structure's hierarchy. - Within each of the structure's parent nodes, you can reorder the child elements by dragging them to their appropriate location.
Data Type action buttons	• **Add**: Adds a new BE or global variable. For more information, see "Adding Business Entities" on page 222. • **Delete**: Deletes the selected BE. For more information, see "Deleting Business Entities" on page 223. • **Import**: Imports a BE from an external JIX file. For more information, see "Importing Business Entities" on page 223.

Option	Description
	• **Export**: Exports the selected BE to an external JIX file. For more information, see "Exporting Business Entities" on page 226. • **Duplicate**: Duplicates the selected BE and assigns it a new name. For more information, see "Duplicating Business Entities" on page 227. • **Copy to**: Copies the selected BE from one data type to another. For more information, see "Copying a BE from One Data Type to Other Data Types" on page 228). • **Print** ( ): Launches the **Print** dialog box, allowing you to print all or part of the Business Entity Editor's current tab. For information on the **Print** dialog box, see "Printing the Display" on page 109.
Information pane	Provides the following information on the BE: • **isArray**: Indicates that the object in question is a repeating data type. For information on array data types, see "Indicating a Data Type is an Array" on page 232. • **isSync'd**: Indicates that the selected BE is synchronized with another BE. For information on synchronization, see "Synchronizing the Source BE with the Copied BE(s)" on page 228.

Option	Description
Comment	Space for any type of written comment. When the selected data type has been copied from another data type, this pane automatically indicates the name of the source data type.

The Business Entity Editor Shortcut Menus

The Business Entity Editor allows you to manage IBEs and XBEs using shortcut menus. These menus are available by right clicking the object of interest in either the **Structure Definition** pane or the **Structure** pane.

The **Structure** pane includes two different shortcut menus: one for the structure's root and another for the structure's elements (which can be either primitives or structures).

The available commands are context-sensitive and depend on the type of item selected in the list. For example:

- If the selected item is a source BE that has copies it is synchronized with, changes to the source BE propagate to its copies.
- If the selected item is synchronized with the source BE it was copied from, it cannot be edited and all menu items (besides **Rename**) are disabled (all changes must be made directly to the source BE).
- If the selected item is a Global Variable, most commands are not available, since Global Variables are only instances of other BEs and therefore they cannot be edited (all changes must be made directly to the source BE).

The Structure Definition Pane's Shortcut Menu

To either rename or delete a structure definition, right-click it in the **Structure Definition** pane and select the appropriate command from the shortcut menu.

The Structure Pane's Root Shortcut Menu

The shortcut menu available to the root of the structure is shown in Figure 91.

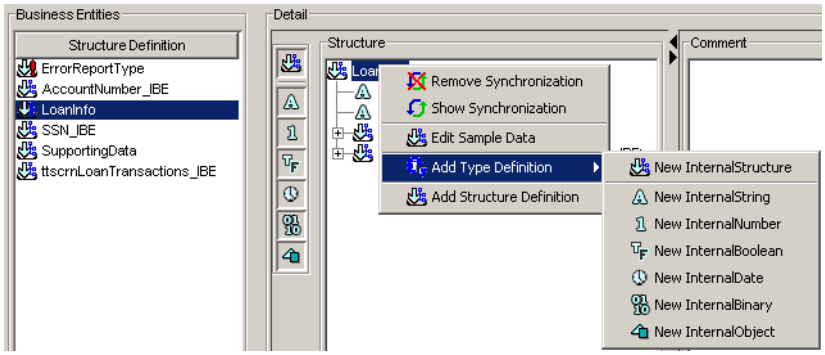


Figure 91. Business Entity Editor Root Shortcut Menu (Add Type Definition Options)

The root shortcut menu provides the following options:

Option	Description
Remove Synchronization	Breaks the synchronization between this structure and the BE(s) you select from the **Choose a Type** dialog box (see "To break the synchronization between BEs:" on page 231).
Show Synchronization	Launches the **Synchronized Business Entities** dialog box, which shows all BEs that are synchronized with the selected structure (see "To view all BEs that are synchronized with a BE of interest:" on page 229).
Edit Default Data	(**Global Variables** tab only) Launches the **TypeDataModel Viewer**, allowing you to insert a default value, which (unlike the **Sample Value**) *does* populate the Global Variable at runtime. For more information, see "Step 5: Viewing and Editing Method Data" on page 383.
Edit Sample Data	(All tabs besides **Global Variables**) To facilitate and automate the testing process, you may assign a sample value to any IBE or XBE type. This sample value populates the code in the appropriate places, saving you the need to insert it manually. **Note:** The sample value is not used at runtime and does not appear in the generated code.

Option	Description
	The **Default Value** or **Sample Value** must conform to one of the following formats of the data type in question: 1) The input format(s) 2) The default format (if the data type has no input format). For example, a non-structured global variable whose type is InternalDate has no input formats, so its 'Default Value' is validated using the InternalDate's default format. If the value does not conform to the relevant format, a warning message is displayed. **Note:** The Start and Stop steps' **General** tabs include a **Clear Global Variables** option, which resets global variables to the above default value.
Add Type Definition	Displays a selection of primitives that can be added to the selected structure. The available type definitions correspond to the horizontal, "Change Type" tool bar.
Add Structure Definition	Displays a selection of existing structures that can be added to the selected structure. The available structure definitions correspond to the ones listed in the **Structure Definition** pane (with the exception of the ErrorReportType IBE. see Figure 91). **Note:** If no other structures are defined, this menu item is empty.

Note: Add Type Definition and **Add Structure Definition** apply only to BEs you can add elements to, i.e. IBEs, XML/XSDs and MLMs. They do not apply to the following BEs:

- Legacy XBEs: elements are added to Legacy XBEs only through MapMaker's **Data** tab.
- BEs that cannot be edited. For example, when the BE is synchronized with its source, elements can only be added directly to the source.

The Structure Pane's Element Shortcut Menu

The shortcut menu available to the structure's elements is shown in Figure 92.

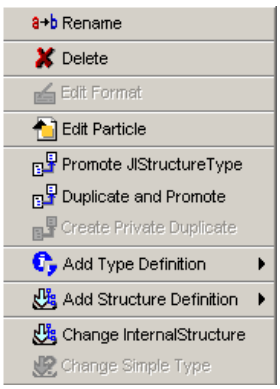


Figure 92. Business Entity Editor Element Shortcut Menu

The element shortcut menu provides the following options:

Option	Description
Rename	Allows you to enter a new name for the element (corresponds to clicking the **F2** key).
Delete	Removes the selected element from the structure.

Option	Description
Edit Format	Launches the **Edit Format** dialog box, where you can create different formats for the inputs and outputs of the selected data type. This option allows you to receive input in one format and return it as output in a different format (see "Formatting Data Types" on page 236). **Note:** This command is available only to IBE and XSD Number, Date and Boolean data types.
Edit Particle	Specify the minimum and maximum number of times this particle must appear in the XML structure for the structure to be valid. Default is 1 (see "Editing an XML/XSD Particle" on page 244).
Promote JIStructureType	Promotes the selected private structure to a public structure. The **Confirm Change** dialog box is displayed, requiring that you click **OK** to apply the promotion. As a result: • The element's name now shows its public type name in parentheses. For example, instead of `sub_0` the name appears as `sub_0_(sub_0)`. • A new public structure, with the same name as the promoted element, is added to the **Structure Definition** pane. **Note:** This command is not available if the selected object is already a public type, or if it is synchronized with a source BE.

Option	Description
Duplicate and Promote	This operation results in two separate instances of the selected element: the original private structure and a new public structure. The **Confirm Change** dialog box is displayed, requiring that you click **OK** to apply the duplication and promotion. As a result: • The element remains private in the context of the selected structure (as indicated by its name, which does not show a public type in parentheses). • An additional, public copy of the element is added to **Structure Definition** pane. **Note:** This command is not available if the selected object is already a public type, or if it is synchronized with a source BE.
Create Private Duplicate	This operation results in two separate instances of the selected element: the original public structure and a new private structure. The **Confirm Change** dialog box is displayed, requiring that you click **OK** to apply the duplication and demotion. As a result: • The element becomes private in the context of the selected structure (as indicated by its name, which no longer shows a public type in parentheses). • The original, public element remains available in the **Structure Definition** pane. **Note:** This command is not available if the selected object is already a private type, or if it is synchronized with a source BE.

Note: The following two properties, **Add Type Definition** and **Add Structure Definition**, apply only to BEs you can add elements to, i.e. IBEs, XML/XSDs and MLMs. They do not apply to the following BEs:

- Legacy XBEs: Elements are added to Legacy XBEs only through MapMaker's **Data** tab.
- BEs that cannot be edited. For example, when the BE is synchronized with its source, elements can only be added directly to the source.

Option	Description
Add Type Definition	Displays a selection of primitives that can be added to the selected structure. The available type definitions correspond to the horizontal, "Change Type" tool bar.
Add Structure Definition	Displays a selection of existing structures that can be added to the selected structure. The available structure definitions correspond to the ones listed in the **Structure Definition** pane (with the exception of the ErrorReportType IBE. see Figure 91). **Note:** If no structures other than the root structure are defined, this menu item is empty.
Change InternalStructure	Maintains the internal structure's name, while allowing you to change its type to one of the public types available in the **Structure Definition** list. The **Choose a Type** dialog box is displayed, allowing you to select the element's new public type. Clicking **OK** launches the **Confirm Change** dialog, requiring that you click **OK** to apply the type change. As a result, the internal structure's name now shows the selected public structure's name in parentheses.
Change Simple Type	Allows you to change the selected primitive's type to another primitive type (e.g. change an InternalString to an InternalNumber). The available simple type definitions correspond to the vertical, "Add Type" tool bar.

Managing Business Entities

This section describes:

- "Adding Business Entities" on page 222
- "Deleting Business Entities" on page 223
- "Importing Business Entities" on page 223
- "Exporting Business Entities" on page 226

- "Duplicating Business Entities" on page 227
- "Copying a BE from One Data Type to Other Data Types" on page 228
- "Indicating a Data Type is an Array" on page 232

Adding Business Entities

The Business Entity Editor allows you to add all types of BEs, except for Legacy XBEs that are managed exclusively in MapMaker's **Data** tab (Legacy XBEs are added as data templates or table templates, and are automatically displayed in the Business Entity Editor's **Legacy Screen** tab).

IBEs, XML/XSDs and MLMs are all added in the same way (see "To add a new IBE, XML/XSD or MLM data type:" on page 222). Global Variables are added using a different procedure, in which their data type is selected ("To add a new Global Variable:" on page 222).

To add a new IBE, XML/XSD or MLM data type:

1. Display the tab managing the type of BE you wish to add.
2. Click **Add**.

   A new data type is added to the **Structure Definition** list. The data type's default name consists of the data type's prefix (IBE, XML/XSD or MLM) followed by an underscore and a number ranging from 0 to n (e.g. `IBE_0`).
3. Rename the new data type in one of the following ways:
   - Double-click on it in the list.
   - Right-click on it and select **Rename**
   - Click on it and press the **F2** key.
4. Enter the new name, and then apply it by pressing **Enter** or selecting another item in the list.

To add a new Global Variable:

1. In the **Global Variables** tab, click **Add**.

   The **Choose a Type** dialog box opens (Figure 93).

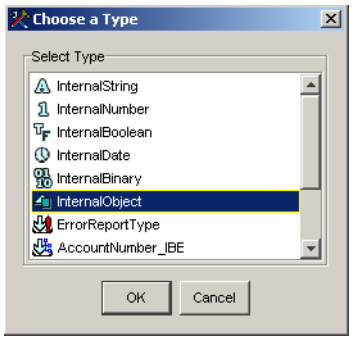


Figure 93. The Choose a Type Dialog Box

2. Select the data type to be assigned to the new global variable. Available options are standard data types (e.g. InternalString) or previously-defined IBEs (e.g. **AccountNumber_IBE**).
3. Click **OK**.
   The new global variable is added to the **Instance** list. Its default name consists of the prefix `gv` followed by the name of the data type it was copied from. If more than one global variable has been copied from that data type, an integer is added to make the name unique (e.g. `gvString_3`).

Deleting Business Entities

All BEs and Global Variables are deleted in the same way, with the exception of Legacy Screen XBEs that are managed exclusively in MapMaker's **Data** tab.

To delete a BE or a Global Variable:

Select the data type you wish to delete and click **Delete**.

The data type is removed from the list.

Importing Business Entities

BEs of IBE, XML/XSD or MLM type that have been previously exported out of the system can be imported back into the Business Entity Editor.

> **Note:** The import operation creates a copy of the exported BE and brings that copy back into the Business Entity Editor. The exported BE itself remains intact in the exported files directory.

BEs are imported as webMethods JI Exchange files (`*.jix`), which are internal, proprietary files that must not be edited.

> **Note:** XML documents may also be imported in `*.xml` format.

A `*.jix` file of a single BE contains all data types (public and private) that are elements of that BE structure. When you import a BE in a `*.jix` format, webMethods JI provides you a list of all public data types included in the file, and you can choose the specific public type(s) to be imported.

> **Note:** All dependent data types are automatically imported (regardless of the public types you select manually).

The default directory from which BEs are imported is set in the **Properties** dialog box, in the **Default Directories** tab's **Map Directory** field (see "Default Directories Tab" on page 100). By default, the import operation searches for `*.jix` files in the selected directory. These settings can be changed as needed.

When a data type is imported, the system checks if it already exists in the Business Entity Editor. If not, it is added to the **Structure Definition** list. If it does exist, no files are added but the source and the imported BEs are compared and any changes are merged. The existing source BE takes precedence over the imported BE. If there is a name or type difference, the existing BE takes precedence. The only possible change to an existing BE is the addition of new elements.

To import a BE from an external file:

1. Display the tab managing the type of BE you wish to import: **Internal Data Definitions**, **XML/XSD** or **Map-List-Map (MLM)**.
2. Click **Import**.

   An **Import Warning** message is displayed, informing you that this operation may modify your current BEs and is not automatically reversible.
3. Click **Continue**.

   The **Import from File** dialog box opens (Figure 94).

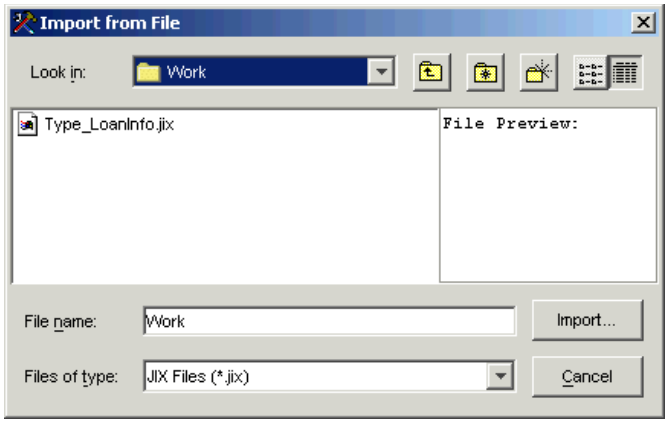


Figure 94. The Import from File Dialog Box

4. By default, this dialog box displays all the `*.jix` files saved in your map directory.
   - To import a different type of file, select **All Files (*.*)** from the drop-down list.
   - To import a file from a different location, browse to the desired directory.
5. Select the file to be imported.

   The **File Preview** pane displays all public types included in the selected file (Figure 95).

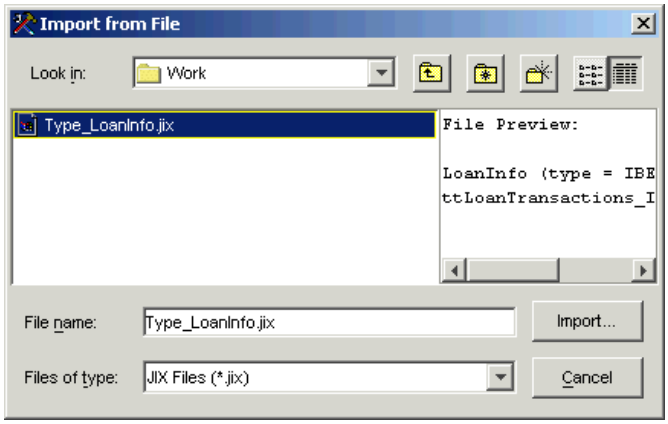


Figure 95. Import from File Dialog Box > File Preview

6 Click **Import**.

The **Choose a Type** dialog box is displayed, listing all public data types included in the selected `*.jix` file (Figure 96).

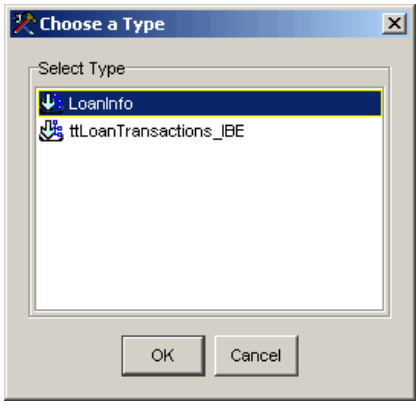


Figure 96. Choose a Type Dialog Box

7 Select the public data type(s) you wish to import and click **OK**.

> **Note:** You can select several types using the **Shift** and **Ctrl** keys. All dependent elements (public or private) are automatically imported, even if they are not manually selected.

A **Finished Importing** message is displayed, specifying how many items were added or merged and how many errors occurred in the import process.

8 Click **OK** to close the message box.

The BE you have imported is added to the **Structure Definition** list.

Importing BEs Containing Boolean Data Types

When a business element containing a boolean data type is imported, if the input formats of the boolean contain one-state formats, the one-state formats are moved to the bottom of the formats list and a warning message is inserted into the details log. The one-state formats are moved to the bottom of the formats list to ensure that they have lowest precedence when the input field is being examined at runtime.

If the import process finds more than one-state format for a boolean item, a warning message is inserted into the details log. See also "One-State Boolean Input Formats" on page 239.

Exporting Business Entities

BEs of an IBE, XML/XSD or MLM type can be exported from the Business Entity Editor to an external directory. BEs can be exported only as `*.jix` files.

> **Note:** The export operation saves a copy of the source BE in the desired location, while the source BE itself remains intact in the Business Entity Editor.

All public and private data types that are dependent on the exported BE (that is, all data types that are elements of the exported BE structure) are automatically included in the export operation.

The default directory to which BEs are exported is set in the **Properties** dialog box, in the **Default Directories** tab's **Map Directory** field (see "Default Directories Tab" on page 100). The default name of the exported file consists of the prefix `Type`, followed by an underscore, the source BE name and the file extension `.jix` (e.g. `Type_LoanInfo.jix`).

To export a BE to a JIX directory:

1. Select the desired data type and click **Export**.

   The **Export to File** dialog box opens (Figure 97).

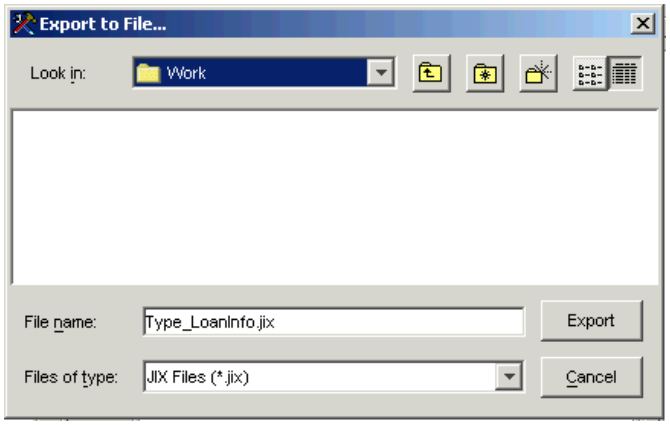


Figure 97. The Export to File Dialog Box

2 Browse to the appropriate directory and click **Export**.

The exported BE file is created in the selected location, as indicated by the **File was Saved** status message.

> **Note:** The exported BE file name is automatically generated based on the source BE name (e.g. Type_LoanInfo.jix)

3 Click **OK** to close the status message window.

Duplicating Business Entities

All BEs and Global Variables are duplicated in the same way, except for Legacy XBEs that are managed exclusively in MapMaker's **Data** tab.

To duplicate a BE (or a Global Variable):

1 Select the object you wish to duplicate and click **Duplicate**.

The selected data type is copied into the list with a unique new name.

- The default name of a copied BE consists of the source BE's name, followed by an underscore and a serial number ranging from 0 to n (e.g. `LoanInfo_0`).
- The default name of a copied Global Variable consists of the prefix `gv` followed by the name of the data type it was copied from. If more than one Global Variable has been copied from that data type, an integer is added to make the name unique (e.g. `gvString_3`).

2 Rename the new object in one of the following ways:

- Double-click on it in the **Instance** list.
- Right-click on it and select **Rename**

- Select it and press the **F2** key.

3 Enter the new name, and then apply it by pressing **Enter** or selecting another item in the list.

Copying a BE from One Data Type to Other Data Types

The "Copy To" operation allows you to copy a selected BE from one data type into other data types. In addition to creating copies of the source BE, the "Copy To" operation allows you to do the following:

- Synchronize the source and the copied BEs (see "Synchronizing the Source BE with the Copied BE(s)" on page 228).
- Define the relationship between the elements of the source and the copied BEs (see "Creating Relations Between the Source BE and the Copied BE(s)" on page 231).

Note: The "Copy To" operation does not allow you to copy a BE into the same data type. To do so, use the Business Entity Editor's **Duplicate** button.

For data to be transferred to and from the webMethods JI service, each XBE must have a corresponding IBE. Therefore, an XBE's "Copy To" operation always includes a mandatory IBE target, while other target data types are optional.

The following table summarizes the mandatory and optional target data types (D.T.) available per source data type:

Source D.T.	Mandatory Target D.T.	Optional Target D.T.
IBE	None	XML XSD, MLM
XML/XSD	Internal	MLM
MLM	Internal	XSD
Legacy	Internal	XML/XSD, MLM

Synchronizing the Source BE with the Copied BE(s) Synchronization enables you to maintain an element-per-element identity between the source BE and the copied BE(s). This feature facilitates and expedites BE management, since manually updating an element of the source BE automatically updates all corresponding elements in the synchronized, copied BEs.

For example, synchronizing a source IBE with its copied XML/XSD XBE allows you to use the latter as the method output. In this case, whenever the IBE is updated, the method output is automatically updated.

You can use the synchronization mechanism to create chains of synchronized objects.

To view all BEs that are synchronized with a BE of interest:

1. Display the BE of interest in the **Structure** pane.
2. Right click the root element of the structure, and select **Show Synchronization** from the shortcut menu.

   The **Synchronized Business Entities** dialog box is displayed, showing all BEs that are synchronized with the selected BE.

   Figure 98 shows the BEs that are synchronized with the **ttLoanTransactions** Legacy BE.

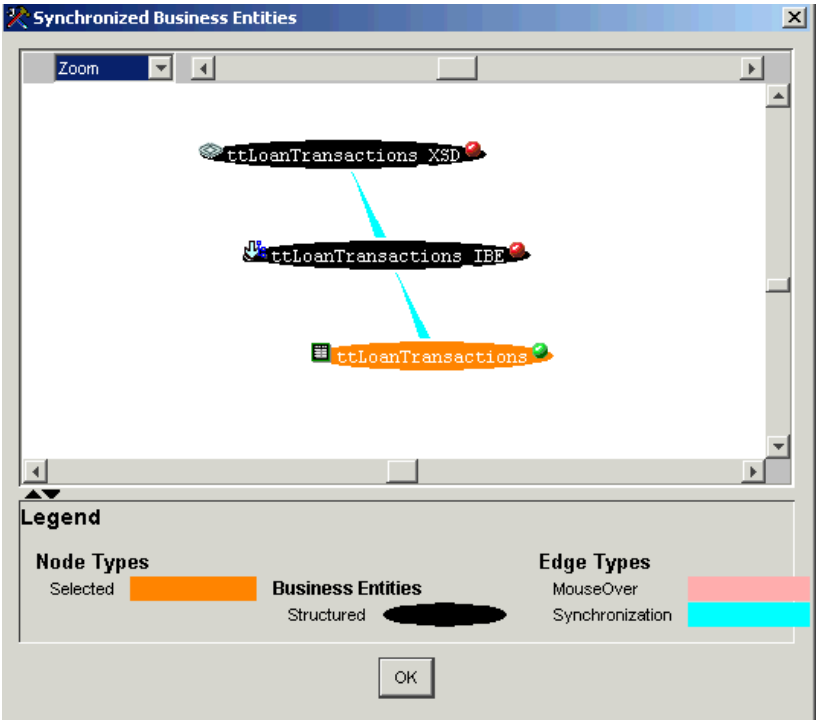


Figure 98. Synchronized Business Entities Dialog Box

The **Synchronized Business Entities** dialog box provides the following options:

Option	Description
Drop-down list	The mode selected in the drop-down list determines the functionality of the top, horizontal scroll bar: • **Zoom** (the default mode): Allows you to use the top scroll bar to zoom in and out of the display, in respect to the selected BE. This mode does not change the display's content, which by default shows only BEs that are synchronized with the selected BE. • **Rotate**: Allows you to use the top scroll-bar to move the BE around, adjusting the display as needed. • **Locality**: Allows you to use the top scroll bar as a filtering mechanism, controlling the display's scope of reference. Moving the scroll bar all the way to the right changes the display's content, expanding from the selected BE through all BEs it is synchronized with, up to all BEs defined in the system.

Option	Description
Node Types	• **Selected**: The BE you have selected in the **Structure** pane is colored orange (the BEs it is synchronized with are all colored black). • **Business Entities**: **Structured** BEs are indicated by a circle. **Note:** The **Synchronized Business Entities** dialog box does not show single elements, only structures.
Edge Types	• **MouseOver**: The link that is currently selected by the cursor is colored pink. • **Synchronized**: The links between synchronized edges are colored turquoise.

To break the synchronization between BEs:

1. Display either the source or the copied BE in the **Structure** pane.
2. Right click the BE and choose **Remove Synchronization** from the shortcut menu.

   The **Choose a Type** dialog box is displayed, listing all BEs that are synchronized with the selected BE (a source BE may be synchronized with multiple copies).
3. Select the BE with which the synchronization is to be broken, and click **OK**. The **Confirm Change** message is displayed.
4. Click **OK** to break the synchronization.

Note: Once synchronization has been broken, it cannot be reestablished.

Creating Relations Between the Source BE and the Copied BE(s) While performing a "Copy To" operation, you can automatically create one-to-one mapping relations between each element of the source BE and the corresponding elements of the copied BE(s). Creating these relations enables information to be automatically transferred between elements that are mapped to each other. Creating relations while performing a "Copy To" operation facilitates and expedites the mapping process, by saving you the need to define each of these relations manually in the Structure Relationship Editor.

If the "Copy To" operation includes synchronization, relations are automatically created as well. If the "Copy To" operation does not include synchronization, you can choose whether or not to create relations.

To copy a BE from one data type to other data types, proceed as follows:

1 In the **Structure Definition** list, select the BE you wish to copy.

2 Click the **Copy To** button.

The **Copy To** dialog box opens, showing the selected BE's **Name** and **Type** and enabling the appropriate **Target(s)** data types to which this BE can be copied.

Example 10 - 1

Figure 99 shows an example **Copy To** dialog box, allowing you to copy an **Internal Business Entity** named **LoanInfo** to an **XSD** and/or an **MLM** BE **Target(s)**.

Note: Since the "Copy To" operation does not allow you to copy a BE into the same data type, the **Internal** check box is cleared.

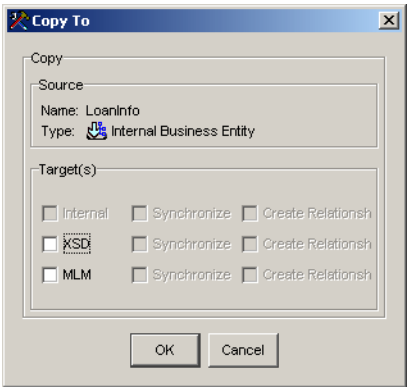


Figure 99. Copying an IBE

1 Check the **Target(s)** BE type(s) to which you wish to copy the source BE.

2 Click **OK**.

The copies of the XML/XSD XBE are added to the appropriate Business Entity Editor tabs.

Indicating a Data Type is an Array

An array is a repeating data type. webMethods JI supports several types of arrays:

- A table—in the webMethods JI context, a table is an array since it contains a a repeating set of rows.

- A repeating single primitive element (e.g a String or a Number)—for example, a repeating list of ID numbers or part numbers.
- A combination of a table and repeating primitive elements—for example, a list of customer orders. Such a list contains repeating rows of customer orders, which in turn contain repeating purchase numbers and part numbers.

You can control the **isArray** setting of any data type element, with the following exceptions:

- Root elements — a structure's root element cannot be an array.
- Synchronized BE copies—a copied BE that is synchronized with a source BE automatically reflects the source's **isArray** settings. Any changes to these settings must be applied directly to the source BE.
- MLM elements — theoretically, any MLM element can repeat itself. Therefore, any MLM element (except for the root element) is automatically defined as an array.
- Legacy elements — these are reflections of the data templates and table templates created in MapMaker's **Data** tab. In this case, the table templates are merely definitions, as opposed to actual instances of the information. Therefore, Legacy elements cannot be set as arrays.
- Global variables — a Global Variable is an instance of either a primitive definition or an IBE definition. A Global Variable cannot contain any elements and cannot repeat itself. Whether or not a Global Variable is an array is a function of its source's **isArray** setting: primitives cannot be set as arrays. However, if the source IBE contains an array element, the corresponding Global Variable is automatically set as an array.

To indicate that an instance is an array:

1. Select the structure containing the instance in question in the **Structure Definition** pane.
2. Select the instance in question in the **Detail** pane.
3. Set the **isArray** check box.
4. Click **OK**.

Note: An XML/XSD particle's **isArray** property is associated with its **MaxOccurs** property (see "Editing an XML/XSD Particle" on page 244): If **MaxOccurs** is larger than 1, the particle is automatically defined as an array. Similarly, clearing the particle's **isArray** box automatically sets its **MaxOccurs** property to 1.

The Business Entity Editor Confirm Change Dialog Box

Some of the actions performed on Business Entities and Global Variables require multiple steps. When you invoke such a multi-step action, a subset of actions known as ChangeActions is generated to perform these steps.

A ChangeAction set may:

- Delete a Business Entity or Global Variable.
- Remove the synchronization between Business Entities.
- Change an element's data type.
- Promote a data type from private to public, or demote it from public to private.
- Contain an error (e.g. perform an invalid name change, using illegal characters).

In these cases, the Business Entity Editor displays the **Confirm Change** dialog box (Figure 100). This dialog box details all of the actions to be completed as part of the larger ChangeAction, and allows you to either execute or abort the action.

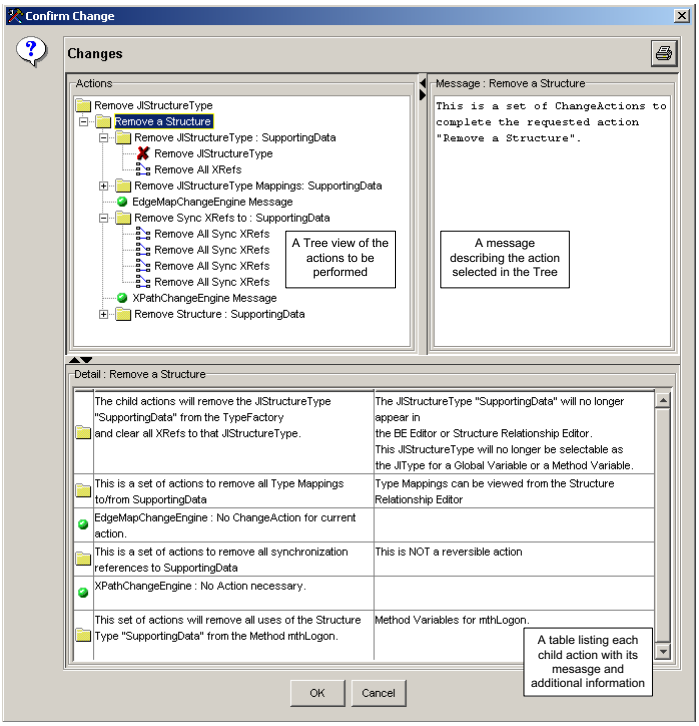


Figure 100. The Confirm Change Dialog Box

The **Confirm Change** dialog box provides the following options:

Option	Description
Actions Tree	Provides a hierarchical view of the ChangeAction set. When this set is executed, the child ChangeActions are invoked in a depth-first/top-down order.
Message Pane	This text area provides a detailed account of all operations to be performed by the ChangeAction currently selected in the Tree.
Details Table	Lists all the actions to be performed, providing the following information about each one: • **Icon**: Indicates the type of the selected action. • **Message**: Informs you what the action is going to do. • **Info**: Provides additional information about the ChangeAction, such as the MapMaker component that shows the results of the change, or a warning about possible ramifications of this ChangeAction.
OK	Executes the change.
Cancel	Aborts the change (this option is always available).

Note: If there is an error in the ChangeAction set, the dialog box's name changes from **Confirm Change** to **Cannot Change**. In this case, the **OK** button is not displayed and the only available button is **Cancel**.

The **Actions** tree and **Details** table are associated as follows:

- Selecting a node in the **Actions** tree displays its children in the **Details** table.
- Selecting a row in the table highlights the corresponding tree node (in a different color than the currently selected source ChangeAction), but does not change the context of the table (i.e. the node whose children are displayed).
- Double-clicking a row in the table has the same effect as selecting a node in the tree: the children of this node are displayed in the table.

Formatting Data Types

The Business Entity Editor provides you with the tools to create different formats for the inputs and outputs of certain IBE and XSD data types. This capability allows you to receive input information in one format and return it as output information in a different format that addresses your specific needs.

For example, you can create an InternalDate input format, which can receive date input from a green screen in the abbreviated “MM/dd/yyyy” layout (e.g., “08/30/2003”); then use an InternalDate output format to transform the data into a more detailed, “EEE, MMM d, “yy” layout (e.g. “Sat, August 30, '03"), and finally return the information to the client using an XSD string.

The following data types support formatting, each using its own formatting language:

- **Number** — The number language is defined by the Java class `java.text.DecimalFormat`. This can be found at: http://download.oracle.com/javase/1.4.2/docs/api/java/text/decimalformat.html

  The following table provides examples of common Number formats:

Format	Example
#,##0.00;(#,##0.00)	1,230.00 'or' (1,230.00)
#,##,###,####	1,23,456,7891 'or' -1,23,456,7891
$###,###.##;$(###,###.##)	$987,654.00 'or' $(987,654.00)

- **Date** — The date language is defined by the Java class `java.text.SimpleDateFormat`. This can be found at: http://download.oracle.com/javase/1.4.2/docs/api/java/text/dateformat.html

  The following table provides examples of common Date formats:

Format	Example
EEE, MMM d, ''yy	Wed, July 10, '96
h:mm a	12:08 PM
hh 'o''clock' a, zzzz	12 o'clock PM, Pacific Daylight Time

- **Boolean** — The boolean language is a proprietary webMethods JI language, defined by the Java class `com.jacada.util.BooleanFormat`. This language can be found at: *<JI_Install_Dir>/html/easerver/com/jacada/util/BooleanFormat.html*.

  The following table provides examples of common Boolean formats:

Format	Example
true;false	true 'or' false
yes;no	yes 'or' no
ok;cancel	ok 'or' cancel
1	1 'or' <any other value>

Note: The language used by each data type is identical for IBEs and XBEs.

To format a data type:

Data type inputs and outputs are formatted using the Business Entity Editor's shortcut menu **Edit Format** option.

1. In the Business Entity Editor, display the tab managing the data type you wish to format (**Internal Data Definitions** or **XML/XSD**).
2. Select the data type you wish to format in the **Structure Definition** pane, to display its structure in the **Structure** pane.

3 Right click the element you wish to format (Number, Date or Boolean).

The Business Entity Editor's shortcut menu is displayed.

Figure 101 shows the shortcut menu of the **Internal Data Definitions** tab's **Structure** pane (a corresponding menu is available in the **XML/XSD** tab's **XML Structure** pane).

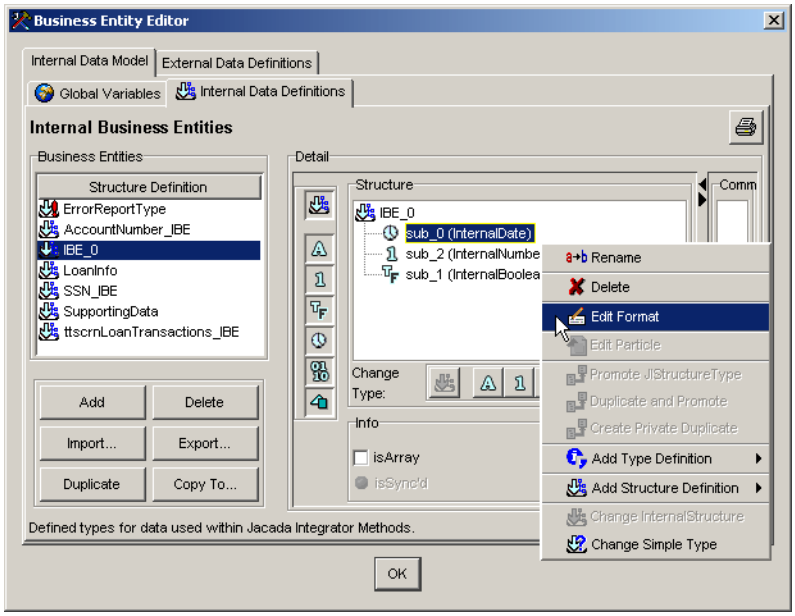


Figure 101. Internal Data Definitions Tab > Edit Format Menu Option

4 From the shortcut menu, select the **Edit Format** option.

The **Edit Format** dialog box is displayed, allowing you to manage both input and output data type formats. The following Date and Boolean data type formats are available by default (there are no default Number data type formats):

- Date — EEE MMM dd HH:mm:ss z yyyy
- Boolean — true;false, yes;no, ok;cancel, 1

Note: When defining boolean data type formats, the first value is considered "true" and the second value is considered "false".

Figure 102shows the **Edit Format** dialog box of Date data types.

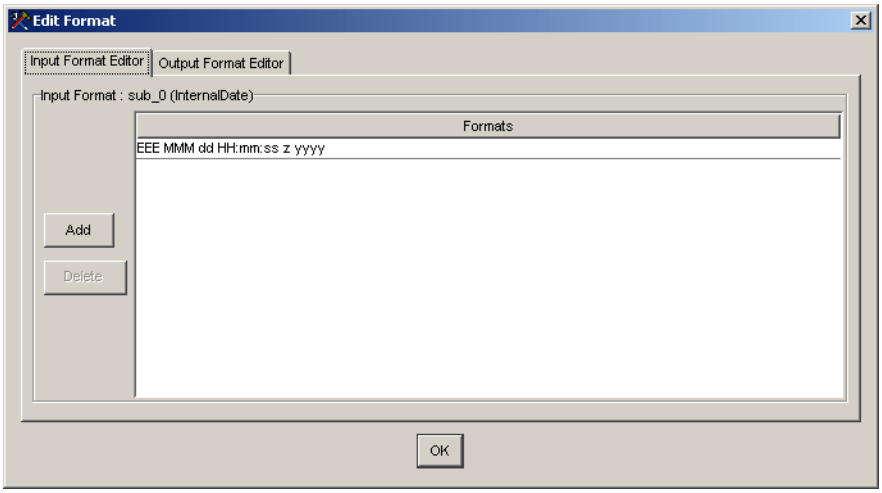


Figure 102. The Date Data Type Edit Format Dialog Box > Input Format Editor Tab

Formatting Inputs

A single Number, Date or Boolean data type definition can have multiple data type formats. The format's position in the **Formats** list determines its precedence: at run time, the formats are attempted from the top down, in the order in which they are listed. Once a match is found, that format is used to transform the information into the applicable back end type. Any formats placed lower on the list are not checked.

> **Note:** Partial matches are considered a full match, because the formatting objects return a valid input when there is more information in the data than the format requires. If a partial match is listed before a full match, the full match is not checked. Therefore, to avoid data loss, it is important to place the more specific formats before the less specific ones.

To add an input format:

1. In the **Input Format Editor** tab, click **Add**.

   A new entry is added to the **Formats** table, reading `(Enter Format Here)`.
2. Double click the new entry to edit it and enter the desired format.
3. Continue to add all of the input formats required for this data type.
4. Click **OK** to save your changes.

One-State Boolean Input Formats

A one-state boolean format is a format where just one value is defined. For example, the last line in Figure 103 that contains "1" is a one-state format.

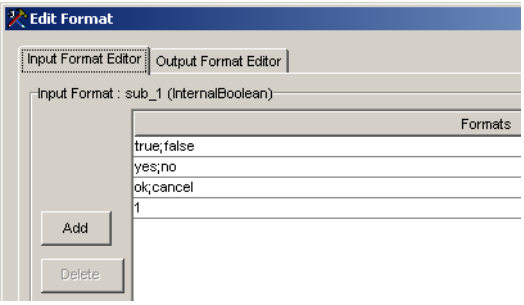


Figure 103. The Boolean Type Edit Format Dialog Box > Input Format Editor Tab

When a one-state format is added to a boolean data type, after you press **OK** the one-state format is moved to the bottom of the list. This ensures that it has lowest precedence over the two-state formulas at runtime when the input field is being checked.

If you try to add a second one-state format to a boolean format, a warning dialog is displayed and the newly-entered one-state format is deleted.

If you modify an existing two-state format and change it to a one-state format resulting in there being more than one one-state format, a warning dialog is displayed and the change is disallowed.

See also “Importing BEs Containing Boolean Data Types” on page 226.

Formatting Outputs

Output Data types are formatted using the **Edit Format** dialog box’s **Output Format Editor** tab (Figure 104).

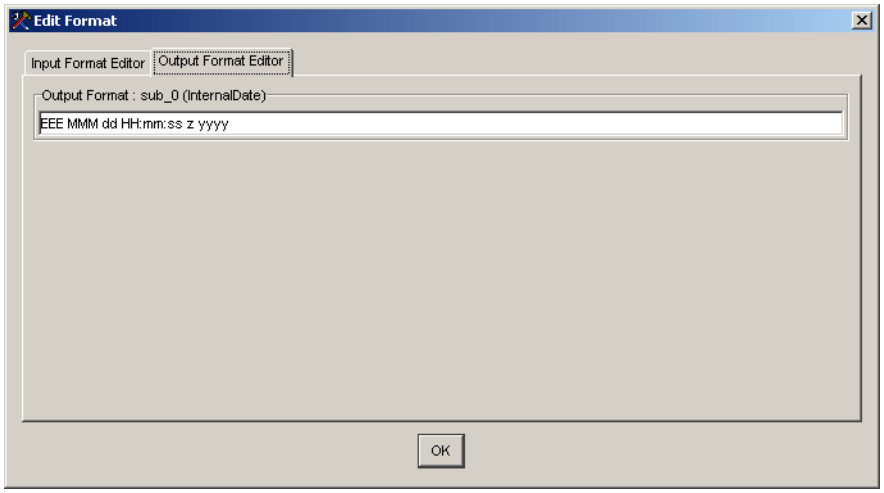


Figure 104. The Date Data Type Edit Format Dialog Box > Input Format Editor Tab

While a Number, Date or Boolean data type definition may have multiple input formats, they can only have a single output format. When data from a webMethods JI type definition (e.g. an Internal Date) with an output format is assigned to another webMethods JI type definition (e.g. an XSD String), the data is transformed by the output format before being sent on the client. This allows you to customize the format of the numeric, date and boolean values to address your specific needs.

Note: You are not limited to the data format provided by the back end data source. For example, given an Internal Boolean with a "true" value, an output format can transform a "true" value into the string "Tuesday".

To add an output format:

In the **Output Format Editor** tab, enter the data type's format into the **Output Format** property, and click **OK** to save your changes.

The Business Entity Editor Tabs

This section describes:

- "The Internal Data Definitions Tab" on page 242
- "The Global Variables Tab" on page 243
- "The XML/XSD Tab" on page 244
- "The MLM Tab" on page 246
- "The Legacy Screen Tab" on page 247

The Internal Data Definitions Tab

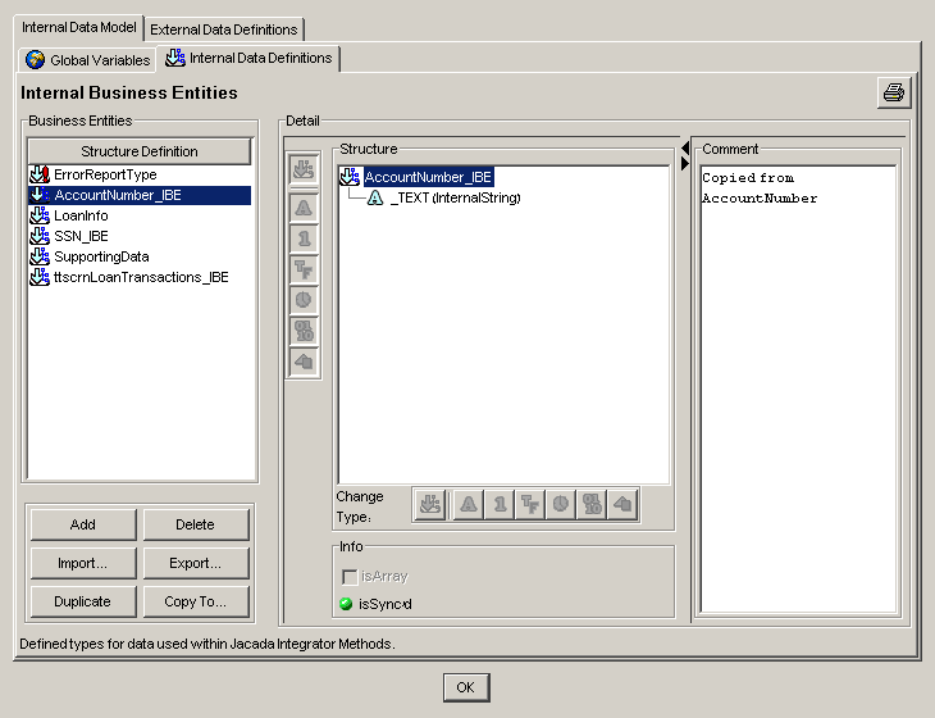


Figure 105. The Internal Data Definitions Tab

The **Internal Data Definitions** tab (Figure 105) displays the existing IBE data types and provides the necessary tools to edit and add new IBEs.

The Global Variables Tab

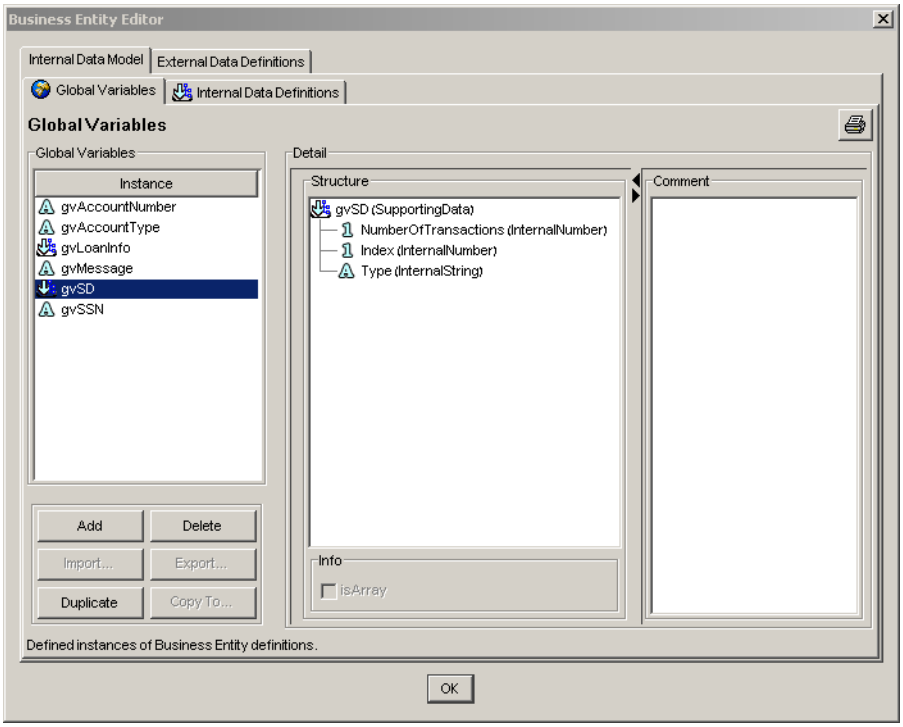


Figure 106. The Global Variables Tab

The **Global Variables** tab (Figure 106) displays all existing Global Variables, and provides the necessary tools for managing them.

> **Note:** Global Variables can also be added, deleted and duplicated from the **Edit Variables** dialog box. For more information, see "The Edit Variables Dialog Box Shortcut Menus" on page 285.

The XML/XSD Tab

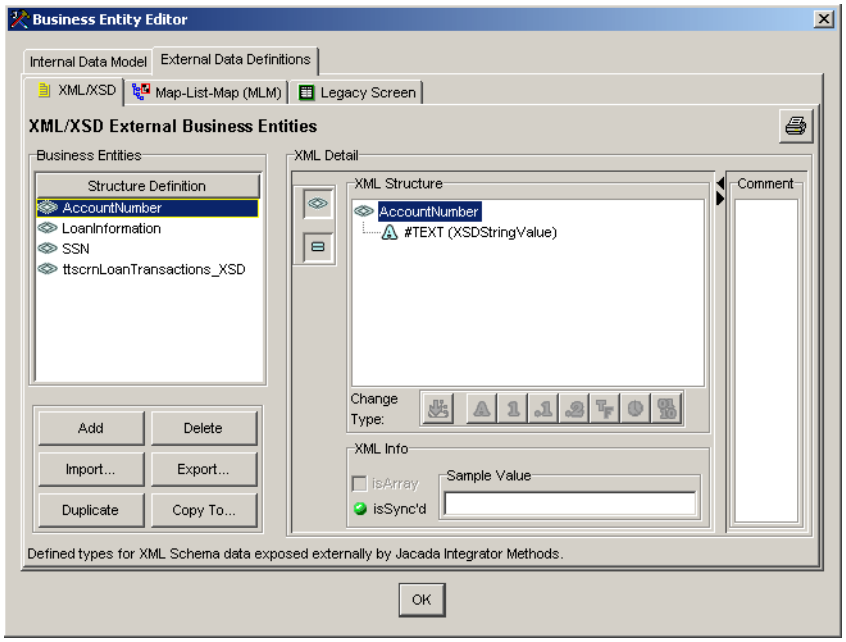


Figure 107. The XML/XSD Tab

The **XML/XSD** tab (Figure 107) displays all the XML/XSD XBEs and provides the tools necessary for managing them.

Editing an XML/XSD Particle

When an XML input is received, webMethods JI verifies that each of the XML structure's elements (or particles) occurs as many times as needed. By default, there must be at least one occurrence of each particle, but you may change the number of required occurrences as needed.

At runtime, before any data has been received, webMethods JI sets up the default XML structure to be used as an output, based on the minimum number of occurrences defined for each particle. The runtime validation of the particle count determines whether or not the XML document as a whole is valid. If it is not valid, the XML is returned to the client (with an error message).

To edit an XML/XSD particle:

1 In the **XML/XSD** tab, select the structure containing the particle you wish to edit in the **Structure Definition** pane.

  The structure's particles are displayed in the **Structure** pane.

2 Right-click the particle you wish to edit and select **Edit Particle** from the shortcut menu.

The **Edit Particle** dialog box is displayed (Figure 108).

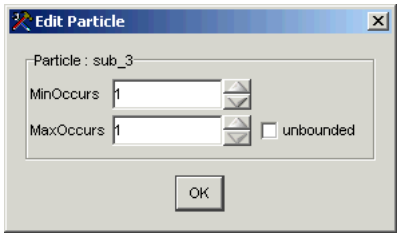


Figure 108. Edit Particle Dialog Box

3 Define the particle's valid number of occurrences:

- To define a specific number of occurrence, enter the same value into the **MinOccurs** and **MaxOccurs** fields (default is 1).
- To define a range of valid occurrence, enter the lowest and the highest valid numbers of occurrence into the **MinOccurs** and **MaxOccurs** fields, respectively.
- To allow an unlimited number of occurrences, set the **Unbounded** checkbox.
- To indicate the element is not required to be included in the generated XML, set **MinOccurs** to 0.

4 Click **OK** to apply the changes.

Note: An XML/XSD particle's **MaxOccurs** property is associated with its **isArray** property (see "Indicating a Data Type is an Array" on page 232): if **MaxOccurs** is larger than 1, the particle is automatically defined as an array. Similarly, clearing the particle's **isArray** box automatically sets its **MaxOccurs** property to 1.

The MLM Tab

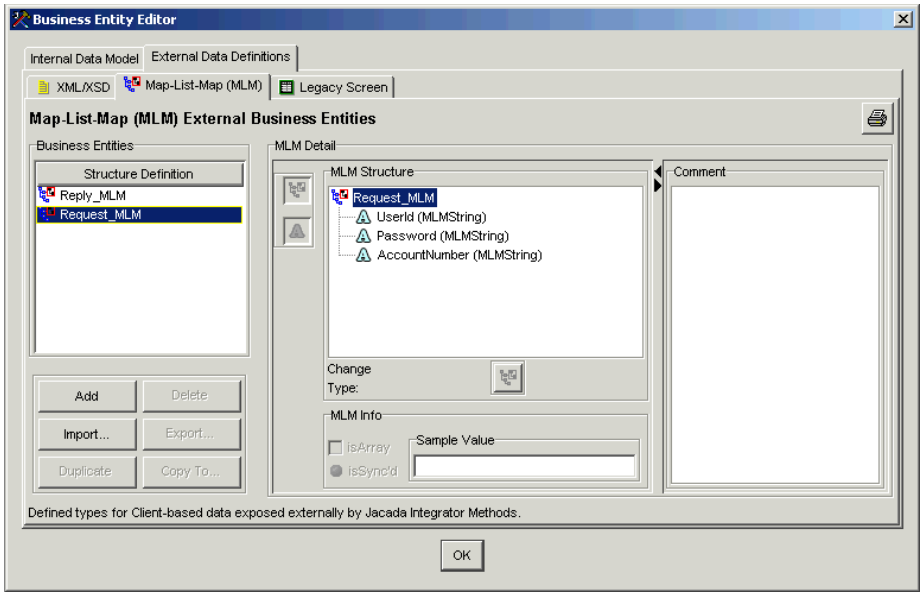


Figure 109. The MLM Tab

The **MLM** tab (Figure 109) displays all the MLM XBEs and provides the tools necessary for managing them.

The Legacy Screen Tab

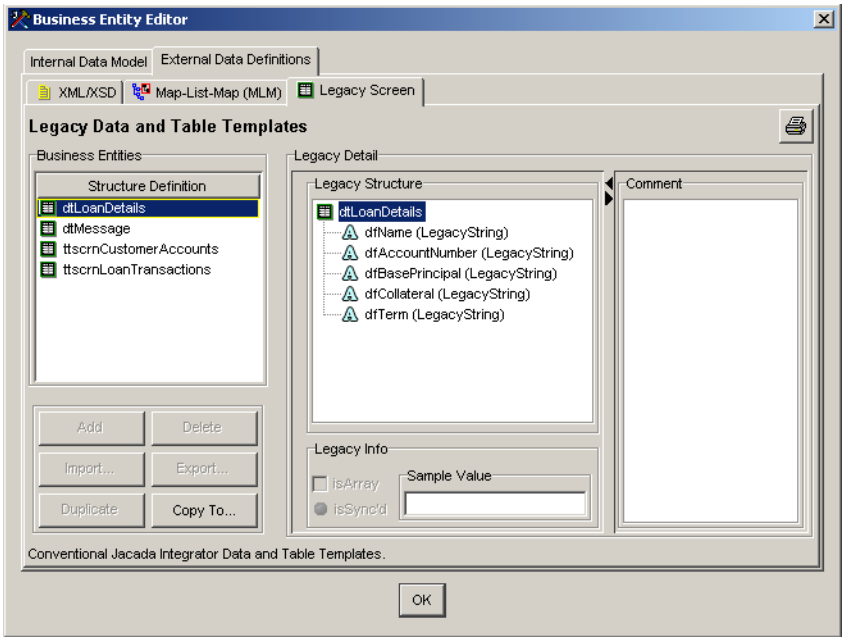


Figure 110. The Legacy Screen Tab

The **Legacy Screen** tab (Figure 110) displays the Legacy Screen XBEs. The only action you can perform on Legacy Screen XBEs is copying them. This is because Legacy Screen XBEs are managed exclusively in MapMaker's **Data** tab. All template changes performed in the **Data** tab are automatically reflected in the Business Entity Editor's **Legacy Screen** tab.

Creating a Legacy Screen XBE

By creating a data template or a table template in MapMaker's **Data** tab, you effectively create a Legacy Screen XBE. For example:

1. In the Tree view, right-click on the **scrnCustomerLookup** screen and select **Add Data Template** from the pop-up menu.
2. Rename the new data template `dtAccountLookup`.
3. With **dtAccountLookup** selected in the Tree view, select the account number field in the Presentation view (`1234567898`) by double clicking on it.
4. Right-click on the field and choose **Add Data Field** from the shortcut menu.
5. In the Tree view, rename the new data field that has been added to the **dtAccountLookup** data template `dfAccountNumber`.

   This field is the field in which the end-user must type an account number.

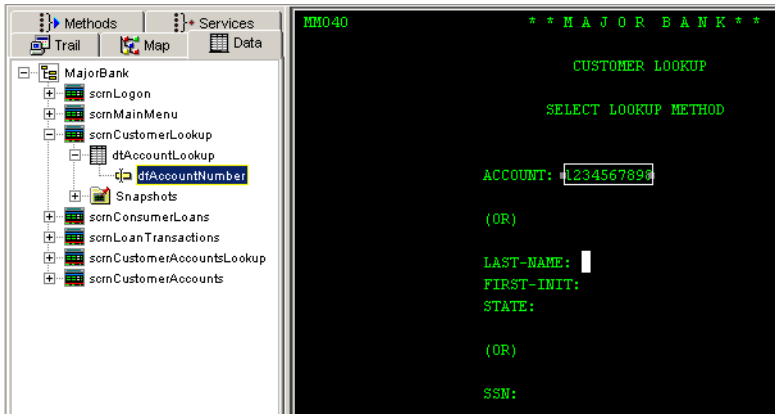


Figure 111. Creating a Data Template

> **Note:** For instructions on working with data templates and table templates, see Chapter 6 - "Data" on page 181.

The Business Entity Editor's **Legacy Screen** tab displays this data template as a Legacy Screen XBE data type. This is seen in Figure 112.

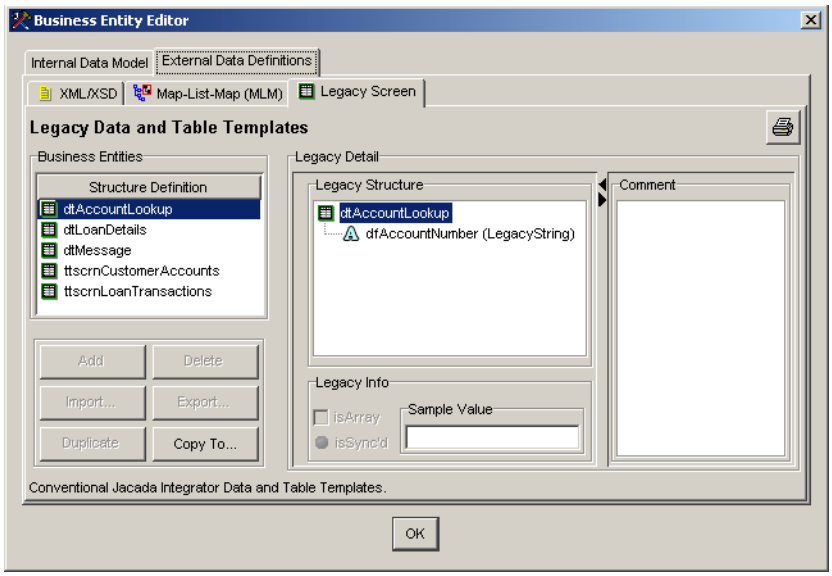


Figure 112. A Data Template Automatically Displayed as a Legacy Screen XBE

Defining Data Structure Relationships

Structure relationships provide a mechanism for transforming data from one data type to another according to a defined set of rules. Data structure relationships are defined in the **Structure Relationship Editor** (Figure 113).

The Structure Relationship Editor

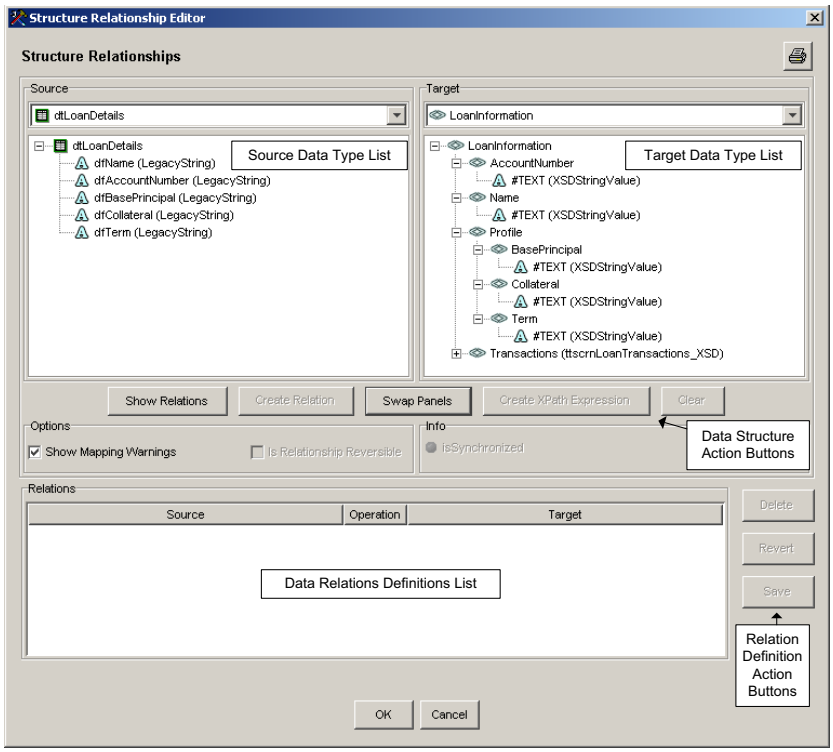


Figure 113. The Structure Relationship Editor

The Structure Relationship Editor is used to define a relationship between two data types, or edit an existing relationship.

The Structure Relationship Editor consists of the following components:

- A **Source** list box containing all data types.
- A **Target** list box containing all data types.
- A **Relations** list box displaying the set of relations, if any exist, between the data type selected in the **Source** list and the one selected in the **Target** list.
- The tools necessary for defining or editing these relations.

The Structure Relationship Editor provides the following options:

Option	Description
Print ( )	Prints all or part of the Structure Relationship Editor's display.
Create Relation	Creates a relation between the data type selected in the **Source** list and the one selected in the **Target** list. Alternatively, you can use the Structure Relationship Editor's drag and drop functionality: Drag an element from the **Source** list to the **Target** list, and drop it on top of the element you wish to create a relation to. **Note:** This operation is enabled only when the conversion between the data types selected in the **Source** and **Target** lists is allowed by the conversion rules (see "Data Type Conversion Rules" on page 252).
Swap Panels	Causes the content of the **Source** and **Target** lists to be swapped. This operation clears all selections made in these lists. **Note:** Once the **Create Relation** button is clicked, the **Swap Panels** button is disabled until the relationship is either saved or reverted.
Create XPath Expression	Enables you to define the relation between a **Source** data type and a **Target** data type through an XPath expression. **Note:** This button is enabled only when the **Source** data type is the parent.
Clear	Clears current selections in both the **Source** and the **Target** lists.
Show Mapping Warnings	Specifies whether or not to issue a warning during data mapping, if applicable.

Option	Description
Is Relationship Reversible	Specifies whether or not the relationship can be reversed. **Note:** This property is disabled when it is not possible to create a reversible relationship.
Is Synchronized	Indicates that the **Source** data structure is synchronized with the **Target** data structure (see "Synchronizing the Source BE with the Copied BE(s)" on page 228)
Delete	Deletes the selected relation(s) from the **Relations** list.
Reset	Reverts the **Relations** list to the last save.
Save	Saves changes to the **Relations** list.

Defining a Relationship Between Two Data Types

To define the relationship between two Data Types:

1. In the **Source** list box, expand the Source data type from which you wish to create a structure relationship.
2. In the **Target** list box, expand the Target data type to which you wish to create a structure relationship.
3. Select a sub-element from the expanded **Source** data type, then select the sub-element from the **Target** data type to which you want to relate the Source sub-element, and click **Create Relation**.

> **Note:** This button is enabled only when the conversion between the source and target data types is allowed by the conversion rules (see "Data Type Conversion Rules" on page 252).

4. Repeat the previous step for other sub-elements, until you have finished relating all **Source** sub-elements with their desired **Target** counterparts.
5. Click **Save**.

Data Type Conversion Rules

During runtime, the method attempts to perform the data mappings defined in the Structure Relationship Editor (see "The Structure Relationship Editor" on page 249) and in the Data Mapping Editor (see "The Data Mapping Editor" on page 268). These data mappings include conversions from one data type to another, based on the following definitions:

- The input and output variables' formatting (see "Formatting Data Types" on page 236), which describe how conversions are to be accomplished.
- A set of data type conversion rules, which describe which conversions are valid.

The table in Figure 114 lists all combinations of source (vertical axis)-to-target (horizontal axis) data type conversions, and specifies whether each conversion is allowed using the following key:

- Au — auto conversions (or self conversions): these allow you to move data from one Internal type to the same Internal type (e.g. copying data from an Internal Boolean to another Internal Boolean).
- A— allowed conversions: these allow you to move data from any given BE type to a different BE type (e.g. from an Internal Boolean to an Internal String).
- P — "probably allowed" conversions: these conversions are allowed by webMethods JI, but might fail at runtime.
- N — no conversion: this particular conversion is not allowed for a particular class of BE, but there are other valid conversions to the same or to another class of BE. For example, Internal Date is not convertible to Internal Binary (No Conversion), but Internal Date *is* convertible to Internal Number (Allowed Conversion).
- X — illegal conversion: there are no valid conversions from one XBE type (XSD, MLM or Legacy) to another XBE type nor to the same XBE type (an auto conversion).

Target / Source	Internal Binary	Internal Boolean	Internal Date	Internal Number	Internal Object	Internal String	Legacy String	MLM String	XSD Attribute	XSD Base64	XSD Boolean	XSD CData	XSD Date	XSD Decimal	XSD Double	XSD Float	XSD String
Internal Binary	Au	N	N	N	A	A	A	A	A	A	N	A	N	N	N	N	A
Internal Boolean	N	Au	N	A	A	A	A	A	A	N	A	A	N	P	P	P	A
Internal Date	N	N	Au	A	A	A	A	A	A	N	N	A	A	A	A	A	A
Internal Number	N	P	A	Au	A	A	A	A	A	N	P	A	A	A	A	A	A
Internal Object	N	N	N	N	Au	A	A	A	A	N	N	A	N	N	N	N	A
Internal String	A	P	P	P	A	Au	A	A	A	A	P	A	P	P	P	P	A
Legacy String	A	P	P	P	A	A	X	X	X	X	X	X	X	X	X	X	X
MLM String	A	P	P	P	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Attribute	A	P	P	P	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Base64	A	N	N	N	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Boolean	N	A	N	A	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD CData	A	P	P	P	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Date	N	N	A	A	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Decimal	N	P	A	A	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Double	N	P	A	A	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD Float	N	P	A	A	A	A	X	X	X	X	X	X	X	X	X	X	X
XSD String	A	P	P	P	A	A	X	X	X	X	X	X	X	X	X	X	X

Figure 114. Data Type Conversion Table

> **Note:** Formattable data types are colored blue.

All XBE String types (Legacy String, MLM String, XSD Attribute, XSD CData and XSD String) have identical conversion properties. XBE to XBE conversions, as well as XBE self-conversions (to the same data type), are not allowed. However, IBE data types are automatically assignable to their own type (regardless of their input/output formatting).

Certain conversions are defined as “probably allowed”, in the sense that they are allowed by webMethods JI but might fail at runtime. Such conversions generally depend on the target’s input formatting to match the source’s output formatting (whether the output data is formatted, e.g. by an Internal Number; or unformatted, e.g. by an Internal String).

> **Note:** When a conversion between a pair of source and target data types is allowed by these rules, the **Create Relation** button is enabled in the Structure Relationship Editor, and the **Create Data Map** button is enabled in the Data Mapping Editor. When such a conversion is not allowed, these buttons are disabled.

If the data type conversion fails at runtime, the method’s onFail edge is invoked. If no onFail edge is defined, the exception is propagated back to the user.

Using XPath Expressions in the Structure Relationship Editor

The Structure Relationship Editor enables you to define the relationship between a **Source** data type and a **Target** data type through an XPath expression. XPath is a language for selectively addressing parts of a hierarchical set of data, such as an XML document.

webMethods JI currently supports XML documents, and internally represents the XML information using a Java-based hierarchical data representation.

MapMaker allows the user to symbolically address desired elements within the internal representation of the XML data. This functions just like an XPath expression operating directly on an XML document. Since webMethods JI stores all XBE, IBE, and Legacy BE data in similar hierarchical data representations, XPath expressions can be used in MapMaker to extract data from any of the available BE types.

Examples of how XPath expressions can be useful include:

- Concatenating two or more separate fields to produce a single field. For example, combining a FirstName field, a space, and a LastName field, and moving the result into a FullName field.
- Decomposing a single field into its components. For example, extracting the Month from a date field that is in the format YYYYMMDD.
- Accessing individual occurrences of a data element in an array.

To define a relation using XPath syntax:

1. In the **Source** data type list, select the root element of the selected data type.
2. In the **Target** data type list, select the sub-element of the data type to which you wish to create a relation.
3. Click **Create an XPath Expression**.

   A new data relation with an empty expression is added at the bottom of the **Relations** list (Figure 115):

Source	Operation	Target
(Enter XPath Expression Here)	XPATH->	Transactions/Type

Figure 115. A New XPath Expression

4. Double-click **Enter XPath Expression Here**.
5. Enter the appropriate XPath expression and press **Enter**.
6. Click **Save**.

MapMaker evaluates all XPath expressions entered in the Structure Relationship Editor for proper syntax and to ensure the validity of any references to Global Variables. Error messages are displayed on the lower left of the **Structure Relationship Editor** dialog box, and a colored ball icon appears to the left of the XPath expression.

- If the ball is red, it indicates a “hard” error that must be corrected.
- If the ball is yellow, it indicates that the XPath expression cannot be fully validated and may or may not work at runtime. The XPath expression is accepted as written. The message line explains why the expression cannot be fully validated.
- If the ball is green, it indicates that the XPath expression is valid.

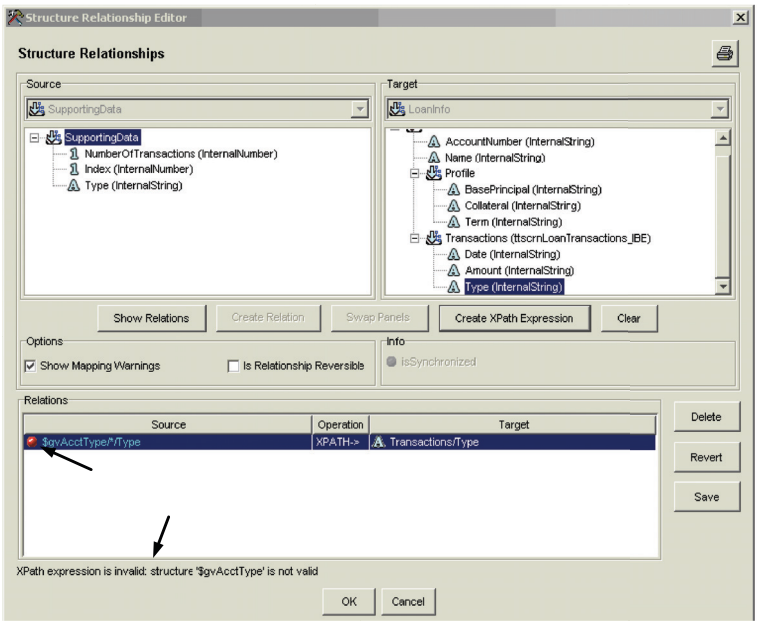


Figure 116. Edit Error Messages for the XPath Expression

Example 10 - 2 Using XPath in the Structure Relationship Editor

XPath expressions can be used to populate one or more elements in the Target from a Source element, and likewise an XPath statement can combine one or more elements in the Source to populate a single element in the Target. Note that a single XPath expression cannot affect more than one Target item.

The relationship between the **Source** data structure (on the left) and the **Target** data structure (on the right) is sketched below (Figure 117).

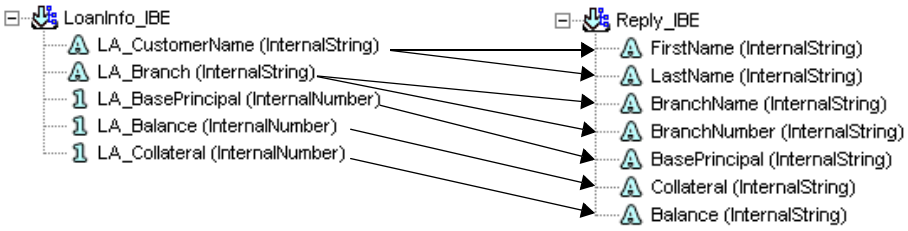


Figure 117. A Complex Structure Relation Definition

In this example:

- The **LA_CustomerName** sub-element in the source IBE is related to two sub-elements in the target IBE: **FirstName** and **LastName**.
- The **LA_Branch** sub-element in the source IBE is related to two sub-elements in target IBE: **BranchName** and **BranchNumber**.
- The rest of the elements are mapped one-to-one.

The data relations between the IBEs appear in the Structure Relationship Editor as follows (Figure 118):

Relations

Source	Operation	Target
substring-before(/*/LA_CustomerName," ")	XPATH->	FirstName
substring-after(/*/LA_CustomerName," ")	XPATH->	LastName
substring(/*/LA_Branch,1,4)	XPATH->	BranchName
LA_BasePrincipal	XPATH->	BasePrincipal
LA_Balance	XPATH->	Balance
LA_Collateral	XPATH->	Collateral

Figure 118. Defining Structure Relationships Through an XPath Expression

We will elaborate on the source IBE's **LA_CustomerName** element:

The data contained in the **LA_CustomerName** element in the source IBE is made up of a first name and a last name, separated by a space (e.g. "`John Doe`"). To relate the **LA_CustomerName** source sub-element to two separate sub-elements, you must create two XPath expressions: one for the **FirstName** target sub-element, and another for the **LastName** target sub-element.

FirstName uses the following XPath expression:

```
substring-before(/*/LA_CustomerName," ")
```

The XPath expression above singles out the part of the **LA_CustomerName** sub-element that comes *before* the first space; that is, the first name.

LastName uses the following XPath expression:

```
substring-after(/*/LA_CustomerName," ")
```

The XPath expression above singles out the part of the **LA_CustomerName** sub-element that comes *after* the first space; that is, the last name.

Note: In XPath expressions, the data relationship is defined from the Source Parent element, by typing an asterisk character /*/ rather than the actual name of the Parent element. This means the current instance of the structure type is used to get the data (e.g. a Global Variable).

Verifying the Result of an XPath Expression

The XPath Evaluator enables you to verify the result of an XPath expression, without generating a method or deploying a service. Successful XPath expressions can then be copied from the XPath Evaluator to either the Structure Relationship Editor or to a Set step in a method.

XPath expressions can be evaluated in two different contexts:

- **Current XPath Context**: Evaluates an XPath expression in the context of values that would be assigned to Global and Method variables at runtime. This requires that the appropriate values are first assigned to variables in a Debugger session before performing the evaluation. In the Debugger, values are assigned to variables either manually or as part of the method logic. For more information about the Debugger, see "Debugging Methods" on page 363.
- **Default XPath Context**: Evaluates an XPath expression in the context of default values that were assigned to Global and Method variables at the time of their definition. This requires that you set the XPath Evaluator's **Use Default XPath Context** checkbox before performing the evaluation.

To verify the result of an XPath expression:

1 Click the **XPath Evaluator** button on the tool bar. The XPath Evaluator is displayed:

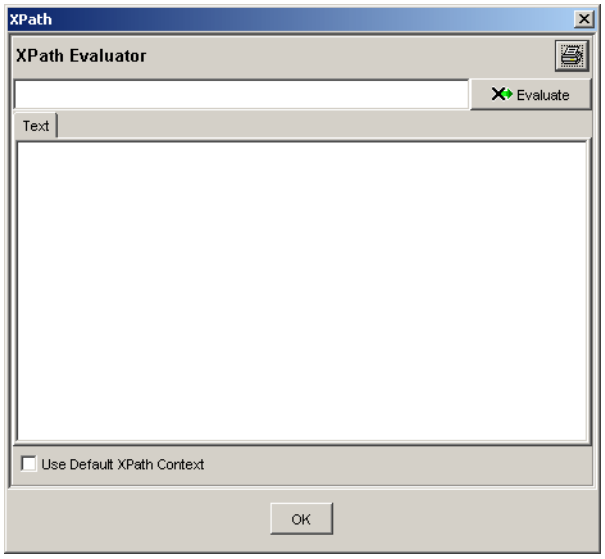


Figure 119. XPath Evaluator

2 Type the XPath expression into the field at the top of the dialog box, and click **Evaluate**. The result of the XPath expression is displayed as a string representation of the resulting data.

3 Click **OK** to close the XPath Evaluator.

Specifying Data Sources in XPath Expressions in the Structure Relationship Editor

Figure 120 shows an example of a data structure that might be seen in the **Source** pane of the Structure Relationship Editor. It represents a 12-month record of the consumption of electric power (in kilowatt-hours) by one utility customer.

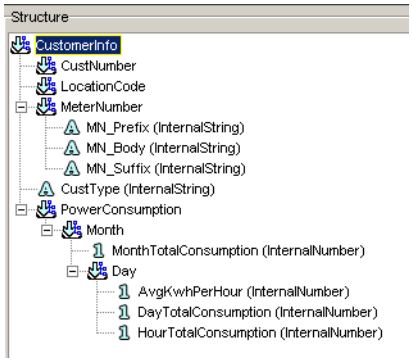


Figure 120. Example of a Data Source Structure in the Structure Relationship Editor

In Figure 120, **Month** is an array that repeats 12 times, and **Day** is an array. Valid references to sub-elements in the above structure include:

```
/*/CustNumber
/*/LocationCode
/*/MeterNumber
/*/MeterNumber/MN_Prefix
/*/MeterNumber/MN_Body
/*/MeterNumber/MN_Suffix
/*/CustType
/*/PowerConsumption
/*/PowerConsumption/Month[n]
/*/PowerConsumption/Month[n]/MonthTotalConsumption
/*/PowerConsumption/Month[n]/Day[n]/
/*/PowerConsumption/Month[n]/Day[n]/DayTotalConsumption
/*/PowerConsumption/Month[n]/Day[n]/HourTotalConsumption[n]
```

An asterisk (*) can be used as a “wildcard” in place of a node name to indicate “all” nodes at that level.

For example, continuing with Figure 120, `/*/*/MN_Prefix` refers to all sub-elements at a depth of three nodes that are named `MN_Prefix`.

In this case, `/*/*/MN_Prefix` refers to the same element as `/*/MeterNumber/MN_Prefix`.

But be careful. If there were another sub-element named `MN_Prefix` sitting under a different “ancestor” node, `/*/*/MN_Prefix` would include it as well.

You can only use the asterisk by itself as a node name; you cannot combine the asterisk with a partial node name.

For example, `/*/*/MN*/` and `/*/*/*_Suffix/` would be considered invalid by MapMaker.

A path that begins with a slash (/) represents an absolute path to an element.

A path that begins with two slashes (//) selects all elements in the document that fulfill the criteria, even if they are at different levels in the data structure.

XPath Expressions Not Reversible

In the Structure Relationship Editor, XPath expressions cannot be defined as reversible.

XPath Axes

XPath uses *axis* notation as a shorthand to specify a specific set of nodes. Axes can be used within XPath expressions. Figure 121 lists some of the XPath axes.

Axis	Description	Viewing Direction
`parent::`	Finds nodes that are immediately above the the context (“current”) node in the node tree.	Upwards
`child::`	Finds nodes that are immediately below the context node in the node tree.	Downward
`ancestor::`	Finds the parent node, grandparent node, great-grandparent node, and so on up the node tree.	Upward
`descendant::`	Finds the child nodes, grandchild nodes, great-grandchild nodes, and so on down the node tree.	Downward
`ancestor-or-self::`	Same as the `ancestor::` axis, but includes the context node.	Upward
`descendant-or-self`	Same as the `decendant::` axis, but includes the context node.	Downward
`preceding::`	Finds all nodes above the context node, excluding ancestors	Upward
`following::`	Finds all nodes below the context node, excluding descendants	Downward
`preceding-sibling`	Finds all nodes that precede the context node and have the same parent	Upward
`following-sibling`	Finds all nodes below the context node that share the same parent	Downward
//*	Finds all nodes in the current structure	—

Figure 121. A Partial Listing of XPath Axes

XPath Expressions and Changes to Business Entities

Any time the definition of a Business Entity (Global Variable, IBE, XML/XSD) is updated in MapMaker, the **Confirm Change** dialog box is invoked. The **Confirm Change** dialog box lists the effects of the change on all parts of the Map, including XPath expressions. You are given the option of confirming or cancelling the update.

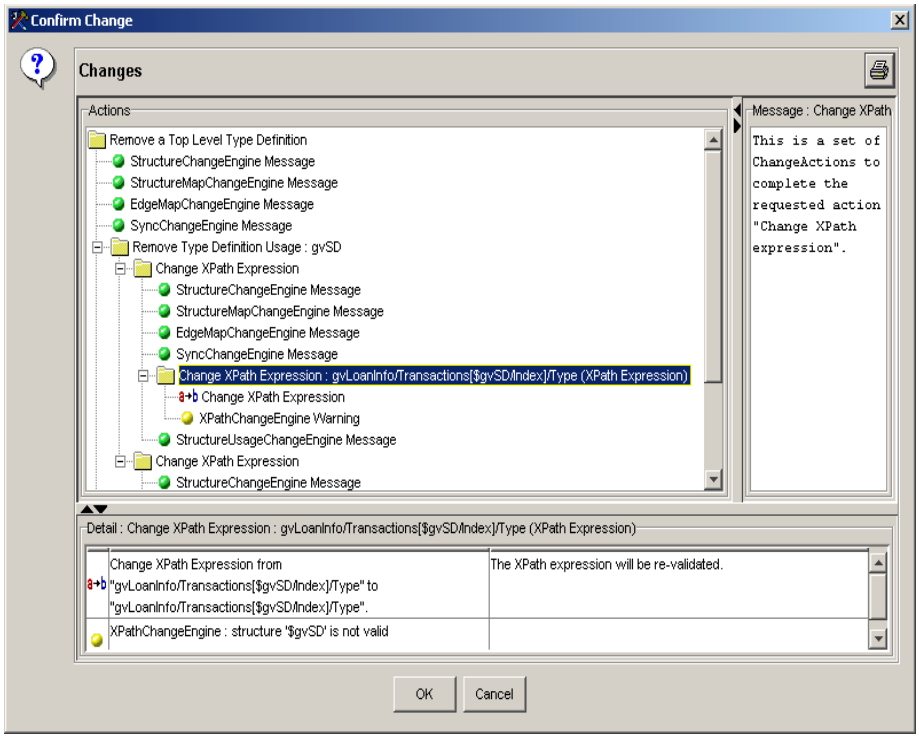


Figure 122. Confirm Change Dialog Box Informs of Effect on XPath Expressions

Additional XPath String Functions

In addition to the `substring-after` and `substring-before` functions described above, there are many other functions available in XPath. For your convenience, we describe a few of them below. This is not intended to be an exhaustive XPath reference. Be aware that in XPath, for those functions that address bytes in a string, the first position in a string is numbered "1", not "0".

String Function Prototypes	Return Type	Description
`string(argument)`	string	Returns the value of `argument` as a string; `argument` can be of any data type.
`concat(string1,string2, ...)`	string	The string arguments are converted into a single string.
`substring(string,number1, number2)`	string	Returns the characters found at `string` position `number1` for a length of `number2` characters.
`string-length(string)`	number	Returns the number of characters in `string`.
`normalize-space(string)`	string	Removes leading and trailing spaces, and converts multiple spaces to single spaces.

Figure 123. Selected XPath "String" Functions

`string(argument1)`
If the argument is a number, its value is returned as a character string. If the argument is boolean, the string "`true`" or "`false`" is returned.

`concat(string1,string2,...)`
Example: `concat(/time/hours,":",/time/minutes)`

`substring(string,number1,number2)`
The first byte in a string is byte 1.
Example: For an element `FirstName` with value `"Tanushri"`, `substring(FirstName,4,1)` returns "`u`".

`string-length(string)`
Returns the number of characters in a string, including spaces.

`normalize-space(string)`
Example: " a b cde " is converted to *"a b cde"*

Numeric Operators in XPath Expressions

Basic arithmetic operations can be included as part of an XPath expression, as shown in the following table.

Operator	Description	Example
+	Add	`(/*/totalprice[1] + (/*/totalprice[2])`
-	Subtract	`(/*/totalprice[1] - (/*/totalprice[2])`
*	Multiply	`(/*/minutes) * 60`
div	Divide	`(/*/seconds) div 60`
mod	Divide and return the remainder	`(/*/months) mod 12`

Figure 124. Numeric Operators in XPath Expressions

Indexing in XPath Expressions

If you have a business element that is defined as an array, you can use indexing in an XPath expression to access a particular element in the array. You express the "element number" in square brackets ([ ]) following the node name of the element in question. In an XPath expression, the first element in an array is element [1], not element [0].

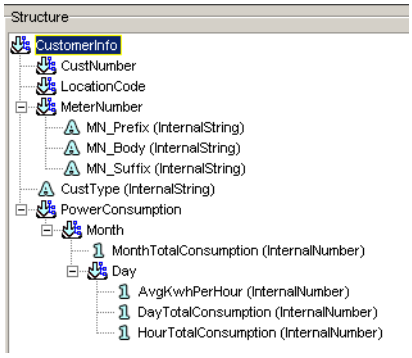


Figure 125. Example of a Data Source Structure in the Structure Relationship Editor

Example 10 - 3

In Figure 125, the sub-elements **Month**, **Day**, and **HourTotalConsumption** are arrays. To access the **MonthTotalConsumption** figure for the third month in the **Month** array, the XPath syntax is:

`"/*/PowerConsumption/Month[3]/MonthTotalConsumption".`

If you do not remember whether a particular sub-element is defined as an array or not, stand over the sub-element in question for a moment and a pop-up window displays the characteristics of the sub-element, including whether or not it is an array. See Figure 126.

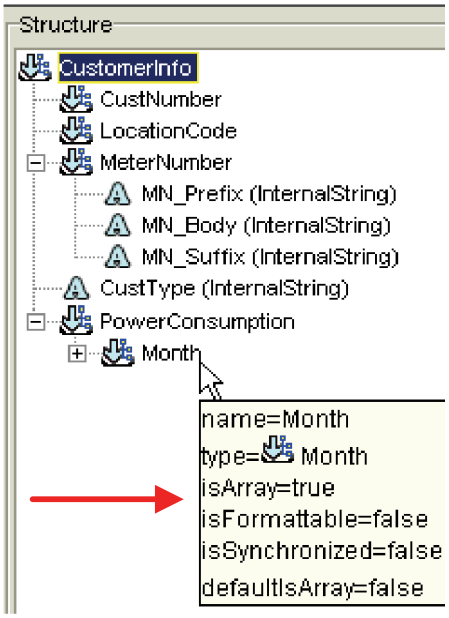


Figure 126. Cursor Over a Sub-element to Determine if it is an Array

To determine the number of entries in an array, use the `last()` function. This function returns a number that is the number of elements in the array.

Example 10 - 4

To access the **Month** structure in the last row of the array, the XPath syntax is:

```
"/*/PowerConsumption/Month[last()]"
```

Example 10 - 5

To compute the average kWh used per day for month three in the structure in Figure 125, the XPath expression is:

```
/*/*/Month[3]/MonthTotalConsumption  div
/*/*/Month[3]/Day[last()]
```

Data Mapping

Data mapping is a means for a method to transfer data from one data type to another. Within the flow of a method, there are two or more instances where data mapping takes place. Data mapping is defined in MapMaker's **Data Mapping Editor**.

Data mapping is performed between three data type structures:

- A **Source** data type - A data structure that receives input from an external source. This is the data output from a method step.
- A **Variable** - used to store the data.
- A **Target** data type - A data structure that sends output to an external source. This is the data input to a method step.

Chronological Order of Events in a Data Mapping Instance

During runtime, a data mapping instance is performed in the following order:

1. Data is transferred from the source data type (if one exists) to a Variable.
2. Data is transferred (if mapping for this exists) between Variables.

   Data transfer from one Variable to another is defined in the **Variables** tab of the Data Mapping Editor.
3. Data is transferred from a Variable to the target data type (if one exists).

Different Types of Data Mapping Instances

A data mapping instance can be one of the following:

- From a **Source** data type to **Variables**.
- From **Variables** to a **Target** data type (if the step is a consumer of data).
- From a **Source** data type to a **Target** data type.

Data Mapping Location in a Method

In the flow of a method, Data Mapping is performed right after the Start step (the first step of a method) and right before the Stop step (the last step of a method), as illustrated in Figure 127.

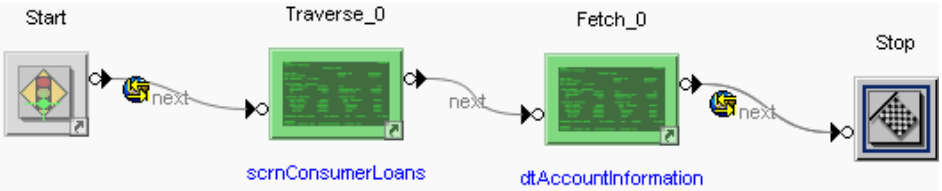


Figure 127. Data Mapping Before the First Method Step and After the Last Method Step

This is because after the Start step, the method receives input from an external source; and before the Stop step, the method sends output to an external source. You may also map data in the middle of a method, for example, after a Fetch step or before a Write step.

In the **Method** tab, a data mapping instance is represented by a colored ball on the link between two method steps (Figure 128):

Figure 128. A data Mapping Instance

The First Data Mapping Instance

The first data mapping does the following:

- Maps the data from the method's input variable(s) to a corresponding variable.
- Distributes the received data from the input variable(s) into variables, which are used later on in the method flow.

The Last Data Mapping Instance

The last data mapping does the following:

- Maps the data from one or more Legacy screens to one or more corresponding output variables.
- Maps the Variables into output variables.

A Data Mapping Instance in the Middle of a Method

Data mapping can take place in the middle of a method in the following cases:

- Before a Write step — here you must map data from a Variable to a Legacy XBE, so that data can be written into the host application's current screen.
- After a Fetch step — here you must map data from a Legacy XBE to a Variable, so that data, retrieved from the host application's current screen, can be stored.
- Any other place in the method — you can use data mapping instead of a Set step, to transfer data from one Variable to another.
- "External" steps that return data (InternalInvoke, ExternalInvoke and JClient3).

The Data Mapping Editor

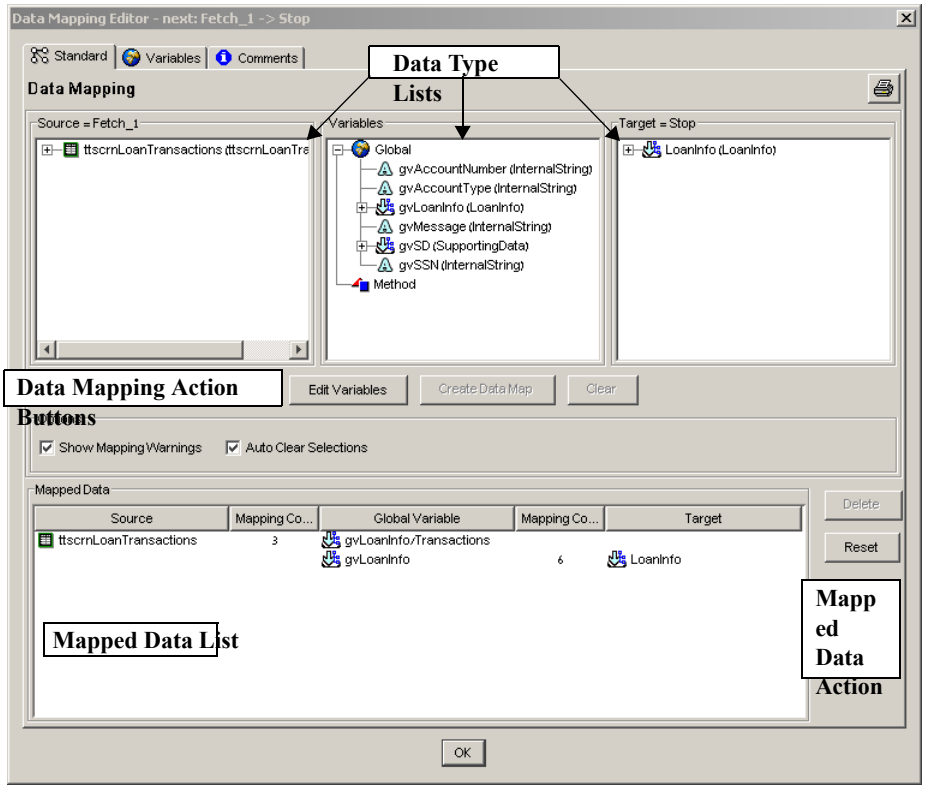


Figure 129. The Data Mapping Editor

The Data Mapping Editor (Figure 129) is used to instruct a method to transfer data from one data structure to another. This window provides the following options:

Option	Description
Print ( )	Prints all or part of the **Data Mapping Editor**'s display.
Edit Variables	Opens the **Edit Variables** dialog box where you can edit and add new Method and Global Variables to the **Variables** list.

Option	Description
Create Data Map	Creates a data map between the data types selected in each list. **Note:** This button is enabled only when the conversion between the source and target data types is allowed by the conversion rules (see "Data Type Conversion Rules" on page 252).
Clear	Clears the current selections in the list boxes.
Show Mapping Warnings	Specifies whether or not to issue a warning during data mapping, if a conversion may be defined, but may fail at runtime.
Auto Clear Selections	Specifies whether or not to automatically deselect all of the highlighted items once a mapping is made (i.e. the **Create Data Map** button is clicked).
Delete	Deletes the selected data mapping entry from the **Mapped Data** list.
Reset	Reverts to the last save.

Working with the Data Mapping Editor

Launching the Data Mapping Editor

Data mapping takes place between two method steps. Whenever you open the Data Mapping Editor, the information displayed is specific to a position in the method flow between two method steps.

To open the Data Mapping Editor:

1. In MapMaker's **Methods** tab, display the method you wish to edit and select the link between the two steps where data mapping is to be performed.
2. Do one of the following:
   - Double-click on the link, *Or*
   - Right-click on the link and select **Data Mapping** from the shortcut menu.

Mapping a Source Data Type to a Variable

If the data mapping instance is located right after input is received from an external source, the data type that represents the input must be mapped to a Variable for storage. This takes place either after the Start step (the first step), which transfers data from the external representation to the internal representation, or after a Fetch step.

To map a Source Data Type to a Variable:

1. In the **Standard** tab, select the data type that represents the input from the **Source** list.
2. In the **Variables** list, select the corresponding Variable to which you wish to map the input data.
3. Click **Create Data Map**.

   A record of this is displayed in the **Mapped Data** list.

Mapping One Variable to Another

To map one Variable to another:

1. In the **Variables** tab, select the two Variables you wish to map to each other in the **Source** and **Target** lists.
2. Click **Create Data Map**.

> **Note:** Parent elements can be mapped to each other (using **Create Data Map**) only after the relationship between their respective data types has been defined in the Structure Relationship Editor.

Mapping a Variable to a Target Data Type

If the data mapping instance is located right before output is sent to an external source, a Variable that stores the data must be mapped to the data type that represents the output. This takes place either before the method's Stop step (the last step), or before a Write step.

To map a Variable to a target data type:

1. In the **Variables** list, select the Variable from which you wish to map the output data. The Variable must correspond to the **Target** data type.
2. In the **Standard** tab, select the data type that represents the output from the **Target** list.
3. Click **Create Data Map**.

   A record of this is displayed in the **Mapped Data** list.

Mapping a Source Data Type to a Target Data Type via a Variable

If a data mapping instance is located right after input is received from an external source, and also right before output is sent to an external data source, the data type that represents the input must be mapped to the data type that represents the output through a Variable. This can happen, for instance, if the last method step before the Stop step is a Fetch step. In that case, the **Data Mapping Editor** displays a Legacy Screen XBE data type in the **Source** list, and an output variable IBE data type in the **Target** list.

To map a source data type to a target data type through a Variable:

> **Note:** This procedure consists of two data transfers:
> 1. From the source data type to Variables.
> 2. From Variables to the target data type.

1. In the **Standard** tab, select the data type that represents the input from the **Source** list.
2. In the **Variables** list, select the Variable that will carry the data.
3. In the **Target** list, select the data type that represents the output.
4. Click **Create Data Map**.

> **Note:** This button is enabled only when the conversion between the source and target data types is allowed by the conversion rules (see "Data Type Conversion Rules" on page 252)

A record of this is displayed in the **Mapped Data** list (Figure 130).

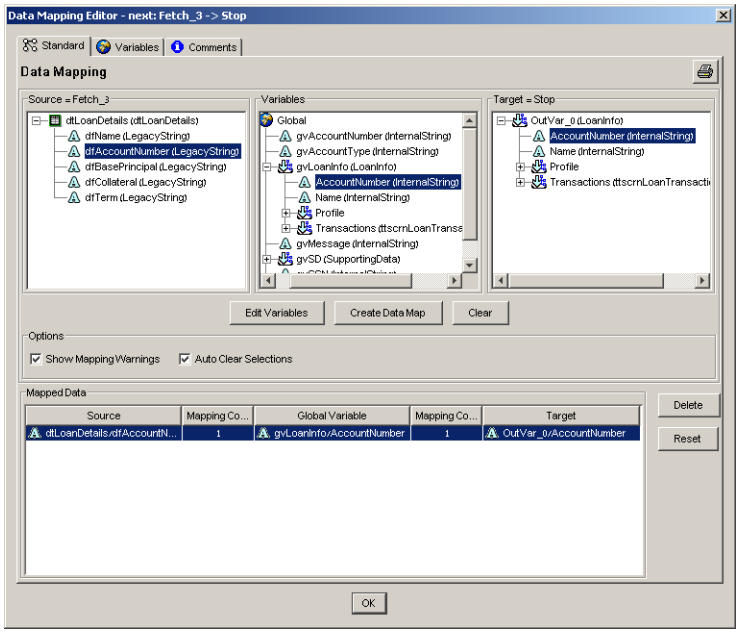


Figure 130. Data Mapping: Source Through Variable to Target Data Types

Suggested Workflow for Creating a Method

To create a method that utilizes webMethods JI Data Modeling, use the following workflow:

1. Create XBEs. These include:
   - Legacy Screen XBE(s), for writing data into legacy screens. This is done in MapMaker's **Data** tab, by creating Data Templates.
   - Legacy Screen XBE(s), for retrieving data from legacy screens. This is done in MapMaker's **Data** tab, by creating Data Templates.
   - An XSD or MLM type XBE for method input. This is done in the Business Entity Editor.
   - An XSD or MLM type XBE for method output. This is done in the Business Entity Editor.
2. Create IBEs that correspond to the XBEs and have enough fields to carry the data you want. This is done by copying the XBEs to IBEs in the Business Entity Editor's **External Data Definitions** tabs.

   Repeat this step for each of the XBEs.
3. Create Global Variables of the IBE data types. This is done by adding new, IBE Global Variables in the Business Entity Editor's **Global Variables** tab.
4. Define any necessary relationship between data structures. This is done in the Structure Relationship Editor.
5. Design the method in MapMaker's Method Editor. This includes:

- Creating Method Variable(s). Method Variables apply to specific methods, and are added in the MapMaker **Methods** tab.
- Defining input variable(s). An input variable requires that you provide both an internal data type and a corresponding external data type.
- Defining output variable(s). An output variable requires you to provide both an internal data type and a corresponding external data type.
- Establish the general method flow by creating method steps and linking them to each other.
- Map the data after the Start step. To launch the Data Mapping Editor, double-click the link between the Start step and the first Traverse step.
- Map the data before the Stop step. To enter the Data Mapping Editor, double-click on the link between the last method step and the Stop step.
- Perform any additional data mappings if necessary.

Chapter 8. Methods

A method is a logical grouping of screen transitions, navigational information and input and output variables. After methods are defined, they are grouped together into one or more services that can execute the methods.

A method can be called by a webMethods JI client to perform a specific transaction against the host.

This section describes:

- "The Methods Tab" on page 275
- "Managing Methods" on page 277
- "Inputs" on page 280
- "Method Variables" on page 282
- "Method Steps" on page 288
- "Method Step Links" on page 355
- "Outputs" on page 360
- "Methods Presentation View Color Code" on page 361
- "Keyboard Shortcuts and Search Options" on page 361

The Methods Tab

The **Methods** tab of the Tree view displays all methods defined for the selected map, as well as the inputs, steps, and output variables included in each method.

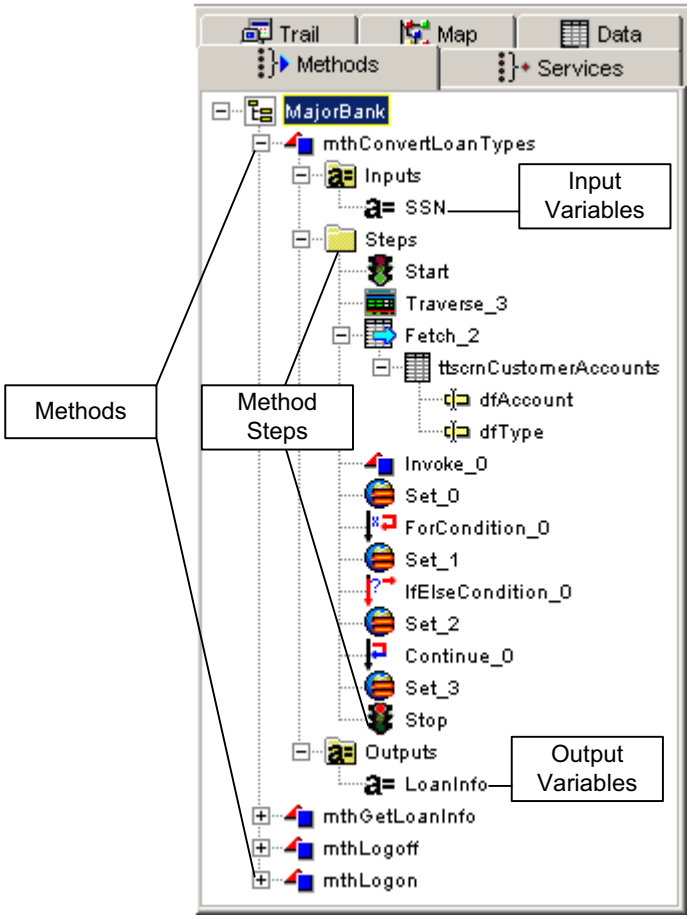


Figure 131. The Methods Tab

The **Methods** tab provides the following:

Component	Description
Method Name	The name of each method.
Inputs	Displays all inputs associated with that specific method. For more information about inputs, see “Inputs” on page 280.
Steps	Displays the method steps included in the method. For information on method steps, see “Method Steps” on page 288.

Component	Description
Outputs	Displays all output variables associated with this method. For information about output variables, see “Outputs” on page 360.

Managing Methods

This section describes:

- “Adding Methods” on page 277
- “Duplicating Methods” on page 278
- “Deleting Methods” on page 278
- “Configuring Method Properties” on page 278

Adding Methods

To add a method to a map, select the top-level node ( ) in the **Methods** tab of the Tree view, right-click and select **Add Method** from the shortcut menu.

A new method is added to the tree, consisting of two default steps (Figure 132):

- **Start** — begins the transaction performed by this method.
- **Stop** — ends the transaction performed by this method.

The Start and Stop steps include the option to clear the Global Variables.

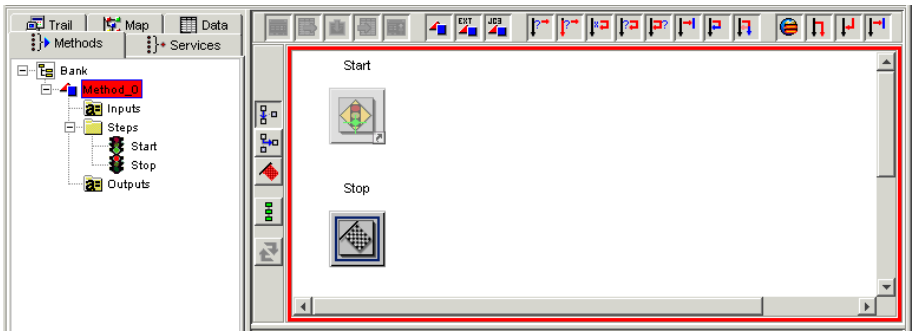


Figure 132. Adding a New Method

After the method has been added, change its name by right-clicking on the method name, selecting **Rename** from the shortcut menu and entering the new name.

Duplicating Methods

To duplicate a method, right-click the method component in the **Methods** tab of the Tree view, and select **Duplicate** from the shortcut menu. The selected method is copied into the list with a unique new name. The default name of a duplicate method consists of:

- The source method's name, followed by an underscore and a serial number ranging from 1 to n (e.g. `GetLoanInfo_1`).
- If more than one method has been copied from the same source method, an integer is added to make the name unique (e.g. `GetLoanInfo_3`).

All properties of the source method are duplicated and assigned to the new method. This includes:

- All steps, which follow the same naming scheme described above for the duplicate method.
- All edges, including data mapping.
- Input, Output and Method variables.

Changes to either the source method or the new method are not propagated to the other. They are not "synchronized". Input, Output and Method variable names do not change.

After the duplicate method has been added, change its name by right-clicking on the method name, selecting **Rename** from the shortcut menu and entering the new name.

Deleting Methods

To delete a method from a map, select the method component in the **Methods** tab of the Tree view. With the method component highlighted, right-click and select **Delete** from the shortcut menu.

Configuring Method Properties

To configure the properties of a method, select it in the Tree view. The method's properties are displayed in the Properties view.

Method Properties > General tab

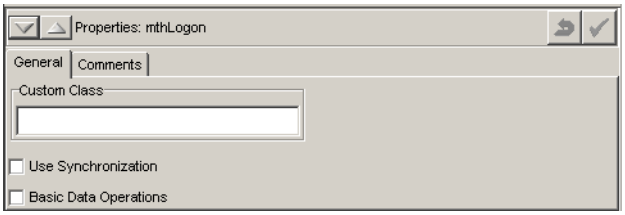


Figure 133. Method Properties View > General Tab

The **General** tab of the method Properties view provides the following options:

Option	Description
Custom Class	The name of custom class that should be used with this method. For more information about custom classes, see "Custom Classes" on page 112. **Note:** If a custom class is created for the method, the name of the custom method class, rather than the name of the method as defined in MapMaker, should be used by clients when invoking the method.
Use Synchronization	Synchronizes objects, during modification, in the return map.
Basic Data Operations	Enables the method's version 3.x Compatibility Mode. This mode makes it possible to use version 3.x objects in a current version 4.x Map, by disabling version 4.x-specific data mapping-related fields, and allowing the 3.x method and its objects to use simple data sources. **Note:** The **Basic Data Operations** checkbox is disabled for methods that contain Method Variables. This is because Method Variables are not 3.x compatible.

Inputs

The webMethods JI services require a mechanism for passing information into legacy application fields (e.g. an account number). This mechanism is provided by Global Variables, which are associated with actions. Global Variables are defined in the **Business Entity Editor** dialog box (see "Adding Business Entities" on page 222).

The variable information contained in Global Variables can be updated in one of the following ways:

- Based on the value that is fetched from a data field (see "Adding Data Fields to a Data Template" on page 186).
- Using an input, which allows input from the client to be used to update the value of a Global Variable.
- Using External steps (InternalInvoke, ExternalInvoke and JClient3) and Set steps.

It is possible to use both the first and second ways listed above, so that the value of a Global Variable is set using the value of a data field, and this can be optionally overridden by the client by sending a different value in an input.

The sequence of events for using inputs is as follows:

1. The values of a Global Variable may be set in advance of the method invocation, using the value of a data field (or by using custom code).
2. Next, when the client invokes the method, the client may pass in values for inputs. If the client uses invalid input names, they are ignored. Values for valid input names are used to update the value of the specified Global Variable.
3. One of the following steps take place:
   - The Traverse step traverses to the specified screen, *Or*
   - The Perform step executes its specified action.
4. The specified action retrieves Global Variable information to input into the field on the legacy screen.
5. The action inputs the data from the Global Variable into the field of the legacy screen and executes the action's AID key to complete the transaction.

Adding Inputs

To add an input, select the **Inputs** node in the Tree view ( ), right-click and select **Add Input** from the shortcut menu. After the input has been added, change its name by right-clicking on the input, selecting **Rename** from the shortcut menu and entering the new name.

Setting Input Properties

To configure the properties of an input, select it in the Tree view. The input properties are displayed in the Properties view (Figure 134).

Input Properties > General Tab

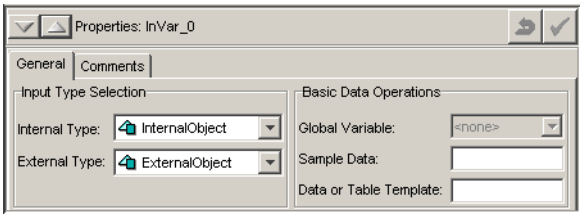


Figure 134. Input Properties View > General Tab

The **General** tab of the input Properties provides the following options:

Option	Description
Input Type Selection	A version 4.x input is associated with two data types: • Internal Type — Choose from a list of IBEs. • External Type — Choose from a list of XBEs Choose the corresponding input types from the **Internal Type** and **External Type** drop-down lists.

Option	Description
Basic Data Operations	A version 3.x input does not support version 4.x complex data structures. Instead, such an input is defined using a simple data source, i.e. a **Global Variable.** • Global Variable — a Global Variable whose type is **Internal Object** that is associated with this input. Choose the appropriate Global Variable from the drop-down list. Default: <none> • Sample Data — example data provided for client testing purposes. • Data or Table Template — the Data Template or Table Template that the data passed in should be associated with for Input Metadata. At runtime, the Metadata for the method can be queried, and this provides the client with a description of the expected input format for the data passed in to that Input variable. This field is enabled for 3.5 compatible methods only.

Enabling and Disabling Inputs

Inputs can be disabled so that a service that uses this method does not include the input in its generated code. To disable or enable an input, select it, right-click and select the appropriate option (**Disable Input** or **Enable Input**, respectively).

Additionally, all of the method's inputs can be disabled or enabled, by selecting the Inputs node in the Tree view, right-clicking, and selecting **Disable All Inputs** or **Enable All Inputs**, respectively.

Method Variables

Method variables store data being transferred between the client and the legacy screen in a webMethods JI service. However, unlike Global variables, Method variables apply only to the specific method in which they are defined, and are not available for inclusion in other methods throughout a Map. This prevents Map-wide data from overriding method-specific data, and vice versa. For more information about Global Variables, see "Global Variables" on page 150 and "Internal Business Entities" on page 204.

> **Note:** Method Variables cannot be added to 3.5 compatibility mode methods, that is, methods for which the **Basic Data Operations** checkbox is set.

Adding a Method Variable to a Method

A Method Variable is added to a method in either of the following ways:

- Using the **Method** node of the Tree view — right-click the **Method** node in the tree and select **Edit Method Variables** from the shortcut menu.

  The **Edit Method Variables** dialog box is displayed, as shown in Figure 135.
- Using the Data Mapping Editor — click the **Edit Variables** button.

  The **Edit Variables** dialog box is displayed, containing the same information as the **Edit Method Variables** dialog box in the **Method** tab.

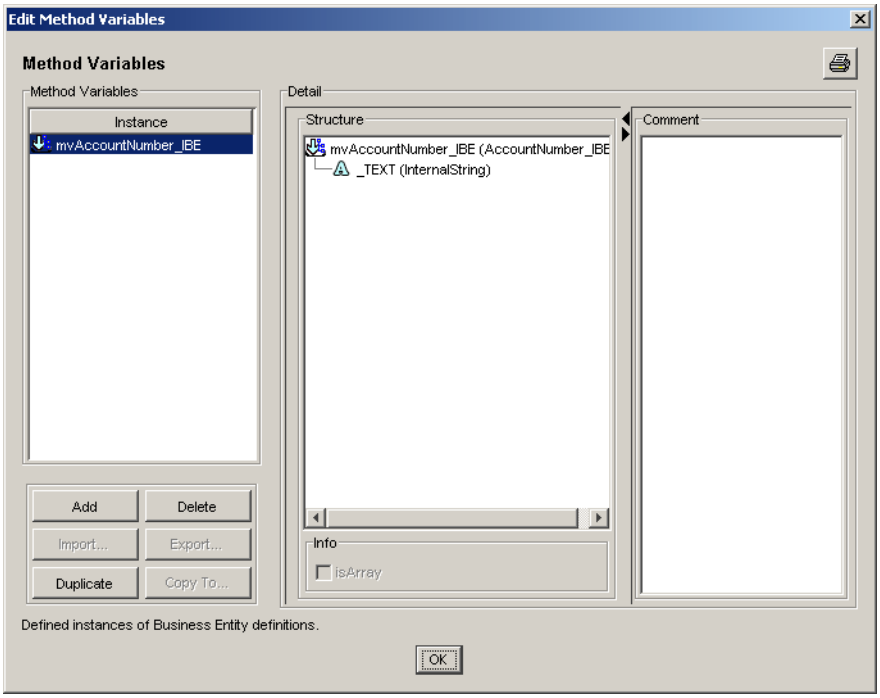


Figure 135. The Edit Method Variables Dialog Box

The **Edit Method Variables** dialog box enables you to add, delete and duplicate Method Variables, and shares the same structure as the Business Entity Editor. For more information about the Business Entity Editor, see "The Business Entity Editor" on page 210.

The **Edit Variables** dialog box provides the following options:

Option	Description
Instance pane	Lists all Method variable instances of Business Entities for the selected method. Method variables are managed using the following tools: • The data type action buttons. • A shortcut menu, allowing you to either rename or delete the selected Method variable. Selecting a Method variable in the **Instance** pane displays its hierarchical structure in the **Structure** pane, allowing you to change its type or edit its default data.
Structure pane	Shows the hierarchical structure of the Method variable selected in the **Instance** pane.
Data Type action buttons	• **Add**: Adds a new Method variable. For more information, see "Adding Method Variables" on page 286. • **Delete**: Deletes the selected Method variable. For more information, see "Deleting Method Variables" on page 287. • **Duplicate**: Duplicates the selected Method variable and assigns it a new name. For more information, see "Duplicating Method Variables" on page 287. **Note:** The **Import**, **Export** and **Copy To** buttons apply only to Business Entities. For this reason they are disabled in the **Edit Variables** dialog box.

Option	Description
	• **Print** ( ): Launches the **Print** dialog box, allowing you to print all or part of the Edit Variables dialog box's current tab. For information on the **Print** dialog box, see "Printing the Display" on page 109.
Information pane	• **isArray**: Indicates that the variable in question is a repeating data type. For information on array data types, see "Indicating a Data Type is an Array" on page 232.
Comment	Space for any type of written comment. When the selected data type has been copied from another data type, this pane automatically indicates the name of the source data type.

The Edit Variables Dialog Box Shortcut Menus

The **Edit Variables** dialog box allows you to manage Global and Method variables using shortcut menus. These menus are available by right clicking the object of interest in either the **Instance** pane or the **Structure** pane. The **Structure** pane includes a shortcut menu for the structure's root.

Since Variables are only instances of other Business Entities, they cannot be edited and all changes must be made directly to the source Business Entity.

The Instance Pane's Shortcut Menu

To either rename or delete a Method Variable, right-click it in the **Instance** pane and select the appropriate command from the shortcut menu.

The Structure Pane's Shortcut Menu

The Structure pane's shortcut menu provides the following options:

Option	Description
Change Type	Allows you to change the selected Method variable's type to another type.
Edit Default Data	The **Default Value** must conform to one of the following formats of the data type in question: 1) The input format(s) 2) The default format (if the data type has no input format). For example, a non-structured global variable whose type is InternalDate has no input formats, so its 'Default Value' is validated using the InternalDate's default format. If the value does not conform to the relevant format, a warning message is displayed.

Managing Method Variables

This section describes how to use Method variables in MapMaker. It contains the following subsections:

- "Adding Method Variables" on page 286
- "Deleting Method Variables" on page 287
- "Duplicating Method Variables" on page 287

Adding Method Variables

To add a new Method variable:

1 In the **Method** tab, click **Add**.
The **Choose a Type** dialog box opens (Figure 136).

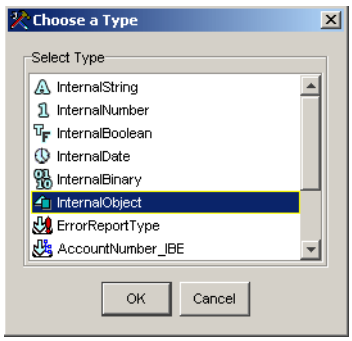


Figure 136. The Choose a Type Dialog Box

2 Select the data type to be assigned to the new Method Variable.

Available options are standard data types, such as InternalString, or previously-defined IBEs, such as **AccountNumber_IBE**.

3 Click **OK**.

The new Method Variable is added to the **Instance** list. Its default name consists of the prefix `mv` followed by the name of the data type it was copied from. If more than one Method Variable has been copied from that data type, an integer is added to make the name unique (e.g. `mvString_3`).

Deleting Method Variables

To delete a Method variable:

Select the Method Variable you wish to delete and click **Delete**.

The Method Variable is removed from the list.

Duplicating Method Variables

To duplicate a Method variable:

1 Select the Method variable you wish to duplicate and click **Duplicate**.

The selected Method variable is copied into the list with a unique new name.

The default name of a copied Method variable consists of the prefix `mv` followed by the name of the Method variable from which it was copied. An integer is added to make the name unique (e.g. `mvString_0`).

2 Rename the new Method variable in one of the following ways:

- Double-click on it in the **Instance** list.
- Right-click on it and select **Rename**.
- Select it and press the **F2** key.

3 Type the new name, and then apply it by pressing **Enter** or selecting another item in the list.

Method Steps

Method steps represent the actual activity performed by the method, such as screen traversal, inputting variable data into fields, and extracting returned data from fields.

When you add a new method, it automatically includes two default steps: Start and Stop (see "Start and Stop Steps" on page 292). In between them, you can add as many steps as needed to perform the transaction in question. To add a step, choose one of the following:

- Right click the Steps node in the Tree view and choose the appropriate step.
- Click the relevant tool bar button.
- Drag the appropriate step's icon from the tool bar to the view.

The available steps and their corresponding tool bar buttons are listed in the following table:.

Button	Description
	Traverse: Instructs the method to traverse from the current screen to another screen. For more information on Traverse steps, see "Traverse Steps" on page 300.
	Fetch: Extracts data from the screen, based on the specified data template or table template. For more information on Fetch steps, see "Fetch Steps" on page 304.
	Perform: Instructs the method to perform a specified action. For more information on Perform steps, see "Perform Steps" on page 308.
	Write: Inputs data to the screen, based on the specified data template or table template. For more information on Write steps, see "Write Steps" on page 311.
	UserInteraction: Allows users to navigate the host application manually. A read/write-enabled PATerm or JTerm window must be open on the client's display for this step to successfully function at runtime. For more information on UserInteraction steps, see "UserInteraction Steps" on page 313. **Note:** UserInteraction steps are not supported when the webMethods JI Service is deployed as a Resource Adapter.

Button	Description
	Internal Invoke: Calls another method residing in the same map. The two methods can use the same Variables and no network operations are required. For more information on Internal Invoke steps, see "Internal Invoke Steps" on page 315.
	External Invoke: Calls another method residing in a different map. The method being invoked shares the same host connection as the calling method. This step provides multi-developer support, allowing different developers to work on different parts of the same legacy system. An External Invoke step requires passing input and output parameters to the external method, using Data Mapping. No network operations are required. For more information on External Invoke Steps steps, see "External Invoke Steps" on page 316.
	JClient3: Calls another method from other webMethods JI services, using JClient3 API calls. This step allows you to access multiple, parallel legacy applications. Since the invoked methods belong to a different map, data mapping operations are required. In addition, since the invoked method is located on a different virtual machine, network operations are required as well. For more information on JClient3 Steps, see "JClient3 Steps" on page 329.
	IfCondition: Provides a comparison between two objects. • If the evaluation of the Condition is "true", the step(s) on the conditional or **branch** link are executed. • If the evaluation of the Condition is "false", the primary or **next** method logic path continues. If the true/conditional step(s) are executed, the primary or **next** method logic path resumes once they have completed. For more information on conditional steps, see "Conditional Steps" on page 333.

Button	Description
	IfElseCondition: Provides a comparison between two objects. • If the evaluation of the Condition is “true”, the step(s) on the conditional or **branch** link are executed. • If the evaluation of the Condition is “false”, i.e. the “Else” part of the condition, the primary or **next** method logic path continues. If the true/conditional step(s) are executed, the primary or **next** method logic path is not followed. For more information on conditional steps, see “Conditional Steps” on page 333.
	ForCondition: Executes the step(s) on the conditional or **branch** link, in a loop that is based on specified Initializer, Conditional, and Increment settings. When the For loop ends, the primary or **next** method logic path continues. For more information on conditional steps, see “Conditional Steps” on page 333.
	WhileCondition: Executes the step(s) on the conditional or **branch** link based on the specified conditional settings. When the While loop ends, the primary or **next** method logic path continues. For more information on conditional steps, see “Conditional Steps” on page 333.
	DoWhileCondition: Executes the step(s) on the conditional or **branch** link, based on the specified conditional settings. When the DoWhile loop ends, the primary or **next** method logic path continues. For more information on conditional steps, see “Conditional Steps” on page 333.
	EndIf: Indicates the completion of an IfCondition **branch** path. Once the EndIf step has been reached, the primary, **next** method path is resumed. For more information on conditional steps, see “Conditional Steps” on page 333.
	Continue: Indicates the completion of a ForCondition, a WhileCondition or a DoWhileCondition loop. The Continue step causes the next iteration of the loop to begin. If the loop condition is not satisfied, the primary, **next** method path is resumed. For more information on conditional steps, see “Conditional Steps” on page 333.

Button	Description
	Break: Breaks the ForCondition, WhileCondition or DoWhileCondition loop, goes back to the appropriate step (ForCondition, WhileCondition or DoWhileCondition) and then resumes the primary, **next** method path. For more information on conditional steps, see “Conditional Steps” on page 333.
	Set: Provides a mechanism for setting a Variable and/or adding data to the return map. For more information on Set steps, see “Set Steps” on page 345.
	Thread: Provides for parallel code execution within the method. When a Thread step is added to the method, a new branch is created representing a thread of parallel code execution. Each Thread step represents one additional thread of execution. For more information on Thread steps, see “Thread Steps” on page 350.
	ThreadJoin: Adds a ThreadJoin step to the selected method. A ThreadJoin step provides a method to ‘collect’ the threads within the method, and a point of convergence for the parallel code. For more information on Thread steps, see “Thread Steps” on page 350.
	EndThread: Indicates the completion of a Thread branch. The completed threads within the method are “collected” using a ThreadJoin step. For more information on Thread steps, see “Thread Steps” on page 350.

Method steps can be easily added and deleted using either the Tree view or the Presentation view. To configure the method step properties, select the step in either the Tree view or the Presentation view and then set its properties in the Properties view.

Adding Method Steps

A step can bee added to a method in any of the following ways:

- Using the Steps node of the Tree view — right-click the Steps node in the tree and select the method step to be added from the shortcut menu.

- Clicking the tool bar — left-click the desired step button in the tool bar.
- Dragging from the tool bar — left-click the desired step icon in the tool bar, hold the button down and drag the step to its appropriate place in the Presentation view.

Note: The Steps branch of the Tree view lists steps chronologically, in the order in which they were added, as opposed to the order in which they are executed. To determine the actual sequence of the steps, look at their graphical representation in the Presentation view.

When screen(s) are removed from the map, due to changes in the Trail and/or Map tabs, any method steps that rely on the removed screen(s) become invalid and must be corrected.

Deleting Method Steps

To delete a step from a method, select the step in the Tree view or in the Presentation view, right click and select **Delete** from the shortcut menu.

Configuring Method Step Properties

This section describes:

- "Start and Stop Steps" on page 292
- "Screen-Based Steps" on page 297
- "Invoke Steps" on page 315
- "Conditional Steps" on page 333
- "Set Steps" on page 345
- "Thread Steps" on page 350

Start and Stop Steps

Every method includes two default steps (Figure 137):

- **Start** — begins the transaction performed by this method.
- **Stop** — ends the transaction performed by this method.

Note: Start and Stop are mandatory steps, which are automatically included in every method (as opposed to other types of steps, which are manually added to the method) and cannot be deleted.

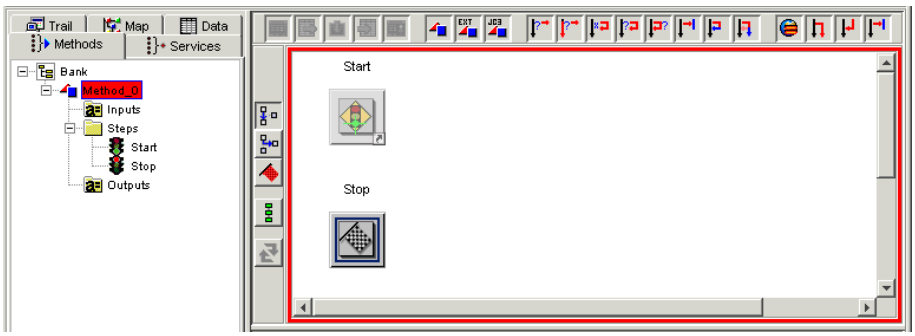


Figure 137. Adding a New Method

In between these two steps, you can add as many steps as needed to perform the transaction in question.

Start Step

The Start step is the entry point to the method. This is the step in which data are transferred from the input variables to the Variables, to be used by the method on the **next** edge.

To define the Start step's settings, select it in the Presentation view. The step's properties are displayed in the Properties view.

Start Step Properties > General Tab

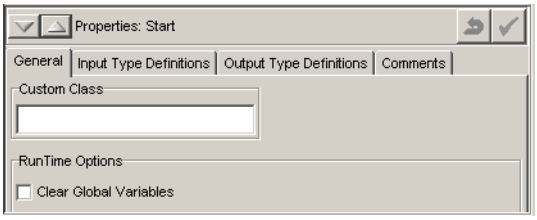


Figure 138. Start Step Properties > General Tab

The **General** tab of the Start step's Properties view (Figure 138) provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.

Option	Description
Clear Global Variables	At runtime, this clears all data from the Global Variables before any edge mappings are done.

Start Step Properties > Input Type Definitions Tab

Note: This description applies to the **Input Type Definitions** tabs of all steps.

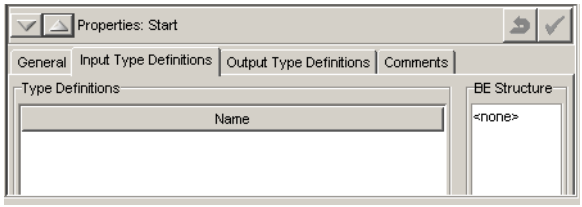


Figure 139. Start Step Properties > Input Type Definitions Tab

The **Input Type Definitions** tab of the Start step's Properties view (Figure 139) provides the following options:

Option	Description
Name	Lists the names of the Business Entities that are mapped in the Data Mapping Editor as the targets of the preceding link.
BE Structure	Displays the structure of the Business Entity selected in the **Name** list on the left.

Start Step Properties > Output Type Definitions Tab

Note: This description applies to the **Output Type Definitions** tabs of all steps.

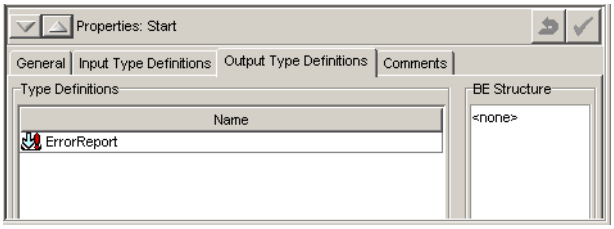


Figure 140. Start Step Properties > Output Type Definitions tab

The **Output Type Definitions** tab of the Start step Properties view (Figure 140) provides the following options:

Option	Description
Name	Lists the names of the Business Entities that are mapped in the Data Mapping Editor as the source of the succeeding link. By default, this list includes a system-level Internal Business Entity named **ErrorReport**. This data type is always available on **onFail** links and is automatically populated when an error occurs within a method.
BE Structure	Displays the structure of the Business Entity selected in the **Name** list.

Start Step Properties > Comments tab

Note: This description applies to the **Comments** tabs of all steps.

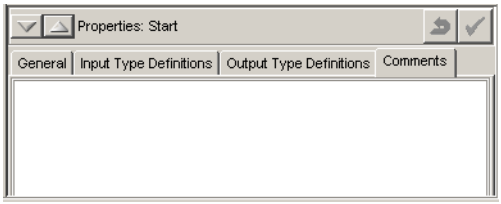


Figure 141. Start Step Properties > Comments Tab

The **Comments** tab of the Start step's Properties view (Figure 141) allows you to enter any free text related to this step.

Stop Step

The Stop step is the exit point from the method. This is the step in which data are transferred from the Variables to the output variables, to be sent back to the client (or to another method acting as a client, i.e. a method that uses an Internal, External, or JClient3 step).

To define the Stop step's settings, select it in the Presentation view. The step's properties are displayed in the Properties view.

Stop Step Properties > General Tab

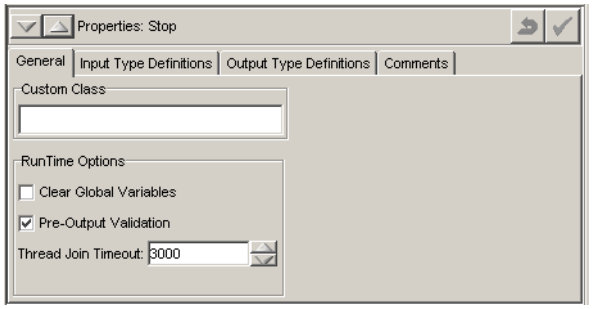


Figure 142. Stop Step Properties > General Tab

The **General** tab of the Stop step's Properties view (Figure 142) provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.

Option	Description
RunTime Options	• **Clear Global Variables**: Clears all data from the Global Variables after any edge mappings are done. • **Pre-Output Validation**: Validates particles on the output XML, to verify minimum and maximum occurrences (see "Editing an XML/XSD Particle" on page 244). If this option fails, the step's **onFail** link is invoked. If no **onFail** processing is defined, an exception is propagated back to the client. • **Thread Join Timeout**: At runtime, the Stop step joins any non-persistent Thread steps that are still running in the method. This option defines how long (in milliseconds) to wait for these threads to be joined before the Thread is stopped.

Stop Step Properties > Input Type Definitions, Output Type Definitions and Comments tabs

These tabs are identical to the Start step's corresponding tabs. For details, see the following:

- "Start Step Properties > Input Type Definitions Tab" on page 294
- "Start Step Properties > Output Type Definitions Tab" on page 294
- "Start Step Properties > Comments tab" on page 295

Screen-Based Steps

This section describes:

- "Traverse Steps" on page 300
- "Fetch Steps" on page 304
- "Perform Steps" on page 308
- "Write Steps" on page 311
- "UserInteraction Steps" on page 313

Screen-based method steps are steps that require a "current screen" context, i.e. the context of the screen that is currently shown in the Presentation view. You can use the space bar to toggle between the screen's displays in the **Methods** tab and in the **Map** tab.

This group of steps includes the Traverse, Fetch, Perform, Write and UserInteraction steps. The first screen-based step in a method, or the first screen-based step after a conditional step, must be a Traverse or a Perform step. Screen-based steps are not permitted on Thread branches, due to the possibility of concurrent host interaction from multiple threads of execution.

In addition to the standard onFail processing, version 4.x screen-based steps feature the following tools for recovery from unmapped destination screens:

- "Validating the Current Screen" on page 298
- "Recovery Action" on page 299

Validating the Current Screen

All screen-based steps, except for UserInteraction, allow you to verify that the proper screen has been reached before executing the step. This validation is configured in the **General** tab of the step's Properties pane, by setting the **Validate Current Screen** check box.

> **Note:** The **Validate Current Screen** option applies only to the beginning of the step's operation. Once the current screen has been validated, this option has no effect on the rest of the step's operation.

Checks and recovery are performed as follows:

1. The runtime operations verify that the current screen matches the destination screen prior to the step's execution. This is done by matching the screen displayed in the **Trail** tab's Presentation view (see "Presentation View" on page 84) against the destination screen specified in the step's properties (in the **General** tab's **Destination Screen** drop-down list).
2. One of the following takes place:
   - If the Presentation view matches the destination screen, the step is executed as usual.
   - If the Presentation view does not match the destination screen, the runtime code attempts to return to the proper screen.
3. To return to the proper screen, the runtime code first determines whether the Presentation view is displaying a known screen, by matching it against the previously recorded screens.
   - If a matching screen is found *and* the step is configured to use the AutoContinue functionality (i.e. the **is AutoContinue** check box is checked under this step's **RunTime Options**), the code attempts to get back to the original destination screen and continue the method step from there.
   - If a matching screen cannot be found, or the step is *not* configured to use the AutoContinue functionality, the code attempts recovery using the **Recovery Action** set for this step (see "Recovery Action" on page 299).

4 The recovery action attempts to reach the proper destination screen.
   - If the proper destination screen is reached, the step continues as usual.
   - If the proper destination screen is not reached, the standard onFail processing takes over (see "Method Step Links" on page 355).

Note: For Write and Perform steps, the AutoContinue option is disabled. Instead, the code simply attempts to recover using the selected recovery action. If no recovery action is selected and the correct destination screen has not been reached, the step fails and standard onFail processing takes over.

Recovery Action

When the current screen (shown in the Presentation view) does not match a screen-based step's destination (set in its **General** tab), *and* the AutoContinue functionality is either disabled or fails, you may run an action that attempts to reach the proper destination screen. This action is known as a recovery action.

Note: The recovery action may be performed in conjunction with the Validate Current Screen operation, but it may also be configured and used separately.

The recovery action is set in the **General** tab of the step's Properties pane, by selecting any defined global action from the **Recovery Action** drop-down list. This action should be defined to contain all mapped screens as possible destinations, thereby allowing the runtime operations to quickly identify the current location, and determine whether the recovery was successful.

The timeout of the recovery action, i.e. the amount of time the service waits for a known screen to be reached, before marking the action as failing, should be set to a fairly low value (the default is 60 seconds), so as to not cause large delays when an unknown screen is encountered.

The recovery is attempted as follows:

1 The selected recovery action is sent to the host application and attempts to reach a known screen.
2 The screen that has been reached is compared to the step's appropriate destination.
3 One of the following takes place:
   - If the recovery action manages to reach the destination screen, the step continues as usual.
   - If the recovery action fails to reach the destination screen, the standard onFail processing takes over (see "Method Step Links" on page 355), including the **is AutoFetch** runtime option (see "AutoFetch Option" on page 302).

Note: For Fetch and Perform steps, the recovery action is executed only if the **Page Down** action (defined in the table template Properties' **Table Paging Control** tab) fails to reach the appropriate destination screen. If the recovery operation fails as well, the step is marked as "failed" and the standard onFail processing takes over.

Traverse Steps

Traverse steps instruct the method to traverse from the current screen to another screen. Since the information required to perform the traversal is contained in the map, all that is required is the identification of which screen to move to.

By default, a Traverse step's destination is the first screen in your map (e.g. the Login screen). To select the destination of the Traverse step, create a link from the previous step (e.g. the Start step) to the Traverse step. The **Choose a Destination** dialog box is displayed (Figure 178), allowing you to select the appropriate destination.

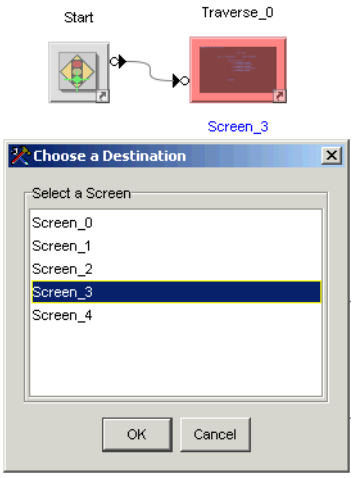


Figure 143. Choose a Destination Dialog Box

After the initial screen selection, you can set the Traverse step's destination using the Properties view. To do so, select the Traverse step in either the Tree view or the Presentation view. The **General** tab of the Traverse step's Properties view is displayed (Figure 144).

Traverse Step Properties > General Tab

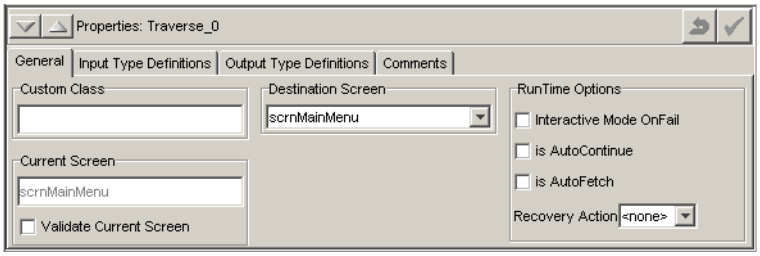


Figure 144. Traverse Method Step Properties > General

The **General** tab of the Traverse method step's Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Current Screen	Shows the map screen currently displayed for this method step.
Validate Current Screen	Verifies that the current screen as defined in the step is the actual current screen before this step is executed. For detailed information, see "Validating the Current Screen" on page 298.
Destination Screen	The screen this step traverses to.

Option	Description
RunTime Options	• **Interactive Mode onFail**: If this checkbox is set, UserInteraction Mode begins if the Traverse method step fails. For more information on the UserInteraction step and its relationship to the Traverse method step, see "UserInteraction Steps" on page 313. **Note:** The Interactive Mode onFail feature is not supported when the webMethods JI Service is deployed as a Resource Adapter. • **is AutoContinue**: If this checkbox is set, it indicates that the method step attempts to continue automatically if a screen that the method step arrives at does not match the expected screen. Applies to Traverse and UserInteraction method steps only. For more information, see "AutoContinue Option" on page 303. • **is AutoFetch**: If this checkbox is set, this indicates that the AutoFetch option is set on the method step. Applies to Traverse, Perform and UserInteraction method steps only. For more information, see "AutoFetch Option" on page 302.
Recovery Action	The global action that attempts to reach the step's destination screen. The recovery action is executed when the current screen does not match the step's destination screen, and the AutoContinue functionality is either disabled or has failed. For more information, see "Recovery Action" on page 299.

AutoFetch Option

AutoFetch is part of the onFail processing and applies to the Traverse, Perform and UserInteraction steps. If the recovery action (see "Recovery Action" on page 299) fails to reach the destination screen, AutoFetch takes over and allows the possible return of some screen information to the client.

To use AutoFetch, proceed as follows:

1 Record a trail that includes the error screen that is encountered.

2. Create a data template for the error screen. In that data template, map the error message field as a data field.
3. Create a Traverse or Perform method step and set **Is AutoFetch** in the **General** tab of the method step Properties view.
4. If an unexpected screen is encountered when the method step is executed, the service code determines if it can recognize the screen. If it recognizes the screen, it extracts the message from the data field. The message is then returned to the client.

Note: Custom coding can be added that will execute another method based on the error message or handle the error in other ways. For more information about custom coding, see "Custom Classes" on page 112.

The AutoFetch option applies to Traverse and Perform method steps only.

AutoContinue Option

During the execution of a Traverse step, a screen that is not included in the standard traversal may be encountered. This is most often an error screen, although it does not have to be. In this event, the Traverse method step can be configured to use its AutoContinue functionality. AutoContinue method steps are similar to AutoFetch, except that rather than extracting an error message and returning the message to the client, AutoContinue method steps attempt to complete the traversal even though an unexpected screen has been encountered.

To use the AutoContinue functionality:

1. Record a trail that starts with the unexpected screen that is encountered and traverses to the destination screen.
2. Create a Traverse method step and set the **Is AutoContinue** checkbox in the **General** tab of the step's Properties view.
3. When the AutoContinue method step is executed and the unexpected screen is encountered, the service code determines if it can recognize the screen and can determine how to traverse from the unexpected screen to the destination screen.
4. If it cannot transfer to the destination screen, the standard (onFail) failure behavior is performed.

The AutoContinue option applies to Traverse method steps only.

Traverse Step Properties > Input/Output Type Definitions Tabs

The Traverse step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. See "Start Step Properties > Input Type Definitions Tab" on page 294.

The Traverse step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. See "Start Step Properties > Output Type Definitions Tab" on page 294.

Fetch Steps

Fetch steps instruct the method to extract data from the data template or table template specified in the Fetch step's properties. To configure the Fetch step's properties, select it in either the Tree view or the Presentation view. The Fetch step properties are displayed in the Properties view.

> **Note:** Data templates or table templates that are associated with a Fetch step are listed under that step in the Tree view.

Fetch Step Properties > General Tab

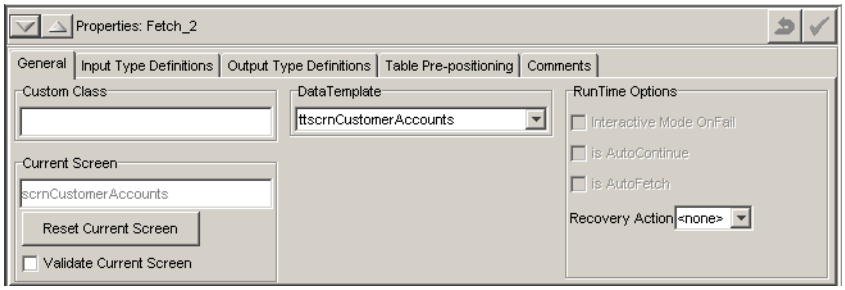


Figure 145. Fetch Step Properties > General Tab

The **General** tab of the Fetch step Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Current Screen	The expected current screen for this method step.
Reset Current Screen	Re-reads the current method flow and resets the current screen accordingly.

Option	Description
Validate Current Screen	Verifies that the current screen as defined in the step is the actual current screen before this step is executed. For detailed information, see "Validating the Current Screen" on page 298.
Data Template	The name of the data template or table template associated with this method step.
RunTime Options	Does not apply to the Fetch step.
Recovery Action	The global action that attempts to reach the step's destination screen. The recovery action is executed when the current screen does not match the step's destination screen, and the AutoContinue functionality is either disabled or has failed. For more information, see "Recovery Action" on page 299.

Disabling Data Fields

After selecting the data template or table template fetched by this step, the template's data fields in are listed in the Tree view. By default, all data fields are included in the service message that is returned to the client. To disable a field to prevent it from being included in the return service message, right-click on the field and select **Disable**. The value can still be extracted from the legacy screen to be included in an Output, but it will not be included in the service message.

Similarly, all fields can be disabled so that the entire data or table template is not included in the return service message, but the fields can still be used for Outputs.

Creating Output Variables

Outputs are used to populate Variables, and are added to the **Method** tree based on the data that is mapped from Fetch method steps. To add output variables to the **Method** tree, see "Outputs" on page 360.

Fetch Step Properties > Input/Output Type Definitions Tabs

The Fetch step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. See "Start Step Properties > Input Type Definitions Tab" on page 294.

The Fetch step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. See "Start Step Properties > Output Type Definitions Tab" on page 294.

Fetch Step Properties > Table Pre-positioning Tab

The **Table Pre-positioning** tab of the Fetch step's Properties view (Figure 146) allows you to search a table for a specific record, and then update the record as needed.

> **Note:** This tab is enabled only if a table template has been selected in the step's **General** tab (see "Fetch Step Properties > General Tab" on page 304).

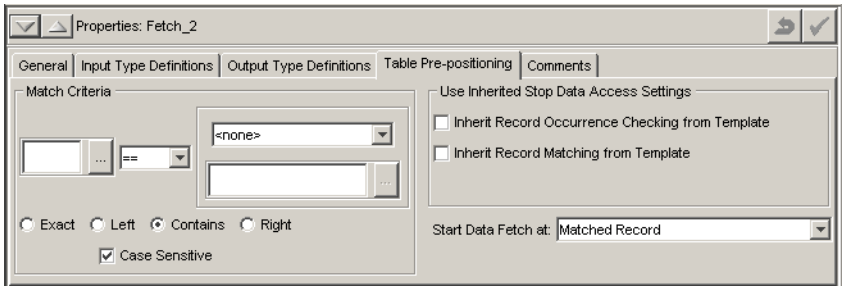


Figure 146. Fetch Step Properties > Table Pre-positioning Tab

The **Table Pre-positioning** tab of the Fetch step Properties view provides the following options:

Option	Description

Match Criteria	Define the criteria against which the table's records are matched. These criteria include:
	• A relationship between two data sources (i.e. Variables or Variable components): Specify the objects to be compared in the left and right fields, and then set the relationship between them using the operators in the middle drop-down list.
	Note: To specify a data source, either enter its name or click the ellipsis button ... to choose it from the **Choose a Data Source** dialog box.
	• Define the extent of the above match using the following options:
	  • **Exact**: The record must match every character in the criteria.
	  • **Left**: The record must match the first (left-most) characters in the criteria.
	  • **Contains**: The record must contain the text, but the text may appear anywhere within the record.
	  • **Right**: The record must match the last (right-most) characters in the criteria.
	  • **Case Sensitive**: The case of the characters in the record must match the case of the text in the criteria.
	Note: If these **Match Criteria** are not fully configured, the **Table Pre-positioning** tab is considered disabled and the step performs as usual.

Option	Description
Use Inherited Stop Data Access Settings	If a record matches one of the following settings, defined in the table template, the search stops and the step is marked as failed: • **Inherit Record Occurrence Checking from Template**: webMethods JI stops retrieving records after one or more occurrences of either empty records or repeated records. • **Inherit Record Matching from Template**: webMethods JI stops retrieving records when the two data sources specified in the table template are matched.
Start Data Fetch at	Select the record from which the data fetch operation begins. Choose one of the following: • **Matched Record**: The fetch starts from the record that met the **Match Criteria**. • **Next Record**: The fetch starts from the record following the one that met the **Match Criteria**.

At runtime, the step searches the table template, record by record, looking for a valid match criteria statement in the following sequence:

- If a record matches one of the properties that are defined in the table template and selected under **Use Inherited Stop Data Access Settings**, the search stops, the step is marked as failed and the standard failure operations take place.
- If a record matches the **Pre-positioning** tab's **Match Criteria**, the search is considered complete and successful. The standard step operations take place, starting from the **Matched** or **Next Record**.
- If no record matches the **Inherited Stop Data Access Settings** or the **Match Criteria**, the step is marked as failed and the standard failure operations take place.

Perform Steps

Perform method steps instruct the method to perform a specified action. The information about the action (such as the AID key to be executed) is contained in the map. To view this information, display the **Map** tab and select the relevant action in the Tree view. The action's settings are displayed in the Properties view. All that is required to define a Perform step is the identification of the action to be executed. Perform steps can also be used in **onFail** links to execute global actions. For more information about global actions, see "Global Actions" on page 153.

To configure the Perform step's properties, select it in either the Tree view or the Presentation view. The Perform step properties are displayed in the Properties view.

Perform Step Properties > General Tab

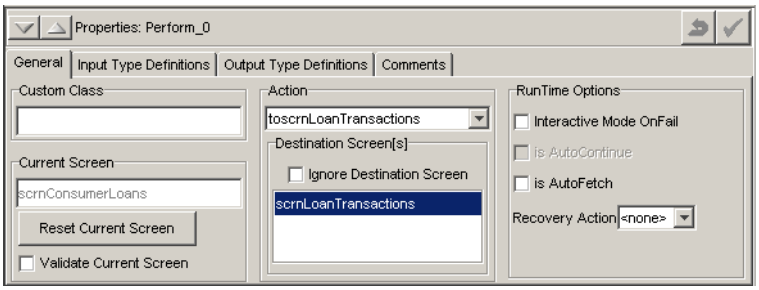


Figure 147. Perform Step Properties > General Tab

The Perform method step Properties **General** tab provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Current Screen	Shows the map screen currently displayed for this method step.
Reset Current Screen	Resets the current screen to the settings that were last applied for the fields and options displayed.
Validate Current Screen	Verifies that the current screen as defined in the step is the actual current screen before this step is executed. For detailed information, see "Validating the Current Screen" on page 298.
Action	Select the name of the action associated with this method step from the drop-down list.

Option	Description
Destination Screen[s]	If the selected action has more than one possible destination screen, select the destination screens that are allowed for this method step. The perform step fails if it arrives at a screen that is not selected in this list. Multiple screens can be selected by holding down the **Shift** key and clicking in the list box to select multiple screens, or by holding down the **Ctrl** key and clicking on a screen name to select individual screens. The initial selected screen is the primary destination of the action (selected in the Action Properties view; see "Modifying Actions" on page 165). When the value is changed here, this does not effect the primary destination defined in the Action Properties view. Similarly, changing the primary destination in the Action Properties view does not change the screens selected for this field.
Ignore Destination Screen	If this checkbox is set, the perform step will not fail as long as any screen known in the map is reached after the action is performed.
RunTime Options	• **Interactive Mode onFail**: If this checkbox is set, UserInteraction Mode begins if the Perform method step fails. For more information on the UserInteraction Step and its relationship to the Perform method step, see "UserInteraction Steps" on page 313. **Note:** *The Interactive Mode onFail feature is not supported when the webMethods JI service is deployed as a Resource Adapter.* • **Is AutoContinue**: Does not apply to Perform steps. • **Is AutoFetch**: If this checkbox is set, this indicates that the AutoFetch option is set on the method step. Applies to Traverse, Perform, and UserInteraction method steps only. For more information, see "AutoFetch Option" on page 302.

Option	Description
Recovery Action	The global action that attempts to reach the step's destination screen. The recovery action is executed when the current screen does not match the step's destination screen, and the AutoContinue functionality is either disabled or has failed. For more information, see "Recovery Action" on page 299.

Perform Step Properties > Input/Output Type Definitions Tabs

The Perform step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The Perform step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information, see "Start Step Properties > Output Type Definitions Tab" on page 294.

Write Steps

Write steps instruct the method to input data into a write or read/write data or table template. A Write step can only be correctly configured if there is a data or table template available on the current screen that is write or read/write enabled.

> **Note:** Data templates or table templates that are associated with a Write step are listed under that step in the Tree view.

To configure the Write step's properties, select it in either the Tree view or the Presentation view. The Write step properties are displayed in the Properties view.

Write Step Properties > General Tab

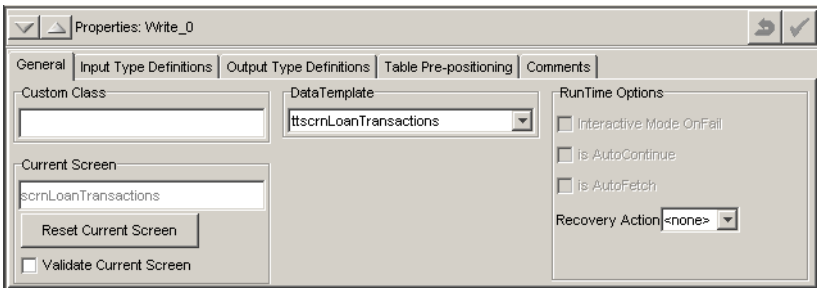


Figure 148. Write Step Properties > General Tab

The **General** tab of the Write step Properties view provides the following options:

Option	Description
Custom Class	The custom class that is to be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Current Screen	The expected current screen for this method step.
Reset Current Screen	Re-reads current method flow and resets the current screen accordingly.
Validate Current Screen	Verifies that the current screen as defined in the step is the actual current screen before this step is executed. For detailed information, see "Validating the Current Screen" on page 298.
Data Template	The name of the template associated with this method step.
RunTime Options	Does not apply to Write steps.
Recovery Action	The global action that attempts to reach the step's destination screen. The recovery action is executed when the current screen does not match the step's destination screen, and the AutoContinue functionality is either disabled or has failed. For more information, see "Recovery Action" on page 299.

Write Step Properties > Input/Output Type Definitions Tabs

The Write step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The Write step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information, see "Start Step Properties > Output Type Definitions Tab" on page 294.

Write Step Properties > Table Pre-Positioning Tab

The Write step's **Table Pre-Positioning** tab is identical to the Fetch step's corresponding tab. For information, see "Fetch Step Properties > Table Pre-positioning Tab" on page 306.

UserInteraction Steps

UserInteraction steps allow the user to navigate interactively with the legacy system.

> **Note:** UserInteraction method steps are not supported when the webMethods JI Service is deployed as a Resource Adapter.

To configure the UserInteraction step's properties, select it in either the Tree view or the Presentation view. The UserInteraction step properties are displayed in the Properties view.

UserInteraction Step Properties > General Tab

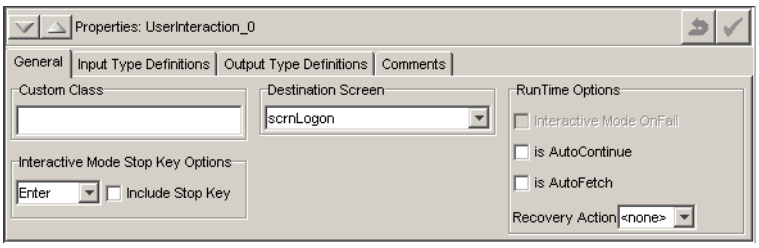


Figure 149. UserInteraction Step Properties > General Tab

The **General** tab of the UserInteraction Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Interactive Mode Stop Key Options	This is a drop-down list containing the available action keys to choose as a stop key with this method step.

Option	Description
Include Stop Key	Set this check box if the stop key is to be passed through to the host application.
Destination Screen	This is a drop-down list containing available screens to choose as a destination for this method step.
RunTime Options	• **Interactive Mode onFail**: This option does not apply to the following: - UserInteraction steps - When the webMethods JI Service is deployed as a Resource Adapter • **Is AutoContinue**: If this checkbox is set, it indicates that the method step will attempt to continue automatically if a screen that the method step arrives at does not match the expected screen. Applies to Traverse and UserInteraction method steps only. For more information, see "AutoContinue Option" on page 303. • **Is AutoFetch**: If this checkbox is set, this indicates that the AutoFetch option is set on the method step. Applies to Traverse, Perform, and UserInteraction method steps only. For more information, see "AutoFetch Option" on page 302.
Recovery Action	The global action that attempts to reach the step's destination screen. The recovery action is executed when the current screen does not match the step's destination screen, and the AutoContinue functionality is either disabled or has failed. For more information, see "Recovery Action" on page 299.

UserInteraction Step Properties > Input/Output Type Definitions Tabs

The UserInteraction step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The UserInteraction step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

Invoke Steps

This section describes:

- "Internal Invoke Steps" on page 315
- "External Invoke Steps" on page 316
- "JClient3 Steps" on page 329

Internal Invoke Steps

An Internal Invoke step allows you to invoke another method that resides in the same map (i.e. the same service). The two methods can use the same Variables, and no network operations are required.

To configure the properties of an Internal Invoke step, select it in either the Tree view or the Presentation view. The step's properties are displayed in the Properties view.

Internal Invoke Step Properties > General Tab

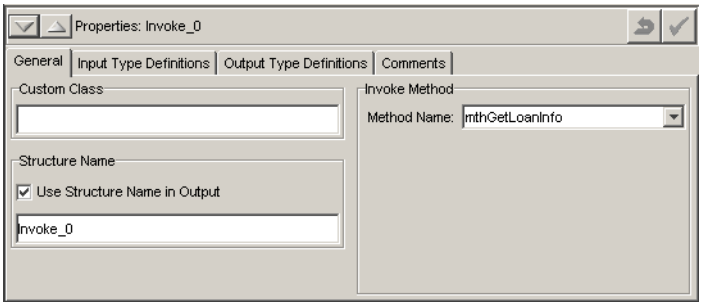


Figure 150. Internal Invoke Step Properties > General Tab

The **General** tab of the Internal Invoke step Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.

Option	Description
Structure Name	The key value inserted into the return map. Setting this option overrides the default setting which is the name of the method step.
Use Structure Name in Output	When set, the Structure Name key is inserted into the method return map.
Invoke Method	Select the method you wish to invoke from the **Method Name** drop-down list, which contains all the methods available in the map.

Internal Invoke Step Properties > Input/Output Type Definitions Tabs

The Internal Invoke step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The Internal Invoke step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

External Invoke Steps

An External Invoke step allows you to invoke a method that resides in a different map (i.e. a different service) on the same virtual machine. This step provides multi-developer support, allowing different users to share a host connection and work on different parts of the same legacy system. An External Invoke step requires passing input and output parameters to the external method, using Data Mapping. No network operations are required.

> **Note:** Make sure you do not create a condition that may cause the same exact method to be re-executed, as this may create an infinite loop.
> In addition, before and after an external invoke operation, make sure the invoker service is at a screen that is defined in the external service; and similarly, that the external service is at a screen that is defined in the invoker service.

To configure the properties of an External Invoke step, select it in either the Tree view or the Presentation view. The step's properties are displayed in the Properties view.

This section describes:

- "External Invoke Step Properties > General Tab" on page 317
- "External Invoke Step Properties > Metadata Tab" on page 318
- "Invoking a Version 3.x Method from a Version 4.x Method Using Internal Metadata" on page 321
- "External Service Configuration Properties" on page 326
- "Preventing Client Access to External Services" on page 326
- "External Service Logging" on page 326
- "Representation in Client Code" on page 327
- "Representation in Service Interface Report" on page 327
- "ea_admin Representation" on page 328
- "Notes on Global Variable Sharing" on page 328

External Invoke Step Properties > General Tab

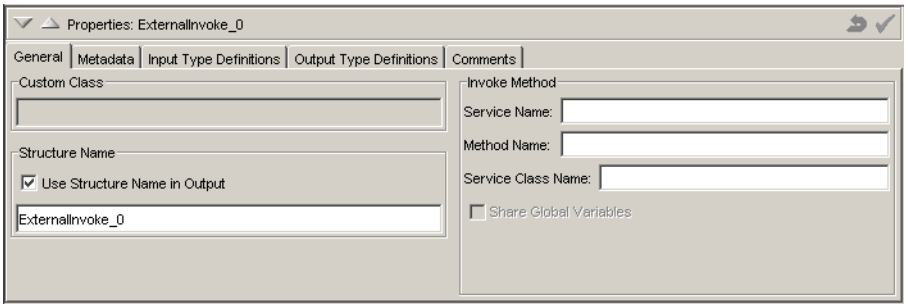


Figure 151. External Invoke Step Properties > General Tab

The **General** tab of the External Invoke Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Structure Name	The key value inserted into the return map. Setting this option overrides the default setting which is the name of the method step.
Use Structure Name in Output	When set, the Structure Name key is inserted into the method return map.

Option	Description
Invoke Method	Select the external method to be invoked: • **Service Name**: The name of the service in which the external method is located, as it is referenced in the Service Master of the same JI Resource Database. Be careful to maintain case-sensitivity. • **Method Name**: The name of the external method to invoke. Be careful to maintain case-sensitivity. • **Service Class Name**: The class name of the service in which the external method is located. You must provide a fully qualified class name. The naming is as follows: `<Service Package Name>.<Service Name>` The Service Package Name is obtained from the **Service Package Name** field in the **Generate Code** dialog box from within the external service's map file.
Share Global Variables	If set, the values of Global Variables are transferred from the invoking service to the external service before the external method is invoked and transferred back to the invoker after the invocation is finished. Default Value: Cleared. For more information, see "Notes on Global Variable Sharing" on page 328. **Note:** This property is only applicable to version 3.X compatibility methods.

External Invoke Step Properties > Metadata Tab

Note: The **Metadata** tab is common to both the External Invoke step and the JClient3 step. This section (pages 318-325) refers to the External Invoke step, but applies to the JClient3 step as well.

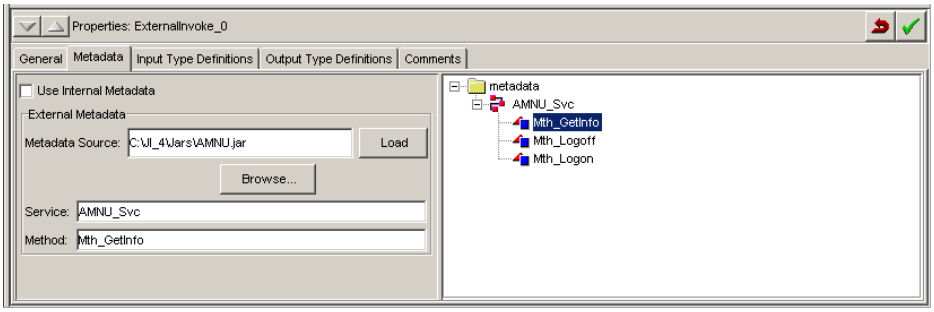


Figure 152. External Invoke Step Properties > Metadata Tab

The **Metadata** tab allows you to specify if and how metadata are imported from the external source.

> **Note:** The data mapping can only be performed if the version of the invoker method is 4.x (i.e. the **Basic Data Operations** property is not set in the **General** tab of the method's Properties view). If the method is version 3.x compatible, the techniques that were used in version 3.5 are applied.

The Data Mapping is performed as follows:

- From a Variable in the invoker method to the inputs of the invoked method (on the input edge of the step).
- From the output of the invoked method to Variables in the invoker method (on the output edge of the step).

The information required for performing the data mapping is gathered using one of the following techniques:

- External Metadata — fully supports webMethods JI version 4.x data mapping. This technique enables you to extract method, input and output information from the JAR file that is created when MapMaker generates the service code. In addition, it allows you to select the external version 4.x method to be invoked, which automatically completes the **Service Name** and **Method Name** properties in the **General** tab of the External Invoke step's Properties view (see "External Invoke Step Properties > General Tab" on page 317). External Metadata is the preferred technique for invoking version 4.x methods.
- Internal Metadata — this technique is mainly used for invoking version 3.x compatibility methods. It requires that you create MLM XBEs that correspond to the invoked method's input and output data structures (for detailed instructions, see "Invoking a Version 3.x Method from a Version 4.x Method Using Internal Metadata" on page 321).

The **Metadata** tab provides the following options:

Option	Description
Use Internal Metadata	If set, metadata is not imported from the external source and the internal metadata is used instead. This disables all other options and displays two list-boxes (see Figure 153): • **Input MLM:** The MLM data type used as input. • **Output MLM:** The MLM data type used as output. You must select the appropriate data types in each of these lists.
External Metadata	• **Metadata Source**: Enter the path and name of the JAR file of the external map. • **Browse**: Click to browse to the location of the JAR of the external map. Once the JAR file is chosen, this is reflected in the Metadata Source field. • **Load**: Click to update the Metadata source. If there have been changes made to the external map since last entered, these are updated in the Tree view on the right. • **Service**: Displays the external service. To determine which service is displayed in this field, click the desired service from the Tree view on the right. • **Method**: Displays the external method. To determine which method is displayed in this field, click the desired method from the Tree view on the right.

External Invoke Step Properties > Input/Output Type Definitions Tabs

The External Invoke step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The External Invoke step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

Invoking a Version 3.x Method from a Version 4.x Method Using Internal Metadata

When invoking a version 3.x method from a version 4.x method, you can still perform partial data mapping by using the internal metadata feature. This feature is defined using MLM structures that correspond to the input and output of the version 3.x method.

The internal metadata feature is enabled by setting the **Use Internal Metadata** checkbox in the **Metadata** tab of the External Invoke step's Properties view (Figure 152).

> **Note:** If the **Use Internal Metadata** checkbox is set but the version of the invoked method is actually 4.x, the data mapping is still performed as if the invoked method was a version 3.x compatibility method.

To invoke a version 3.x method from a version 4.x method using Internal Metadata:

1. Create the MLM structure(s) that correspond to the invoked method's inputs and outputs (for detailed instructions, see "Creating MLM Structures when Using Internal Metadata" on page 323).
2. In MapMaker's **Methods** tab, add an External Invoke step to the method flow by clicking the External Invoke step toolbar button ( ).
3. Select the new External Invoke step in the Presentation view. The step's properties are displayed in the Properties view.
4. Select the **Metadata** tab and set the **Use Internal Metadata** checkbox.

   The **Metadata** tab display is reset, showing the **Internal Metadata** options (Figure 153).

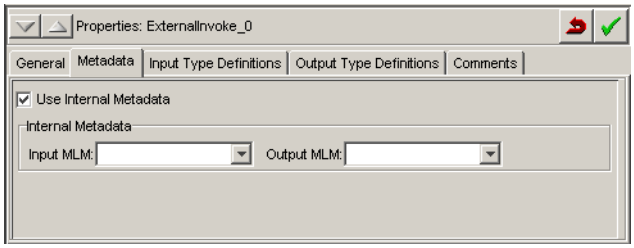


Figure 153. External Invoke Step Properties > Metadata Tab with Internal Metadata

5. Select the proper **Input MLM** and **Output MLM** values from the drop-down lists.
6. Create the necessary IBEs and Global Variables.

7 Drag the External Invoke step to its appropriate place in the method flow.

8 Create the input edge of the step, by dragging a **next** link from the previous step in the flow to the External Invoke step.

9 Data map the input edge of the step, by double clicking on it to display the **Data Mapping Editor** and mapping the appropriate Global Variables to the **Target** input MLM (Figure 154).

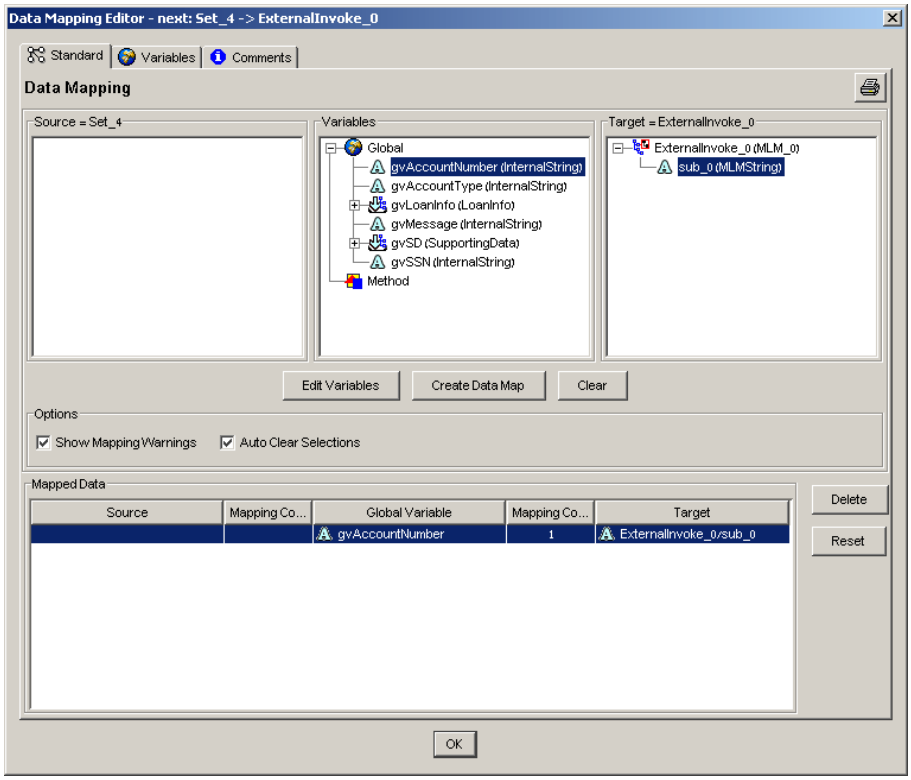


Figure 154. Mapping the Input Edge of the External Invoke Step

10 Create the output edge of the step, by dragging a **next** link from the External Invoke step to the following step in the method flow.

11 Data map the output edge of the step, by double clicking on it to display the **Data Mapping Editor** and mapping the **Source** output MLM to the appropriate Global Variables (Figure 155).

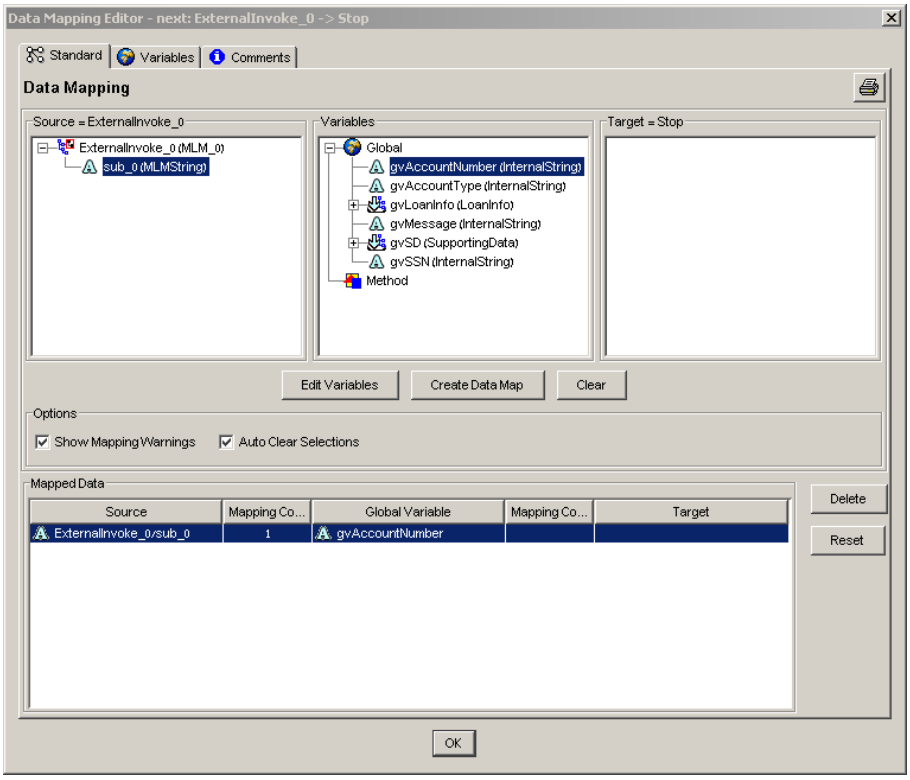


Figure 155. Mapping the Output Edge of the External Invoke Step

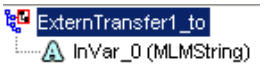
Creating MLM Structures when Using Internal Metadata

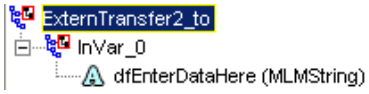
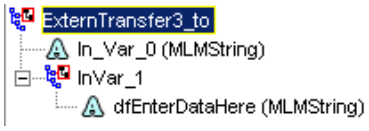
In order to use internal metadata, you must create MLM structures that correspond to the input and output data of the invoked version 3.x method. These MLMs must then be selected from the **Input MLM** and **Output MLM** drop-down lists of the External Invoke step's **Metadata** tab (Figure 153).

This section provides the following instructions:

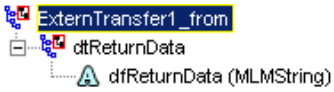
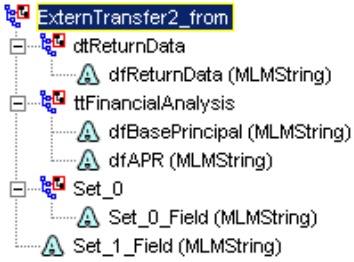
- "Creating the Input MLM" on page 323
- "Creating the Output MLM" on page 324

Creating the Input MLM You are required to create an input MLM that corresponds to the inputs of the invoked version 3.x method. The table below lists commonly used method input types, and provides examples of their corresponding input MLMs (assuming the method input name is `InVar_0`).

Method Input Description	Input MLM
Simple	

Method Input Description	Input MLM
Structured (based on a data template called “dtEnterDataHere” with one field called “dfEnterDataHere”)	
Two inputs, named `InVar_0` and `InVar_1`	

Creating the Output MLM You are required to create an output MLM that corresponds to the data returned from the invoked version 3.x method. The table below lists commonly used method output types and provides examples of their corresponding MLMs:

Method Output Description	Output MLM
A fetch of a data template named `dtReturnData` with one field named `dfReturnData`	
A fetch of multiple return data, from the following sources: • A data template • A table template • Several Set steps	

Data Transfer to the Invoked Method

Before the external method is invoked, its inputs are set using the data defined in the Input MLM structure. This is done using the following algorithm:

```
For (each method input in the external method) {
   Get the value from the Input MLM by using the input name;
   If (the value exists) {
   Store the value in the input;
   else {
   Log a message;
   }
```

```
}
```

If the input MLM structure does not contain elements for all of the method input fields (for example, the method input may include three fields while the input MLM has only two corresponding elements), no fatal error occurs. Instead, one of the following takes place:

- If the invoked method's input is simple, a warning message is logged.
- If the invoked method's input is structured (i.e. based on a template), no warning message is logged.

Once the values of the inputs are set, it is up to the invoked method to use the required data.

Data Transfer from the Invoked Method

After the external method is invoked, its returned map is parsed and used to set the required values in the Output MLM structure. This is done using the following algorithm:

```
For (each leaf node in the invoke return map) {
   If (Output MLM has the matching field) {
   Set the value in the MLM;
   }
   else {
   Log a message;
   }
}
```

If the output MLM structure does not contain elements for all of the method output's fields, no fatal error occurs. Instead, a warning message is logged.

Table Template Return Data If the invoked method's output contains multiple leaf nodes with the same path, they are stored as an array, with the repetition occurring at the template name.

For example, if the invoked map contains a table template named `ttReturnData`, which in turn has a field called `tfReturnDataC1`, the returned map may include several values whose path is `ttReturnData/tfReturnData`. These values are stored internally as `ttReturnData[<index>]/tfReturnData`, with the `<index>` value starting at zero and counting up for every instance encountered.

Generated Service Code

To facilitate the implementation of the **Use Internal Metadata** feature, the External Invoke step creates the following:

- A method input variable, whose type is set to the Input MLM value. This input variable is set by the input edge data mapping.
- A method output variable, whose type is set to the Output MLM value. This output variable is read by the output edge data mapping.

The creation of these input and output variables is performed by the generated service code. This can be seen by examining the generated Java code of the externally invoked method step.

External Service Configuration Properties

The service master and service detail configuration properties for external services are not used, except for the following properties, which are used during runtime to enable the loading of the external service:

- Class name for application map
- Version ID for map
- Map location
- Path to local map
- MapMaker-designated service name

If a change is made to the configuration settings of an external service, the change is handled as if the current service has been changed. This behavior is governed by the Configuration Manager in the **Environment behavior for Svc config changes** property. This property is set in the Environment Manager configurations. For more information, see “About Environment Managers” on page 464.

Preventing Client Access to External Services

You may wish to create a service whose only purpose is to house methods for external invocation from other services. In such a case, you would have to prevent client access to this service. To do so, launch the Configuration Manager and set the **Status** property of this service to **Disabled**. For more information, see “Setting Properties for Service Masters” on page 490

External Service Logging

The current service log output includes the log output of any external services as well. Therefore, the external service’s logging level is set by the current service. For more information about logging support, see Chapter 5 - "Logging Information" on page 155 of the *Supplemental Reference Guide*.

Representation in Client Code

Note: This section applies only to version 3.X compatible methods.

During design time, the metadata for External Invoke methods is not available. Instead of the metadata in the client code are placeholders in the form of comments. You may choose to update these manually. The following table shows these comments as they appear in each file:

File(s)	Representation
<service>_<method name>_out.html	In place of the data template and data field information, the following comment appears: `<!-- No output generated for external invoke step: <step name> -->`
<service>_<method name>.dtd <service>_<method name>_out.dtd <service>_<method name>_out.xml	• The DTD starts with the following comment: `<!-- This method contains one or more external invoke steps, please fill in the proper DTD information. -->` • Each External Invoke step contains the following comment: `<!-- Insert DTD information for external invoke step <step name> here. -->`

Representation in Service Interface Report

In a Service Interface Report generated by MapMaker, each representation of an External Invoke step has three properties followed by a comment indicating that some information is not available. These properties are `"Service"`, `"Method"`, and `"step type"`.

Following is an example of an External Invoke step named `"Invoke_0"`:

```
Invoke_0 (Service: my_service Method: my_method) (step type =
ExternalMethodStepInvoke)
```

From the MapMaker **Tools** menu, select **Reports** and then **Service Interface** to view the Interface Report for Listed Service and Method Metadata Listing.

ea_admin Representation

The `ea_admin jclusters -e` command displays information about every active service in every JCluster. External services are represented in the `jserviceExternalServices` property. This property contains a list of the external services for each JService.

The format of each entry is as follows:

```
<service name><host connection>
```

where <host connection> is either `"@SHARED"` if the host connection is open, or blank if the external service is not connected to the host.

Notes on Global Variable Sharing

> **Note:** This section applies only to version 3.X compatible methods that perform an external invoke.

When an external method is invoked, method inputs, followed by Global Variables, are transferred to the external service.

The order of events is as follows:

1. First, the client-generated service message is applied to the invoked method's inputs.
2. Then, the Global Variable values are transferred.

When Global Variable values are transferred to/from an external service, the value is transferred by a reference to the Java object using the `getValue()/setValue()` methods in the GBVar class. If the value of the Global Variable is a basic Java class object (e.g. `java.lang.String, java.lang.Integer`, etc.), it cannot be changed and therefore there are no side effects.

If the object is a more complex class (e.g. `java.util.List` or `java.util.Map`), an external service can retain a reference to the object and modify it later, even though the **Share Global Variables** check box was not set for the External Invoke.

The following example illustrates this:

Assume there are two services, Service1 and Service2. Service1 contains Method1 and Service2 contains Method2. Both services contain the Global Variable `gvGlobalVar`. Method1 in Service1 has two External Invoke steps that invoke Method2 in Service2. One has the **Share Global Variables** option set and the other one does not.

The following steps are performed:

1. Method1 in Service1 is invoked by a client. It sets the value of `gvGlobalVar` to be a `java.util.ArrayList`. Method2 in Service2 is invoked by the first External Invoke step. The contents of `gvGlobalVar` are transferred to Service2 before the invoke.
2. Method2 modifies the value of the Global Variable by adding a string to the `java.util.ArrayList` object. After the invoke, the contents of the Global Variable are transferred back to Service1.
3. The second External Invoke step in Method1 causes Method2 in Service2 to be invoked. The contents of `gvGlobalVar` are not transferred to Service2 because the **Share Global Variables** check box is not set for this invoke step. Method2 modifies the contents of its Global Variable. This also modifies the contents of the Global Variable in Service1, since Service2 still has a reference to the `java.util.ArrayList` that was transferred in step #1.

JClient3 Steps

The JClient3 step allows you to invoke a method in another webMethods JI service, using JClient3 API calls. This step allows you to access multiple, parallel legacy applications. Since the invoked methods belong to a different map, data mapping operations are required. In addition, since the invoked method may be located on a different virtual machine, network operations are required as well.

> **Note:** JClient3 steps are also supported when the webMethods JI Service is deployed as a Resource Adapter, but require a webMethods JI Server environment to communicate with at runtime.

To configure the properties of a JClient3 step, select it in either the Tree view or the Presentation view. The step's properties are displayed in the Properties view.

JClient3 Step Properties > General Tab

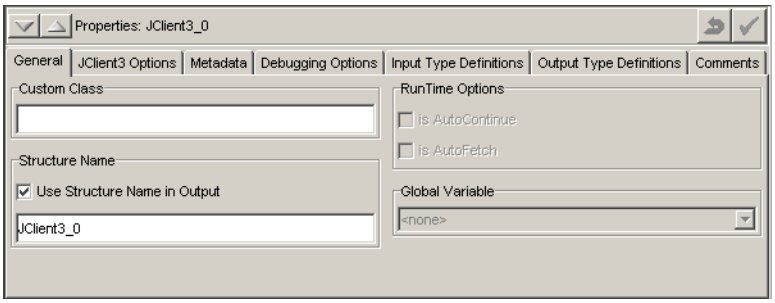


Figure 156. JClient3 Step Properties > General Tab

The **General** tab of the JClient3 step Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Structure Name	The key value inserted into the return map. • **Use Structure Name in Output**: Set to insert the Structure Name key into the method return map.
RunTime Options	Does not apply to JClient3 steps.
Global Variable	This list provides the choice of Global Variables assigned to this method.

JClient3 Step Properties > JClient3 Options Tab

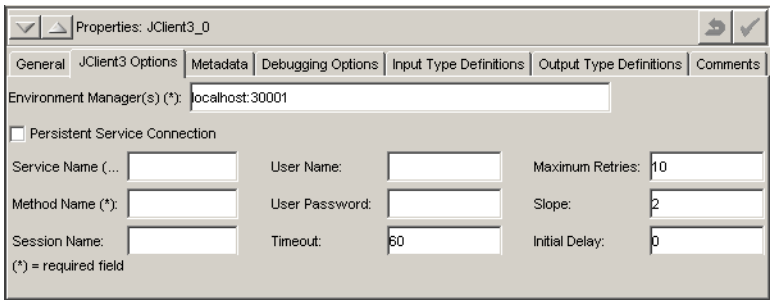


Figure 157. JClient3 Step Properties > JClient3 Options Tab

The **JClient3 Options** tab of the JClient3 step Properties view provides the following options:

> **Note:** An asterisk indicates that the field is required.

Option	Description
Environment Manager(s)(*****) (host1:port1 [host2:port2])	The host name(s) and port number(s) of your webMethods JI Environment Managers. Host name can be either a DNS name or an IP address. This property defaults to the value specified in the **Properties** dialog box, in the **Servers** tab's **Environment Manager** section (see "Servers Tab" on page 105).
Persistent Service Connection	When set, connects this service connection object to a service. The connection persists until `disconnect()` is called, regardless of whether `close()` is called
Service Name (*)	The name of the webMethods JI service.
Method Name (*)	The name of the method in the selected webMethods JI service.
Session Name	The session name used for this JClient3 service
User Name	The user name for this JClient3 service.
User Password	The user password for this JClient3 service.
Timeout	Sets the timeout value (in seconds) to be used for this JClient3 Java bean. The timeout value is used to set the timeout for the underlying `EnvironmentManagerConnection` and `ServiceConnection` objects. This value is overridden by the client timeout value configured for the service (if the client timeout configured for the service is not zero).
Maximum Retries	Sets the maxRetries for this JClient3 service. The maximum number of times to retry connections. Default is 10 retries.

Option	Description
Slope	The rate (in seconds) at which to increase delay between connection retries. Default value is 2 seconds
Initial Delay	The initial delay (in seconds) between connection retries. Default value is 0 seconds

JClient3 Step Properties > Metadata Tab

The **Metadata** tab allows you to specify whether metadata are imported from the invoked method, and if so, whether to gather the information using External Metadata or Internal Metadata.

This tab is identical to the External Invoke step's **Metadata** tab. For detailed information, see "External Invoke Step Properties > Metadata Tab" on page 318.

JClient3 Step Properties > Debugging Options Tab

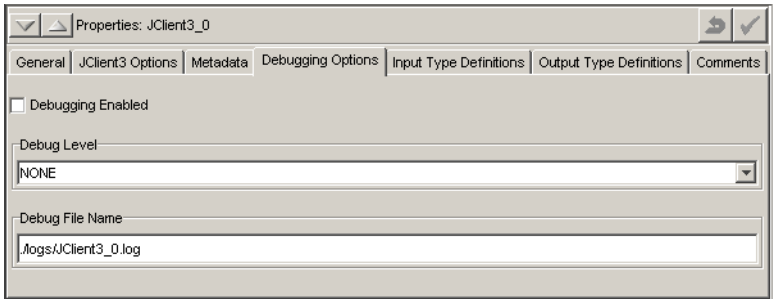


Figure 158. JClient3 Step Properties > Debugging Options Tab

The **Debugging Options** tab of the JClient3 Properties view provides the following options:

Option	Description
Debugging Enabled	When enabled, logging is set at the selected debugging level.

Option	Description
Debug Level	Four levels of debug are available from a drop down list. • **None** • **Basic** • **Normal** • **Detailed**
Debug File Name	The name of the log file for debug information. The default file is *<JI_install_dir>/logs/step_name.log* where *step_name* corresponds to the JClient3 step name.

JClient3 Step Properties > Input/Output Type Definitions Tabs

The JClient3 step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The JClient3 step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

Conditional Steps

Conditional steps are classes that extend the `MethodCondition` class. Conditional steps compare Java objects, which are generally retrieved from Variables. You may also compare an object to "constant" string data.

The Conditional steps available are:

- "IfCondition Steps" on page 338
- "IfElseCondition Steps" on page 339
- "WhileCondition Steps" on page 342
- "DoWhileCondition Steps" on page 343
- "ForCondition Steps" on page 340
- "EndIf Steps" on page 343
- "Continue Steps" on page 344
- "Break Steps" on page 344

Working with Conditional Steps — Overview

To configure the properties of a Conditional step, select it in either the Tree view or the Presentation view. The step's properties are displayed in the Properties view.

All Conditional steps besides EndIf, Continue and Break include a conditional area in the **General** tab (Figure 159).

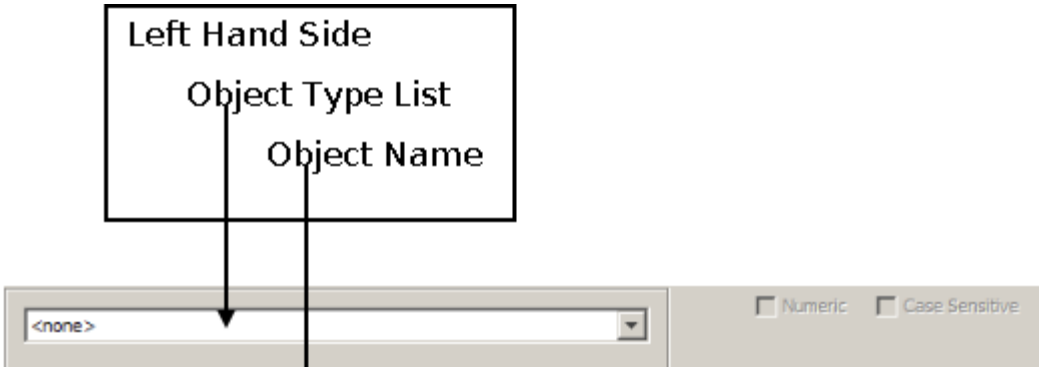


Figure 159. Conditional Area in the General Tab

A condition consists of a right hand side and a left hand side, separated by operators. On each side, you are required to first select the object type (Variable, Internal string etc.) from a drop-down list, and then specify the object name. Either type the name, or click the ellipsis button (...) to select the object from the **Choose a Data Source** dialog box).

> **Note:** Object names must be spelled correctly. Otherwise, the text is colored red, the status bar at the bottom of the MapMaker window indicates that the condition is not set properly, and the settings cannot be saved. Clicking the **Apply** button reverts the properties to the last save.

After selecting an object on each side of the condition, use the center operators to compare between them. The particular operators available are a function of the selected objects' data types. Only valid options are displayed.

For example, if Variables of an Internal Number type are selected on both sides of the condition, the **Numeric** check box is enabled and set, the **Case Sensitive** check box is cleared, and the operator drop-down list is populated with a list of numeric operators.

If the selected objects cannot be used to define a valid condition, the operators are disabled and the status bar describes the first error detected for the step in question (e.g. "IfCondition_0 The left/right combination is invalid...").

Note: You may still save an erroneous condition by clicking **Apply**. In this case, a warning message may be displayed, depending on the severity of the error (for example, when the condition interferes with code generation).

Objects Lists

The objects lists include the following types of objects:

Option	Description
Variable	Compares the specified Global or Method Variable to another Global or Method Variable. Enter the name of the relevant Variable (or Variable element) in the field below this drop-down list.
Current Screen	Compares the current screen to any screen recorded in this map.
Current Date/Time	Sets the runtime date and time as the basis for the fulfillment of a condition.
InternalString	Compares a string value to any of the available Variables. To set a case sensitive comparison, set the **Case Sensitive** check box.
InternalNumber	Compares a number value to any of the available Variables.
InternalBoolean	Compares an InternalBoolean to an InternalString. The InternalBoolean is set to “true” if the value of the InternalString is “true”, “yes”, “ok” or “1”. Otherwise, the InternalBoolean’s value is “false”.
InternalDate	Compares a date value to the current date.
The names of all other screens in this map	Allows comparison between any two screens recorded in this map.

Operators

The operators available for comparing the two objects are:

Option	Description
Numeric	Set to compare only numeric attributes. **Note:** This box is automatically enabled and set when an Internal Number is set in either the right or left hand side of the condition.
Case Sensitive	Set to perform a case-sensitive comparison. **Note:** This box is automatically disabled when an Internal Number is set in either the right or left hand side of the condition.
= =	**Equal to**: This is the default operation. Compares the referent values to determine if they are equal (e.g. “A” at location abc containing “hello” compared with “B” at location xyz also containing “hello”, are referring to different locations. This operator compares the *referents*, NOT contents). Any pair of objects may be compared in this way.
! =	**Not equal to**: Compares the referent values to determine if they are unequal. Any pair of objects may be compared in this way.
<	**Less than**: Objects must implement the compareTo method of the `java.lang.Comparable` interface. Returns a negative, zero, or positive integer, based on results of the comparison.
>	**Greater than**: Objects must implement the compareTo method of the `java.lang.Comparable` interface. Returns a negative, zero, or positive integer, based on results of the comparison.
<=	**Less than or equal to**: Objects must implement the compareTo method of the `java.lang.Comparable` interface. Returns a negative, zero, or positive integer, based on results of the comparison.

Option	Description
>=	**Greater than or equal to**: Objects must implement the compareTo method of the `java.lang.Comparable` interface. Returns a negative, zero, or positive integer, based on results of the comparison.

The following operators compare objects by their type:

instanceof	Determines if the left argument object is of a specific type. Any object may be the left argument, while the right argument is a design-time user entered String of the Class.
!instanceof	Determines if the left argument object is not of the specified type. Any object may be the left argument, while the right argument is a design-time user entered String of the Class.

The following operators compare objects by their content:

equals	Any pair of objects may be compared in this way (e.g. the “A equals B” comparison described above will now evaluate true, since the contents of strings “A” and “B” are identical).
!equals	Any pair of objects may be compared in this way (the “!” modifier is a boolean negation of the operator, which basically negates the result of the comparison. For example, “A! equals B” for the “hello” string would evaluate false).
equalsIgnoreCase	Any pair of objects may be compared in this way (`toString()` will be used if the object is not of a String type).
startsWith	Any pair of objects may be compared in this way (`toString()` will be used if object is not of a String type).

Option	Description
endsWith	Any pair of objects may be compared in this way (`toString()` will be used if object is not of a String type).
contains	Any pair of objects may be compared in this way.
!contains	Any pair of objects may be compared in this way.

Note: For detailed information about Java language conventions, see the Java language documentation at http://www.oracle.com/technetwork/java/index-jsp-142903.html#documentation.

IfCondition Steps

The IfCondition provides a comparison between two objects:

- If the evaluation of the Condition is "true" or satisfied, the step(s) on the conditional or **branch** link are executed. After the **branch** is finished, the main method logic continues (i.e. the **next** edge is followed).

  The completion of an IfCondition **branch** path is indicated by an EndIf step (see "EndIf Steps" on page 343). When the true/conditional step(s) are completed, the primary method logic path is resumed.
- If the evaluation of the Condition is "false", the primary method logic path continues.

All of the comparison operators are supported for the IfCondition.

IfCondition Step Properties > General Tab

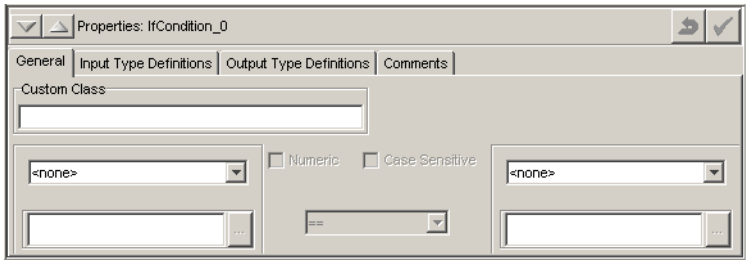


Figure 160. IfCondition Step Properties > General Tab

The **General** tab of the IfCondition step's Properties view allows you to define the following:

- The **Custom Class** that should be used with this method step. For information on custom classes, see "Custom Classes" on page 112.
- **Conditional** properties — see "Working with Conditional Steps — Overview" on page 334.

IfCondition Step Properties > Input/Output Type Definitions Tabs

The IfCondition step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The IfCondition step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

IfElseCondition Steps

The IfElseCondition step provides a comparison between two objects.

- If the evaluation of the Condition is "true", the step(s) on the conditional or **branch** link are executed and the primary method logic path is not followed.
- If the evaluation of the Condition is "false", the primary method logic path continues.

In practice, the conditional method logic path is usually designated to rejoin the primary method logic path at a later point, although it is not required. IfElse Conditions can be chained together to form a "case" statement.

All of the comparison operators are supported for the IfElseCondition.

IfElseCondition Step Properties > General Tab

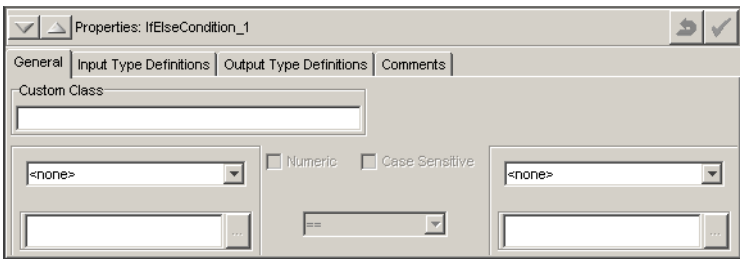


Figure 161. IfCondition Step Properties > General Tab

The **General** tab of the IfElseCondition step's Properties view allows you to define the following:

- The **Custom Class** that should be used with this method step. For information on custom classes, see "Custom Classes" on page 112.
- **Conditional** properties — see "Working with Conditional Steps — Overview" on page 334.

IfElseCondition Step Properties > Input/Output Type Definitions Tabs

The IfElseCondition step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The IfElseCondition step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

ForCondition Steps

The ForCondition step executes the step(s) on the conditional link in a loop, based on a specified numeric count. The numeric count is set in the **Incrementor** tab of the ForCondition Properties view. The conditional operators for this condition are the same as those of an IfCondition step.

The completion of a ForCondition loop is indicated by a Continue step (see "Continue Steps" on page 344). When the ForCondition loop ends, the primary method logic path continues.

To break the ForCondition loop, use the Break step ("Break Steps" on page 344) that goes back to the ForCondition step and then resumes the primary, **next** method path.

ForCondition Step Properties > General Tab

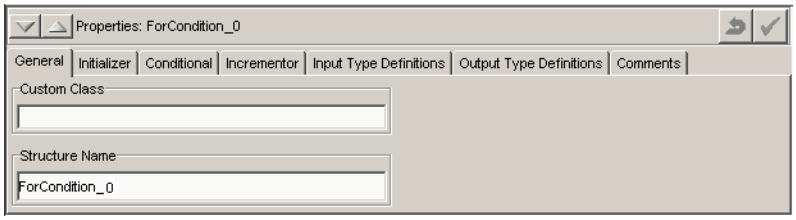


Figure 162. ForCondition Step Properties > General Tab

The **General** tab of the ForCondition Properties view offers the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112. This setting is optional.
Structure Name	The key value inserted into the return map

ForCondition Step Properties > Initializer Tab

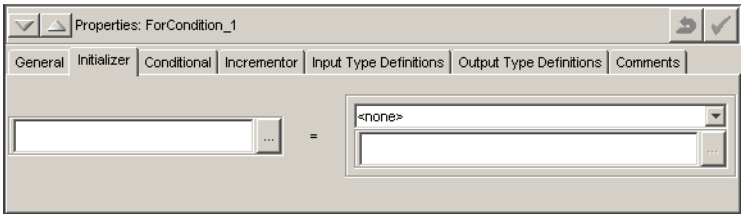


Figure 163. ForCondition Step Properties > Initializer Tab

The **Initializer** tab of the ForCondition step Properties view sets an initial value for the data source, by comparing the data source (a Variable or a Variable component) specified in the left box to the data source specified in the right box.

To specify a data source, choose one of the following:

- Type the data source's name in each box

Or:

- Click the ellipsis button (...) to display the **Choose a Data Source** dialog box, and then select the appropriate data source from the list.

To limit the contents of the right box to a **Variable**, **Current Date/Time** or **InternalString**, choose the appropriate value from the drop-down list located above the box.

ForCondition Step Properties > Conditional Tab

For a description of the ForCondition step Conditional tab, see "Working with Conditional Steps — Overview" on page 334.

ForCondition Step Properties > Incrementor Tab

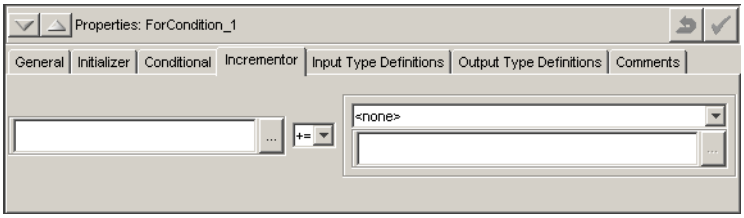


Figure 164. ForCondition Step Properties > Incrementor Tab

The **Incrementor** tab of the ForCondition step Properties view increments the value of the specified data source by an (int) amount.

ForCondition Step Properties > Input/Output Type Definitions tabs

The ForCondition step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.

The ForCondition step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

WhileCondition Steps

The WhileCondition step provides a comparison between two objects.

- If the evaluation of the condition is "true", the step(s) on the conditional or **branch** link are executed. This process loops until the Condition becomes "false".
- When the evaluation of the condition becomes "false", the primary method logic path is resumed.

All of the comparison operators are supported for the WhileCondition.

WhileCondition Step Properties

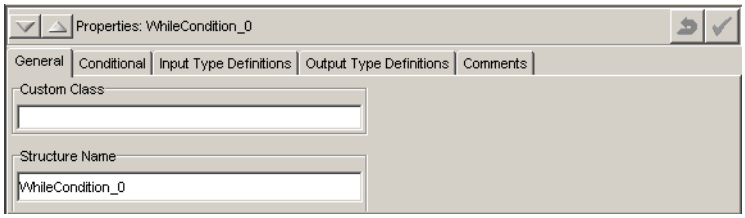


Figure 165. WhileCondition Step Properties > General Tab

> **Note:** The following description applies to the DoWhileCondition step properties as well.

- **General** tab — allows you to define the following:
  - **Custom Class** — the class to be used with this method step. For information on custom classes, see "Custom Classes" on page 112.
  - **Structure Name** — The key value inserted into the return map.
- **Conditional** tab — allows you to compare two objects of interest using a variety of comparators. For a detailed description, see "Working with Conditional Steps — Overview" on page 334

- **Input Type Definitions** tab — this tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.
- **Output Type Definitions** tab — this tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

DoWhileCondition Steps

The DoWhileCondition step executes the step(s) on the conditional or **branch** link, and then provides a comparison between two objects. The **branch** is executed once before the comparison is performed.

- If the evaluation of the condition is "true", this process loops until the condition becomes "false".
- When the evaluation of the condition is "false", the primary method logic path continues.

All of the comparison operators are supported for the DoWhile condition.

DoWhileCondition Step Properties

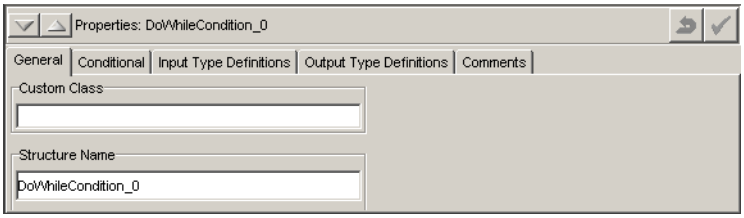


Figure 166. DoWhileCondition Step Properties > General Tab

The DoWhileCondition step properties are identical to the While Condition step properties. See "WhileCondition Step Properties" on page 342.

EndIf Steps

The EndIf step indicates the completion of an IfCondition **branch** path (see "IfCondition Steps" on page 338). After the EndIf step is reached, the primary, **next** method path is then resumed.

EndIf Step Properties

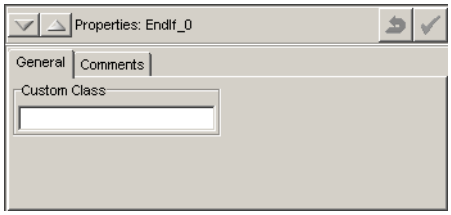


Figure 167. EndIfCondition Step Properties > General Tab

The **General** tab of the EndIf step Properties view allows you define the **Custom Class** that should be used with this method step. For more information about custom classes, see “Custom Classes” on page 112.

Continue Steps

The Continue step indicates the completion of a ForCondition, WhileCondition or DoWhileCondition loop (see “ForCondition Steps” on page 340, “WhileCondition Steps” on page 342 and “DoWhileCondition Steps” on page 343). After the ForCondition loop is repeated as many times as specified in the **General** tab of the ForCondition Properties view, the primary, **next** method path is resumed.

Continue Step Properties

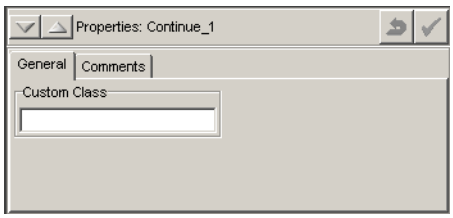


Figure 168. Continue Step Properties > General Tab

The **General** tab of the Continue step Properties view (Figure 168) allows you define the **Custom Class** to be used with this method step. For more information about custom classes, see “Custom Classes” on page 112.

Break Steps

The Break step breaks the ForCondition, WhileCondition or DoWhileCondition loop (see “ForCondition Steps” on page 340, “WhileCondition Steps” on page 342 and “DoWhileCondition Steps” on page 343), goes back to the step in question and then resumes the primary, **next** method path.

Break Step Properties

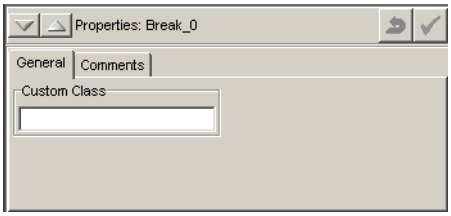


Figure 169. Break Step Properties > General Tab

The **General** tab of the Break step Properties view (Figure 169) allows you to define the **Custom Class** that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.

Set Steps

Set steps allow you to assign the value of a Variable (i.e. a data target) to a variety of objects, such as another Variable, a screen, a step, etc.

> **Note:** Alternatively, the value of a Variable can be assigned to a variety of targets using the Data Mapping Editor. For more information, see "Mapping a Source Data Type to a Target Data Type via a Variable" on page 271.

The value is set in the **Assignment** section of the Set step's **General** tab. There are several assignment options, including setting (the default option), adding or removing the Variable value to or from the object in question. The available options are a function of the selected objects' data types. Only valid options are enabled. For example:

- If the selected objects cannot be used to perform a valid assignment, the assignment drop-down list is disabled and the status bar describes the first error detected for this Set step (e.g. "Set_4 The left/right combination is invalid: One side is a leaf node, the other is not...").

> **Note:** You may still save invalid assignment settings by clicking **Apply**. In this case, a warning message may be displayed, depending on the severity of the error (for example: "The assignment of Break_0 to InternalString will probably produce no useful results at runtime").

- If the data target is not an array, only the **SET** assignment option is displayed.
- If the data target is an array, the assignment drop-down list is enabled and the **Add** and **Remove** options become available as well.

To configure the properties of a Set step, select it in either the Tree view or the Presentation view. The properties of the Set step are displayed in the Properties view.

Set Step Properties > General Tab

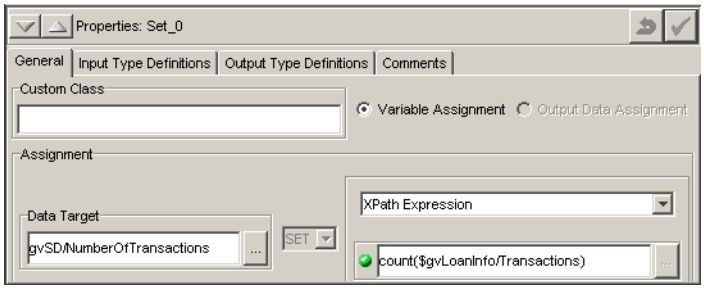


Figure 170. Set Step Properties > General Tab

The **General** tab of the Set step Properties view offers the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112. This setting is optional.
Variable Assignment	Set to add data to the specified data target.
Output Data Assignment	Set to add data directly to the method return map (version 3.5 compatibility methods only).
Assignment	Sets the conditions for assigning the selected **Data Target** (specified on the left hand side) to the selected object (specified on the right hand side). Proceed as follows: **1** Specify the **Data Target** (a Variable or Variable component). **2** Choose the type of object you wish to assign to this data target from the drop-down list on the right.

Option	Description
	3 Specify the object name. The following list describes how to specify the object name for each of the available object types: • **<none>**: This is the initial, unset state of the object name. You cannot leave it unset, it is not considered a valid selection. Change it to one of the other values. • **<null>**: Represents “no value” or “empty value”. • **Variable**: Type the name of the Global or Method Variable, or click the ellipsis button (...) to select it from the **Choose a Data Source** dialog box. • **Current Screen**: The current screen is automatically set as the value of the selected data target. • **Current Date/Time**: At runtime, the current date and time are automatically set as the values of the selected data target. • **Cursor Position**: The zero-based offset for the current cursor position on the current host connection screen. This choice is valid only if the map is a Character Mode map.

Option	Description
	• **XPath Expression**: Enter an XPath expression that includes a reference to a Variable. If the Data Target specified (on the left side of the dialog box) contains an array, and the XPath expression returns multiple values, the multiple values are mapped to the array elements. In an XPath expression, the first element in an array is element [1], not element [0]. MapMaker evaluates the XPath expression entered for proper syntax and to ensure the validity of any references to Variables. A colored ball icon appears to the left of the XPath expression. If the message line and the ball are red, it indicates a "hard" error that must be corrected. If the ball is yellow it indicates that the XPath expression cannot be fully validated and may or may not work at runtime; MapMaker accepts the statement as written. If the ball is green, it indicates that the XPath expression is valid. **Note:** You can verify the result of an XPath expression before runtime. For more information, see "Verifying the Result of an XPath Expression" on page 257. • **InternalString:** Enter a string value. To set a case sensitive comparison, set the **Case Sensitive** checkbox. • **InternalNumber**: Enter a number value. • **InternalBoolean**: Enter a boolean value. • **InternalDate**: Enter a date value. • **All map screens**: Select the appropriate screen. • **All method steps**: Select the appropriate step.

Option	Description
	Note: Object names must be spelled correctly. Otherwise, the text is colored red, the status bar indicates the problem, and the settings cannot be saved (clicking the **Apply** button reverts the properties to the last save).
4	Define the relationship between the **Data Target** and the selected object, or reference Variables to form the index, by selecting one of the following operators from the center drop-down list: • **SET**: The default operator, which replaces the value in the **Data Target** property with the value of the object specified on the right of the operator list. In an array context, the **Data Target** value is replaced at the specified array index. If the array is empty, the "NoValue" element is replaced at the first array index. • **ADD**: This operator is available only if the selected **Data Target** is an array or an element of an array. It combines the value of the **Data Target** property with the value of the object specified on the right. -If the **Data Target** specifies an array index, the new value is inserted into the array at the specified index. - If the **Data Target** does *not* specify an array index, the new value is appended to the end of the array. - If the array is empty, the new value is appended after the "NoValue" element in the first array index. - If the index supplied is greater than the number of elements in the array, empty values are inserted into the array to fill in the missing indices, and the new value is added to the end of the array at the specified index.

Option	Description
	• **REMOVE**: This operation is available only if the selected **Data Target** is an array or an element of an array. It allows you to perform the following operations: - Clear the entire array. - Delete a specified index of an arrayed data target (i.e. remove one row at a time), by using brackets. As the value of the **Data Target** is subtracted from the value of the selected object, the object type and name (specified on the right) are disabled.

Note: Users writing custom code may find it useful to know that when the right side object is of type `com.jacada.types.ValueObject`, the actual value assigned to the Variable is obtained by using the `getValue()` method on the object. This means that the internal value of the object is assigned to the left MapVar, *not* the actual object itself. The `MapVar` and `MethodObject` options are `ValueObject`. **InternalString** and **Current Screen** selections are assigned using the actual referent, not an internal field of the object (because they do not implement `ValueObject`).

Thread Steps

A Thread step allows parallel code execution within the method. When a Thread step is added to the method, a new branch is created, representing a thread of parallel code execution. Each Thread step represents one additional thread of execution. Threads are stopped and/or joined using a ThreadJoin step. Once the Thread is created and the **branch** link starts executing, the primary method logic path continues.

Thread steps have the following restrictions and limitations:

- Steps that access the host connection are not permitted on a Thread branch. For example, Traverse or Perform steps are not allowed.
- No inter-thread communication is provided through the user interface, although this can be added by writing custom code.
- An exception on the main Thread is always sent to the client, but an exception on a Thread branch is not. Use an OnFail step to handle Thread branch exceptions.

> **Note:** Other functionality can be added to the Thread branch using custom code, however the restriction regarding host access **must** be observed.

Only one Thread per Thread method step is allowed. If a persistent Thread from a previous method invoke is still running, a new instance will not be started. In that case, all steps on the **branch** link are ignored. The primary or **next** method logic path continues, regardless.

When a looping conditional step (For, While or DoWhile) is running in a Thread branch, it checks after every iteration to see if its containing thread has been interrupted by a ThreadJoin step, using the Stop action. If the Thread has been interrupted, the looping conditional step stops, regardless of the boolean state of its condition. This also means that a looping conditional step is not interrupted until it finishes its current loop.

Setting Thread Step Properties

To configure the properties of a Thread step, select it in either the Tree view or the Presentation view. The ThreadStep properties are displayed in the Properties view (Figure 171).

Thread Step Properties > General Tab

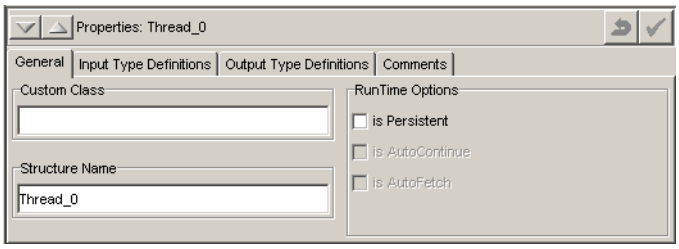


Figure 171. Thread Step Properties > General Tab

The **General** tab of the Thread step's Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Structure Name	The key value inserted into the return map.

Option	Description
RunTime Options	• **Is Persistent**: Indicates whether the thread's life may extend beyond the method invocation. • **Is AutoContinue**: Indicates that the method step attempts to continue automatically if a screen that the method step arrives at does not match the expected screen. Applies to Traverse and UserInteraction method steps only. For more information, see "AutoContinue Option" on page 303. • **Is AutoFetch**: Indicates that the AutoFetch option is set on the method step. Applies to Traverse, Perform and UserInteraction method steps only. For more information, see "AutoFetch Option" on page 302

Thread Step Properties > Input/Output Type Definitions Tabs

- The Thread step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.
- The Thread step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

ThreadJoin Steps

ThreadJoin steps provide the functionality to collect any threads initiated by Thread steps parallel code execution within the method. ThreadJoin automates the termination of parallel "subtasks".

Setting ThreadJoin Step Properties

To configure the properties for a ThreadJoin step, select it in either the Tree view or the Presentation view. The ThreadJoin step properties are displayed in the Properties view.

ThreadJoin Step Properties > General Tab

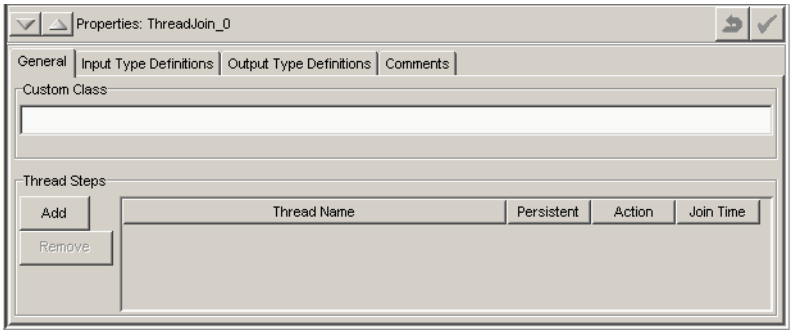


Figure 172. ThreadJoin Step properties > General Tab

The **General** tab of the ThreadJoin Step's Properties view (Figure 172) provides the following options:

Option	Description
Custom Class	The custom class that should be used with this method step. For more information about custom classes, see "Custom Classes" on page 112.
Thread Steps	Lists the Thread Steps to be joined and associated actions for individual Thread Steps
Thread Name	The name of the thread to join.
Persistent	When set, it indicates that a ThreadStep was created with the "IsPersistent" flag set to "True". This provides an easy way to identify whether or not the ThreadStep is persistent.
Action	• <none> - Ignore/do nothing with this thread. • join - Join the Thread with the other selected Threads based on the Join Time selection. • stop - Stop the Thread and join the Thread based on the Join Time selection (see "The Thread Step's Stop Action" on page 354).
Join Time	Number of milliseconds to wait to join the Thread with the other selected Threads. Default is 0, which means to wait indefinitely

Option	Description
Add	Adds one or more Threads to be joined.
Remove	Removes a highlighted Thread from the list of threads to be joined.
onFail link	The **onFail** link associated with this method step. For more information about **onFail** links, see "Method Step Links" on page 355.

ThreadJoin Step Properties > Input/Output Type Definitions Tabs

- The ThreadJoin step's **Input Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Input Type Definitions Tab" on page 294.
- The ThreadJoin step's **Output Type Definitions** tab is identical to the Start step's corresponding tab. For information on this tab, see "Start Step Properties > Output Type Definitions Tab" on page 294.

The Thread Step's Stop Action

It is important to note that the Thread step's Stop is merely a request. Since the `stop()` method on `java.lang.Thread` is deprecated, a Thread cannot be reliably forced to stop.

Stopping a Thread is mostly an issue for custom code, which should use the `getInterruptibleObject()` method of the `GBMethod` object interface to get an instance of the `com.jacada.types.Interruptible` object, and check the value of the `InterruptibleObject.isInterrupted()` method to see if the execution should be stopped. For more information on writing custom code to be run in a Thread Step, see the Javadoc of the `GBMethodStepThread.execute()` method.

EndThread Steps

An EndThread step is a terminator step, which indicates the completion of a Thread branch. The completed threads within the method are "collected" using a ThreadJoin step.

The EndThread step's properties consist of a custom class and a comment (Figure 173).

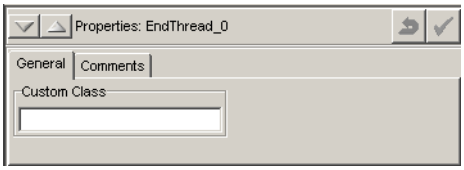


Figure 173. EndThread Step Properties > General Tab

Method Step Links

The method's steps are connected to each other manually, by drawing links between them.

Note: The following table describes the different types (modes) of links, as well as the Presentation view display options, which are included in the Link mode's tool bar.

Button	Description
	next: The default link mode, which proceeds to the next step in the method. This is the primary flow of the method logic.
	branch: An alternative link mode, which diverges from a Conditional or Thread type of step. • If the conditions specified in the step are true, the method follows this **branch** link. When the method reaches the end of the **branch** flow, it returns to the primary, **next** flow. • If the conditions specified in the step are false, the method skips the **branch** flow and follows the primary, **next** flow. For a description of the conditional steps, see "Method Steps" on page 288.

onFail: A backup link mode, used to handle situations in which a method step fails and the host application may be in an unknown state. **onFail** processing allows the service to proceed to another step, which returns the legacy application to a known state, or performs cleanup operations before exiting the method.
For example, when a legacy application receives unexpected input, it might clear the screen and display an error. In this case, you can create an **onFail** link that returns the application to the main menu.

Note: *The Interactive Mode onFail feature is not supported when the webMethods JI Service is deployed as a Resource Adapter.*

Auto Arrange: Aligns all connected method steps in sequence. For methods that do not contain conditional steps, the alignment is vertical.

Swap Content: Toggles between the method flow view and the Presentation view of the highlighted step.

Note: The two views can also be toggled using the space key.

Multiple-Mode Links

A single link may contain up to three different logical link modes, one on top of the other. All types of steps may have **next** and **onFail** link modes, while Conditional and Thread steps may also have a **branch** link mode. Each link mode is indicated by a label.

When the mouse is moved over a label, both the label and the link associated with it are bolded. When multiple edges appear on the same link, only one label can be selected at a time. For example, Figure 174 shows a multiple-edge link whose **onFail** label is selected.

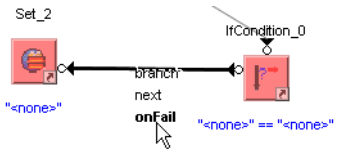


Figure 174. Multiple-Edge Link with onFail Label Selected

Double clicking the selected label (or its data mapping icon, if data mapping has already been configured for that link mode) opens its Data Mapping Editor, whose caption indicates the mode of the link and the names of the steps it connects. For example, double clicking the above onFail label (Figure 174) opens the Data Mapping Editor with the following caption (Figure 175):

Figure 175. Data Mapping Editor Caption

> **Note:** Alternatively, you can open the Data Mapping Editor for a specific link edge by right-clicking the link and selecting the appropriate edge from the shortcut menu.

When multiple link modes appear on the same link, but no particular label is selected (Figure 176), double clicking the link itself automatically opens the Data Mapping Editor for the link mode with the highest precedence available, in the following order:

1 **next**
2 **branch**
3 **onFail**

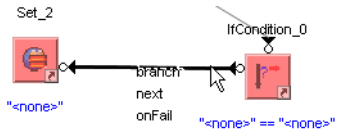


Figure 176. Selecting a Multiple-Mode Link Without Choosing a Specific Mode

Adding Links

Link lines are drawn in the following manner:

1 In the Presentation view, highlight the step from which you wish to create a link.
2 In the Link Edges tool bar, click the type of link you wish to create: **next** , **branch** or **onFail** .
3 Click and hold the left mouse button over the "link arrow" of the origin step (see Figure 177), and drag a link to the destination step.

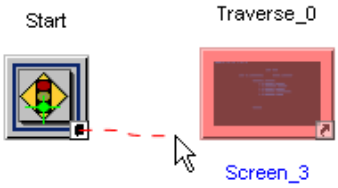


Figure 177. Linking Method Steps

> **Note:** Dropping the link on a blank space or on an inappropriate destination step (e.g. dragging a **branch** link from a step that is not conditional) causes the link operation to fail. For more information on linking restrictions, see "Method Step Restrictions" on page 359.

4 If there are other optional destinations (besides the current destination), the **Choose a Destination** dialog box is displayed (Figure 178).

5 Select the appropriate destination step and click **OK**.

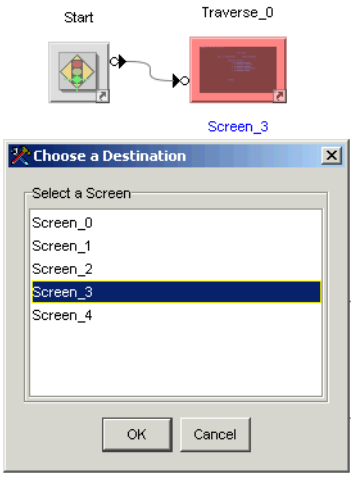


Figure 178. Choose a Destination Dialog Box

Once the two steps are properly linked, the following changes take place (Figure 179):

- The destination step is validated, as indicated by its color (for information on method color codes, see "Methods Presentation View Color Code" on page 361).
- The mode of the link (**next**, **branch** or **onFail**) is indicated.

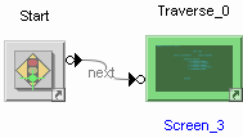


Figure 179. Linked Steps

Figure 180 shows an example method, whose steps are all properly linked. This method consists of three flows: the **next** flow, which is the primary flow; the **branch** flow, followed when the conditions specified in the **IfCondition_0** step are true; and the **onFail** flow, followed when the **IfCondition_0** step cannot be executed.

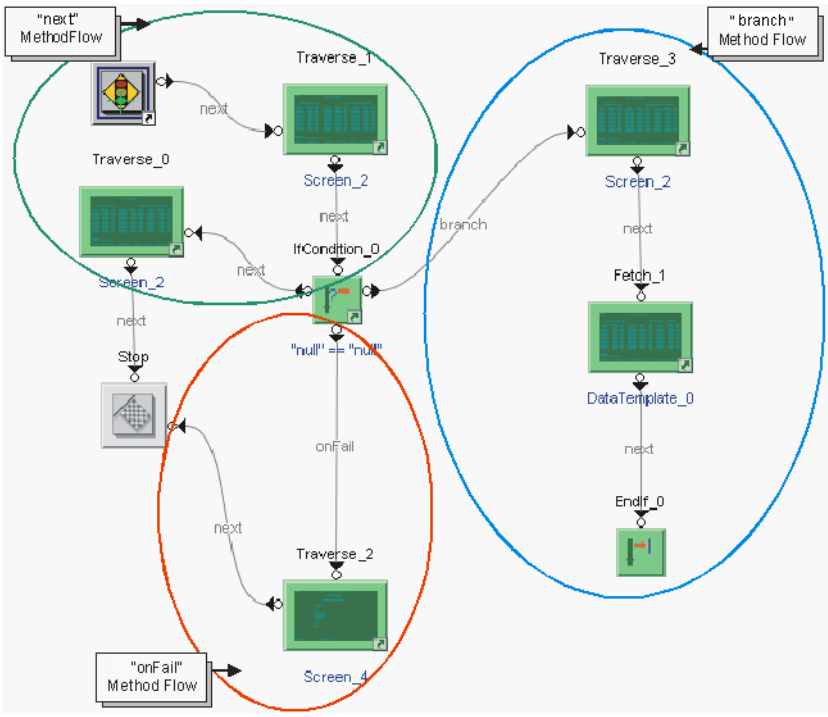


Figure 180. A Method with Three Flows: next, branch and onFail

Method Step Restrictions

The following method step restrictions may prevent steps from being linked:

- Screen-based steps require a “current screen” context. The first screen-based step in a method, or the first screen-based step after a conditional step, must be a Traverse or a Perform step.
- Screen-based steps are not permitted on Thread branches due to the possibility of concurrent host interaction from multiple threads of execution.
- ThreadJoin steps cannot join the same Thread they are running in.

Unlinked steps appear in red. When you properly link these steps into the method logic flow, they turn to green.

Deleting Steps or Links

To delete a step or a link, select the item in question in the Presentation view, right-click, and select **Delete**. Deleting steps and/or links may invalidate all or part the method logic flow. Alternatively, you may draw a link to the newly desired step, which automatically deletes the original link.

Outputs

Adding Outputs

To add an output, select the Outputs node of the relevant method in the Tree view, right-click and select **Add Output**. The new output is added to the Tree and its properties are displayed in the Properties view (Figure 181).

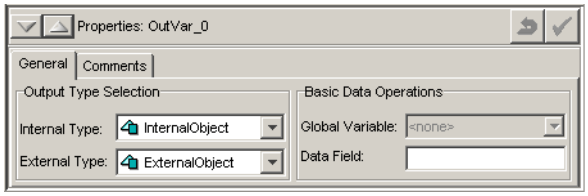


Figure 181. Output Properties > General Tab

The **General** tab of the Output Properties view provides the following options:

Option	Description
Output Type Selection	A version 4.x output is associated with two data types: • Internal Type — Choose from a list of IBEs. • External Type — Choose from a list of XBEs. Choose the corresponding output types from the **Internal Type** and **External Type** drop-down lists.
Basic Data Operations	A version 3.x output does not support version 4.x complex data structures. Instead, such an output is defined using a simple data source, i.e. a Global Variable. • **Global Variable** — a Global Variable whose type is **Internal Object** that is associated with this input. Choose the appropriate Global Variable from the drop-down list. Default: `<none>` • **Data Field** — Here you specify the data field from a Data Template or a Table Template, from which the value of the Global Variable is obtained.

Enabling and Disabling Outputs

Outputs can be enabled or disabled so that a generated service that uses this method does not include the output in its generated code. To disable an output, or to enable an output that has been disabled, right-click the output and select the appropriate option from the shortcut menu.

Additionally, all outputs can be enabled or disabled, by selecting the **Outputs** node of the Tree view and selecting **Enable All Outputs** or **Disable All Outputs** (respectively) from the shortcut menu.

Methods Presentation View Color Code

When working in MapMaker's **Methods** tab, the Tree view and Presentation view provide a convenient visual indication of the method's validity, using the following color code:

- Red — some of the method's components (steps, links etc.) are invalid. Examples of invalid items include:
  - Unlinked items.
  - Items without valid and/or required properties set. For example:
    - JClient3 steps require `host:port, service name`, and `method name` properties settings.
    - Screen-based steps, such as Traverse, without a valid screen set.
- Yellow — some of the method's components (steps, links, etc.) are incomplete. An example incomplete item is a step that has no **next** or **onFail** link.
- Green — the method is properly configured.

When items are invalid or incomplete, they are colored red or yellow (respectively) in the following places:

- Tree view — highlighting the method and its invalid steps
- Presentation view — framing the invalid steps, as well as the whole view.

In addition, the Status Bar at the bottom of the MapMaker window provides an error message that describes the problem (e.g. "Fetch_3 has no NEXT node...").

When the problems are fixed, the Tree view is no longer colored and the steps and frame of the Presentation view are colored green.

Keyboard Shortcuts and Search Options

To facilitate working in the **Methods** panel, use the following keyboard shortcuts and search options:

- Keyboard Shortcuts. When a step is selected in either in the Tree view or the Presentation view, its properties are displayed in the Properties view.

  When a screen-based step, such as Traverse, is selected in the Presentation view, the **Space** bar can be used to toggle between the Presentation view display of the **Methods** tab and that of the **Map** tab for that screen.
- Searching for Objects in the **Methods** tab. To find an object of interest in the **Methods** tab, right-click the root of the Methods Tree and choose **Find** from the shortcut menu.

  The **Search for Object** dialog box is displayed (Figure 182).

Figure 182. Search for Object Dialog Box

You can search for any of the following objects, by entering their full name:

- Methods
- Input variables
- Output variables
- Steps

Note: The search is case-sensitive and no wild cards are allowed.

When the object is found, it is highlighted in the Tree view. Methods and steps are also highlighted in the Presentation view.

Chapter 9. Debugging Methods

The **Debugger** enables you to locate and correct errors in a method by invoking and executing the code of a method from within MapMaker. There is no need to deploy a service to check methods, as the debug process assigns values as would be assigned to variables at runtime. Code execution can be paused at any point in order to:

- Identify how a method's transactions are being carried out. See "Step 4: Viewing the Connected Host" on page 379.
- Find and correct any errors contained in a method's variables. See "Step 5: Viewing and Editing Method Data" on page 383.

Note: The process of creating and modifying the method flow is described in "Managing Methods" on page 277, and cannot be performed from the **Debugger**.

This chapter describes both the general procedure for working with the **Debugger**, and the specific steps involved in debugging a method. This chapter contains the following sections:

- "General Debug Procedure" on page 363
- "Workflow for Debugging a Method" on page 364

General Debug Procedure

This section describes the general procedure for working with the **Debugger**.

To debug a method:

1. Select the required method in the **Method** tab. The method flow is displayed in the Presentation view, with all breakpoints enabled.
2. Disable the breakpoints for any steps and links in which you are not interested. Leave the breakpoints enabled for the steps and links at which you want to pause code execution, and take a closer look. See "Step 2: Enabling and Disabling Breakpoints" on page 377.
3. Click **Start**. The following occurs:
   - The **Host Screen Viewer** is displayed as a task bar button, and as an option in the **Window** menu.

> **Note:** If you have already started the **Host Screen Viewer** in the current Debug session, that instance of the **Host Screen Viewer** is used by the **Debugger**.

- The **Run** and **Start** buttons are disabled.
- The Debug cursor is first displayed over the Start step, and then moves through all steps and links in the method, filling the completion bar as it completes each step. The Debug cursor then stops at the first enabled breakpoint.

4. Use the **Host Screen Viewer** to view the results of the method as it executes, or to manually step through the actions in a screen-based step. See "Step 4: Viewing the Connected Host" on page 379.
5. Access the **TypeDataModel Viewer** to view or edit the data used in the method. See "Step 5: Viewing and Editing Method Data" on page 383.
6. When you have finished using the **Host Screen Viewer** and the **TypeDataModel Viewer**, click the **Continue** button to resume code execution. The Debug cursor moves through the remaining steps and links, pausing at the next enabled breakpoint in the method.
7. Click **Stop** to stop debugging the method at any point during method execution.

   When you stop debugging a method during its execution, the **Host Screen Viewer** displays the last screen it encountered. Restarting the same method from the beginning, continues code execution from the last screen displayed in the **Host Screen Viewer**, and not from the first screen of the method as expected. You can restart the code execution from the first screen of the method by disconnecting from and then reconnecting to the host.
8. When the method completes, the **Run** and **Start** buttons are enabled.
9. You can rerun the same method using the same or different breakpoints, or select a new method from the **Method** tab.

Workflow for Debugging a Method

This section contains a detailed workflow for debugging a method. The workflow contains the following steps:

- "Step 1: Accessing the Debugger" on page 365
- "Step 2: Enabling and Disabling Breakpoints" on page 377
- "Step 3: Defining Debug Cursor Properties" on page 378
- "Step 4: Viewing the Connected Host" on page 379
- "Step 5: Viewing and Editing Method Data" on page 383
- "Step 6: Watching Values Update" on page 389

Step 1: Accessing the Debugger

This step describes how to access the **Debugger** and how to display a method for debugging. It also describes each component in the **Debugger** interface.

To access the Debugger:

1 In MapMaker, open the map containing the method you want to debug.

2 Click the **Debugger** button on the toolbar.

The MapMaker interface changes from the Design environment interface to the **Debugger** interface, and lists the methods contained in the currently loaded map:

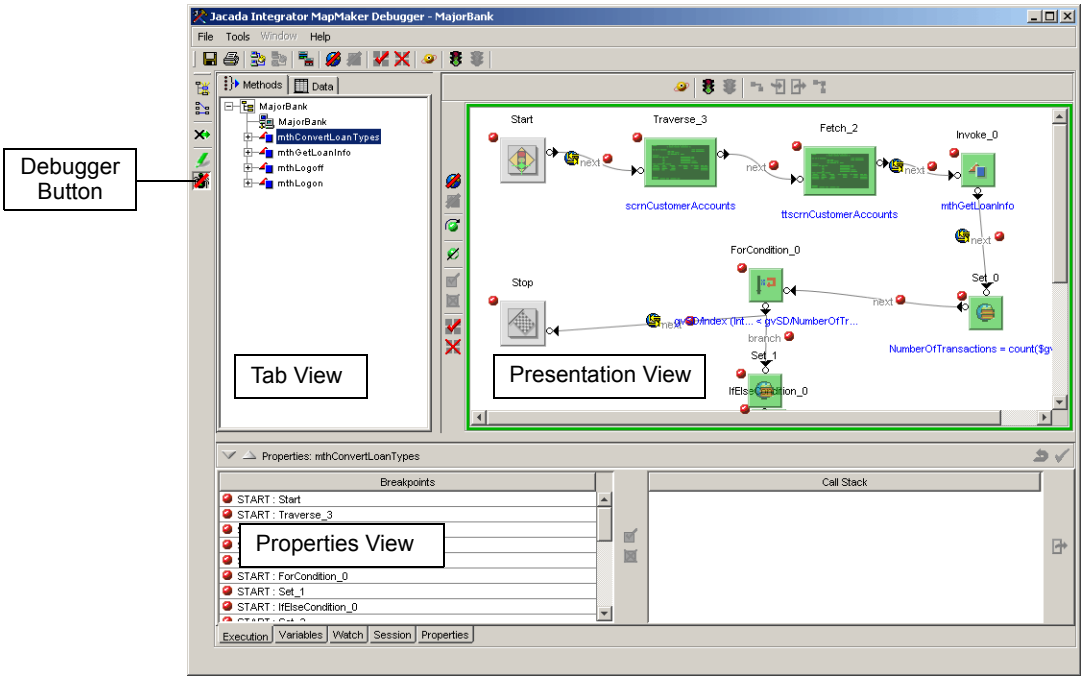


Figure 183. The Debugger Interface

3 From the **Method** tab in the **Tab** View, select a method to debug.

The selected method's method flow and properties are displayed in the Presentation view and Properties view respectively.

Debugger Views

The **Debugger** interface consists of the following views:

- Tab view
- Presentation view
- Properties view

Tab View

The Tab view is located on the left-hand side of the **Debugger**, and consists of the **Methods** tab and the **Data** tab.

Tab	Description
Methods tab	Displays a tree containing all methods defined in the currently loaded map, as well as the steps included in each method.
Data tab	• **Upper pane**: Displays the entire structure of the currently selected Global, Method, Input or Output variable. This variable is selected in the **Variables** tab of the Properties view. See "Variables Tab" on page 369. • **Lower pane**: Displays the data contained in the selected variable.

The **Data** tab contains a smaller version of the **TypeDataModel Viewer**, providing an environment in which to view and edit the content of a method's variables, and to correct any errors found. See "Step 5: Viewing and Editing Method Data" on page 383.

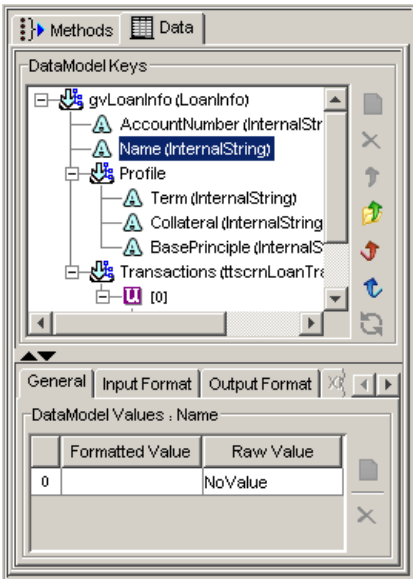


Figure 184. The Debugger Data Tab

Presentation View

The Presentation view is located on the right-hand side of the **Debugger**. It is similar to the Presentation view in the MapMaker Design environment in that it illustrates the flow of the method selected in the **Methods** tab of the Tabs view. The Presentation view also illustrates the method's breakpoints and Debug cursors, which are specific to the **Debugger**.

The Presentation view contains the following components:

- Breakpoints
- Debug cursors
- Debugger toolbars

Breakpoints Breakpoints are points in a method at which code execution is paused, enabling you to view the data contained in the method's variables and correct any errors, as described in "Step 5: Viewing and Editing Method Data" on page 383.

A breakpoint is situated at the start of every step and at the start of every link in a method, and is indicated by a red ball , signifying a stop in code execution. When a method is invoked, method execution is paused each time a breakpoint is encountered. A clock icon is displayed next to the method in the **Method** tab tree, indicating this pause. When a method is paused, you can use the **TypeDataModel Viewer**, as described in "Step 5: Viewing and Editing Method Data" on page 383.

If a pause in code execution is not required at a certain step or link, a breakpoint can be disabled. A disabled breakpoint is indicated by a green ball , signifying the ongoing execution of the code.

The location of breakpoints cannot be modified, but you can enable and disable as many breakpoints as necessary. See "Step 2: Enabling and Disabling Breakpoints" on page 377.

Debug Cursors Debug cursors indicate the current location of the method logic while it is executing in the **Debugger**. A Debug cursor consists of two brackets that surround the name of the current step or link, and a vertical completion bar.

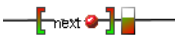


Figure 185. The Debug Cursor

At the start of a Debug process, the Debug cursor is located on the method's Start step. Clicking the **Step** button executes the Start step. When the Start step completes execution, the Debug cursor moves to the link connecting the Start step with the next step. At this point, clicking the **Step** button executes the link. This process is repeated for every step and link until the method is completed. When a Debug cursor pauses at a breakpoint, the vertical completion bar is filled

according to the percentage of the step or link completed. For example, if a Traverse step is 33% through traversing its screens, the Debug cursor shows 33% of the bar filled.

Note: In the rare event that a method step fails to complete during debugging, you may need to stop MapMaker via the Windows Task Manager.

An additional Debug cursor is created for each Conditional, Thread, Loop and Invoke step in a method. A list of Debug cursors in the current method is displayed in the **Session** tab of the Properties view. Selecting a Debug cursor in this list displays its user-definable properties below the list. See "Session Tab" on page 371.

Note: When debugging a method that runs two threads in parallel, there is no guarantee that the threads will finish at the same time. For this reason, the stop action on a ThreadJoin step may act as a join action, and wait for the other thread to finish.

Properties View

The Properties view is located at the bottom of the **Debugger**, and describes code execution and variable properties for the displayed method. The Properties view contains the following tabs:

Tab	Description
Execution tab	Contains a full list of the breakpoints in the currently selected method, and the current execution trace. See "Execution Tab" on page 369.
Variables tab	Lists all variables in the scope of the current **Debugger** execution, and enables you to edit the data contained in the variables. See "Variables Tab" on page 369.
Watch tab	Displays your chosen variables so that you can observe their values as they are updated in "realtime". See "Watch Tab" on page 370.
Session tab	Lists the current Debug cursors and the log output of the Debug process stack. See "Session Tab" on page 371.

Tab	Description
Properties tab	Displays non-editable versions of the Property view tabs that are displayed in the MapMaker Design environment, for the selected step or link. See "Properties Tab" on page 372.

Execution Tab The **Execution** tab displays the location of the debug process within the current method execution, in terms of the current breakpoint, and within the list of methods that have been invoked by the source method so far. The **Execution** tab consists of the lists:

- **Breakpoints**: Lists the breakpoints in the currently selected method, and highlights the current breakpoint. Click **Disable** to disable, or **Enable** to enable the selected breakpoint.
- **Call Stack**: Displays the current execution trace, listing the currently invoked methods.

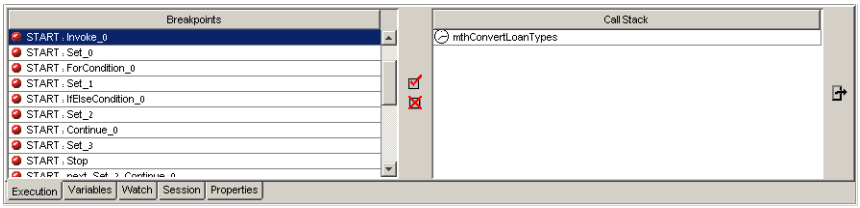


Figure 186. The Execution Tab

When a method yields to another method's Debug process, that is, when an Internal Invoke step is executed, the following occurs:

- The **Breakpoint** list changes to show the breakpoints of the new method.
- The **Call Stack** shows the Debug process hierarchy in a stack format, with the current method at the top of the list, and the original method at the end of the list.
- An hourglass icon is shown next to the original method in the tree, signifying that the original method is yielding, that is, waiting for the invoked method to complete.

Click **Step Out** in the **Execution** tab or in the Presentation view to step out of the method currently being invoked, and return to the previous method in the stack.

Variables Tab The **Variables** tab lists all variables in the scope of the current method. It contains the following lists:

- **Inputs**: Lists the Input variables for the current method.
- **Global**: Lists the Global variables for the current method.
- **Method**: Lists the Method variables for the current method.

- **Step Context**: Lists the Output variables for the step selected in the Presentation view.
- **Outputs**: Lists the Output variables for the current method.

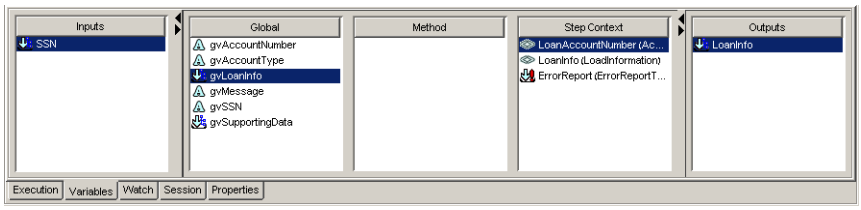


Figure 187. The Variables Tab

The following actions can be performed from the **Variables** tab:

Action	Operation
View and edit the data contained in the Type Data Model of the variable	Right-click a variable in any list, and select **Edit Data** to display the **TypeDataModel Viewer**. This enables you to view the data and correct any errors. See “Step 5: Viewing and Editing Method Data” on page 383.
Add the variable to the **Watch** tab	Right-click a variable in any list, and select **Add to Watch** to observe the variable’s value as it is updated in “realtime” in the **Watch** tab. See “Watch Tab” on page 370.

Watch Tab The **Watch** tab lists your chosen variables so that you can observe their values as they are updated in “realtime” by the method execution. Add variables to the **Watch** tab via the **TypeDataModel Viewer** or the **Variables** tab.

Select a method in the **Method** tab to display your chosen variables for that method in the **Watch** tab. References to Global variables or XPath expressions are always displayed in the **Watch** tab regardless of the method selected.

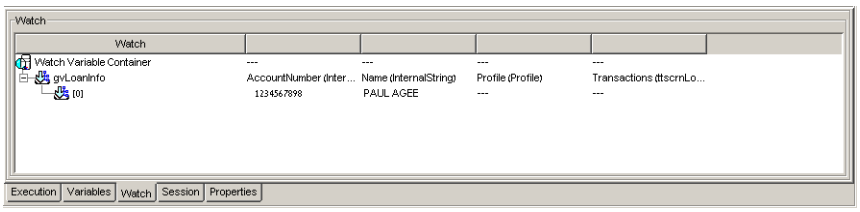


Figure 188. The Watch Tab

The **Watch** tab displays the root node of the variable on the left. If the variable is a complex structure, then its field names are displayed in the columns to the right of the root node.

If a structure in the **Watch** tab contains a single [0] node, it signifies either that the variable is an array with a single element, or that the variable is not an array. The [0] refers to the index in which the data is stored in the Map List Map format. Two or more index nodes in a structure signify that the variable is definitely an array, and the index numbers of the nodes correspond to the Map List Map index locations at which the data of each element is stored.

The following actions can be performed from the **Watch** tab:

Action	Operation
Display a field's path	Right-click a field name, and select **Drill Down** to display the field's path in the **Watch** tab. The columns display the nodes under the selected variable.
Add an XPath expression to the **Watch** tab	Right-click the variable's root node, and select **Add XPath** to display the **XPath Evaluator**, which is used to add an XPath expression to the **Watch** tab. See "Verifying the Result of an XPath Expression" on page 257.
Edit XPath expressions	Right-click an XPath expression in the tree. Select **Edit XPath** to display the **XPath Evaluator**. Edit the selected XPath expression. See "Verifying the Result of an XPath Expression" on page 257.

Selecting a variable in the **Watch** tab displays the variable's entire structure in the **Data** tab, and allows you to edit the variable in the same way as the **TypeDataModel Viewer**. See "Step 5: Viewing and Editing Method Data" on page 383.

Session Tab The **Session** tab enables you to define Debug cursor properties and to view the log output of the Debug process stack.

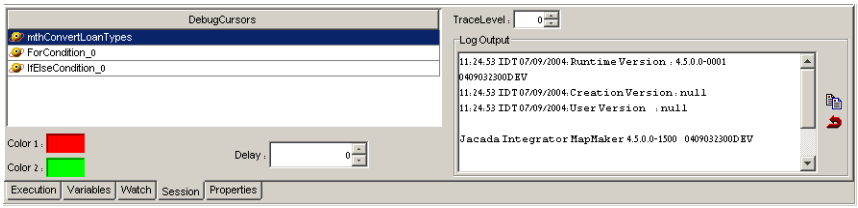


Figure 189. The Session Tab

Debug Cursors: Lists the Debug cursors in the current method. Select a Debug cursor in this list to display its user-definable properties below the list. See "Watch Tab" on page 370.

For more information about Debug cursors themselves, see "Debug Cursors" on page 367.

Log Output area: Displays the log output of the Debug process stack, which is the same text that would be output to the service log in actual runtime. The following functions can be performed from the **Log Output** area:

Operation	Description
Copy the log output text	Select the log output text and click **Copy** to copy the text to the system clipboard for pasting into a text editor.
Clear the log output text	Click **Clear** to remove all text from the log output area.

Trace Level field: Defines the trace level of the log output. This can be set to any value from **0** through **9**, with **0** representing the most basic logging and **9** representing the most verbose logging.

Properties Tab The **Properties** tab displays non-editable versions of the MapMaker Design environment Properties tabs, enabling you to view the properties of the object selected in the Presentation view.

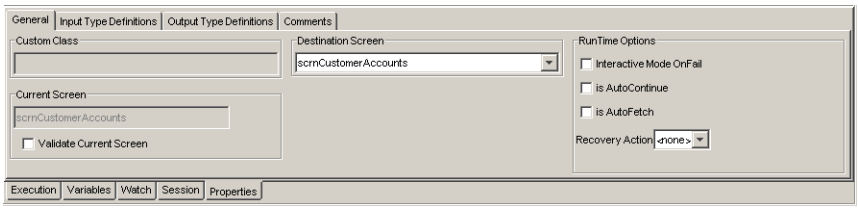


Figure 190. The Properties Tab

Debugger Menus

The **Debugger** menu bar contains the **File**, **Tools** and **Help** menus that list most of the MapMaker Design environment options. See "MapMaker Menus" on page 86.

The **Debugger** menu bar also contains the **Window** menu.

Window Menu

The **Window** menu is enabled when a host connection is established, and lets you switch between each open window in the **Debugger** interface. Open windows can be the **Host Screen Viewer** and as many instances of the **TypeDataModel Viewer** as are open.

Option	Description
JTerm - Connected to host, port *x*, session id = *x*, mode = *x*	Displays the **Host Screen Viewer** window. See "Step 4: Viewing the Connected Host" on page 379.
TypeDataModel Viewer - *variable name*	Displays the **TypeDataModel Viewer** window, containing the specified variable. See "Step 5: Viewing and Editing Method Data" on page 383.

Debugger Toolbars

The **Debugger** interface contains the following toolbars:

- The Method Flow toolbar is located at the top of the Presentation view. See on page 374.
- The Variables and Breakpoints toolbar is located at the left of the Presentation view. See on page 376.

Method Flow Toolbar

The Method Flow commands and their corresponding toolbar buttons are listed in the following table:

Button	Action	Description
	Run	Runs the method ignoring all breakpoints and without displaying Debug cursors. This is useful when the contents of a method are not of interest, such as an initialization method that must be run before the second method in a service can be debugged. A host connection is required to run a method. Clicking **Run** establishes a host connection if one is not already established.
	Start	Starts debugging the selected method by executing the code and pausing at the first enabled breakpoint. A host connection is required to perform debugging. Clicking **Start** establishes a host connection if one is not already established.
	Stop	Stops debugging.
	Step	Executes a single method step or link and pauses at the next step or link whether the step's breakpoint is enabled or disabled.

Button	Action	Description
	Step Into	Steps into the selected composite step (Conditional, Thread, Loop or Internal Invoke step) or link when method execution is paused. If the step is an Internal Invoke step, the **Debugger** executes and displays the actual method invoked. If the step is an External Invoke step, the **Debugger** executes the external method, but does not display the external method. When the **Debugger** executes an internal or external method, • An hourglass icon is shown next to the original method in the tree, signifying that the original method is yielding, that is, waiting for the invoked method to complete. • The **Call Stack** in the **Execution** tab shows the Debug process hierarchy in a stack format, with the current method at the top of the list, and the original method at the end of the list. If the current step is not a composite step, the **Step Into** action behaves as a **Step** action.

Button	Action	Description
	Step Out	Steps out of the nearest parent composite step (Traverse, Conditional, Thread, Loop or Internal Invoke step) or link, and returns to the previous link or step. Stepping out of an invoked method, returns the Debug process to the method from which it was invoked. Stepping out of a link, moves to the next step. If the last action was: • **Step Into:** The **Step Out** action moves to the next step. • **Step:** The **Step Out** action moves to the method from which the current method was invoked. If the current method is the original method, the **Step Out** action resumes and completes the method without stopping at breakpoints.
	Continue	Resumes the execution of a method after a breakpoint. The method runs until another breakpoint is encountered or the method finishes.

Variables and Breakpoints Toolbar

The Variables and Breakpoints commands and their corresponding toolbar buttons are listed in the following table:

Button	Action	Description
	Clear Global Variables	Clears all Global variable data from the last or current Debug session, and returns the data to the default values. Global variable data is persistent between **Debugger** sessions, so when a Map is saved, any data that was present on the Global variable is also saved and displayed the next time the Map is opened.

Button	Action	Description
	Clear Method Variables	Clears all Method variable data from the last or current Debug session, and returns it to the default data. Method variable data is automatically cleared every time the Start step is invoked on the method.
	Skip Disabled Breakpoints	Hides the Debug cursor as it traverses the method on those steps that have disabled breakpoints. The steps with disabled breakpoints are still invoked but there is no graphical indication of the individual invocations. This speeds up the execution of the method to the next enabled breakpoint.
	Hide Disabled Breakpoints	**Hides the icons for disabled breakpoints, in order to reduce clutter in the Presentation view.**
	Enable Breakpoint	Enables the breakpoint selected in the Presentation view.
	Disable Breakpoint	Disables the breakpoint selected in the Presentation view.
	Enable All Breakpoints	Enables all breakpoints in the currently displayed method.
	Disable All Breakpoints	Disables all breakpoints in the currently displayed method.

Step 2: Enabling and Disabling Breakpoints

Breakpoints are points in a method at which code execution is paused, enabling you to view the data contained in the method's variables and correct any errors. See "Step 5: Viewing and Editing Method Data" on page 383. The following procedures describe how to enable and disable these breakpoints.

The **Debugger** must be stopped, or paused at a breakpoint in order to enable or disable breakpoints.

To enable a breakpoint:

1. Click the required disabled (green) breakpoint in the Presentation view.

   The selected breakpoint is highlighted in the **Breakpoints** list on the **Execution** tab.
2. Click the **Enable Breakpoint** button either in the Variables and Breakpoints toolbar, or in the **Execution** tab.

   The selected breakpoint is enabled, and changes from green to red in the Presentation view.

To disable a breakpoint:

1. Click the required enabled (red) breakpoint in the Presentation view.

   The selected breakpoint is highlighted in the **Breakpoints** list of the **Execution** tab.
2. Click the **Disable Breakpoint** button either in the Variables and Breakpoints toolbar, or in the **Execution** tab.

   The selected breakpoint is disabled, and changes from red to green in the Presentation view.

See “Breakpoints” on page 367.

Step 3: Defining Debug Cursor Properties

Each Debug cursor is comprised of two colors which are painted with a gradient brush from the top left to the bottom right of the Debug cursor’s brackets and completion bar. The Session tab enables you to assign a different pair of colors to each Debug cursor in order to visually differentiate between multiple Debug cursors in a method.

In addition, you can control the speed at which the Debug cursor runs through the method so that you can visually follow the method flow.

To define Debug cursor properties:

1. Select the **Session** tab.
2. From the **Debug Cursors** list, select the required Debug cursor. The selected Debug cursor’s properties are displayed in the fields below the list.
3. Double-click the **Color 1** field. The **Change Color** dialog box is displayed.
4. Adjust the color settings as required, and click **OK** to close the **Change Color** dialog box.
5. Repeat steps 3 and 4 for the **Color 2** field.
6. In the **Delay** field, enter the amount of time, in milliseconds, that the Debug cursor waits before moving to the next object in the method logic.

See “Session Tab” on page 371.

Step 4: Viewing the Connected Host

The **Host Screen Viewer** provides you with a view of the current state of the connected host. In addition, the **Host Screen Viewer** enables you to:

- View each step in a screen-based method step in the host application. See "Stepping Through a Screen-Based Step" on page 380.
- Interact with the host application in the usual way, while the **Debugger** is paused.

Accessing the Host Screen Viewer

The **Host Screen Viewer** is started when you click either the **Run** or the **Start** button in the **Debugger** toolbar, and is displayed as a task bar button. Display the **Host Screen Viewer** either from the **Window** menu, or by clicking the task bar button labeled **JTerm - Connected to host, port x, session id = x, mode = x**.

Figure 191. The Host Screen Viewer

The **Host Screen Viewer** consists of the following components:

Component	Description
Screen View	Displays the current host screen with the results of the screen-based steps, and the actions associated with these steps, as they are executed in the connected host application.
Toolbar	Contains the **Step**, **Step Into**, **Step Out** and **Continue** buttons described in "Method Flow Toolbar" on page 374. The **Step** action executes the actions in the **Action List** as a whole. The **Step Into** action steps through each individual action in the list. **Step Out** and **Continue** run the rest of the actions without single stepping. The toolbar also contains the **Screen Compare** button, that enables you to compare the current host screen with the set of known screens and snapshots contained in current Map. See "Comparing Screen Images" on page 171.
Action List	Lists all traversals contained in the currently selected screen-based step. If the current method object is not a screen-based step, this list is empty.
Action Details	Contains the input operations associated with the currently selected traversal in the **Action List**. The AID Key and screen name associated with currently selected action are displayed below the **Action List** pane.

Stepping Through a Screen-Based Step

The **Host Screen Viewer** enables you to manually step through each screen traversal in a screen-based step, as it is executed in the host application. This enables you to isolate the cause of any problems by viewing each detail of the traversal individually, and at your own pace.

In order to isolate each detail of a traversal, the **Host Screen Viewer** splits the traversal process into two distinct steps, each of which can be viewed individually in the host screen. The **Step Into** button must be clicked once for each of these steps:

- **Step 1:** The first click sends the data defined in the screen-based step to the current host screen, at which time you can verify that the correct information is entered into the correct fields.
- **Step 2:** The second click activates the AID Key. The AID Key gathers all the information and sends it to the host application. This results in the traversal to the next screen specified in the screen-based step, at which time you can verify that the correct screen is reached.

Sometimes, a traversal contains an AID Key but no data. In this case, the **Step Into** button needs to be clicked once only.

The following procedure demonstrates how to step through individual traversals contained in a Traverse step. The Traverse step used in this example is from the **mthConvertLoanTypes** method created in the *webMethods JI Tutorial*, and it contains the following traversals:

1. From **scrnLogon** to **scrnMainMenu**
2. From **scrnMainMenu** to **scrnCustomerAccountsLookup**
3. From **scrnCustomerAccountsLookup** to **scrnCustomerAccounts**

To step through a screen-based step:

1. Ensure that the method has an enabled breakpoint at the screen-based step in question, in this case the **Traverse_3** step.
2. Advance the **Debugger** to the **Traverse_3** step.

   This can be achieved in either of the following ways:
   - Click **Step** at each enabled breakpoint, until the **Traverse_3** step is reached.
   - Disable all breakpoints before the **Traverse_3** step, and click **Start** to start debugging and automatically pause at the **Traverse_3** step.
3. Click the **JTerm** task bar button. The **Host Screen Viewer** is displayed, in which:
   - The **scrnLogon** screen is displayed in the Screen view.
   - The Traverse step's three individual traversals are listed in the **Action List**.
   - The first action in the **Action List**, and the first item in the **Action Details** list are highlighted.
   - The AID Key and screen name associated with the first action are displayed below the **Action Details** pane.

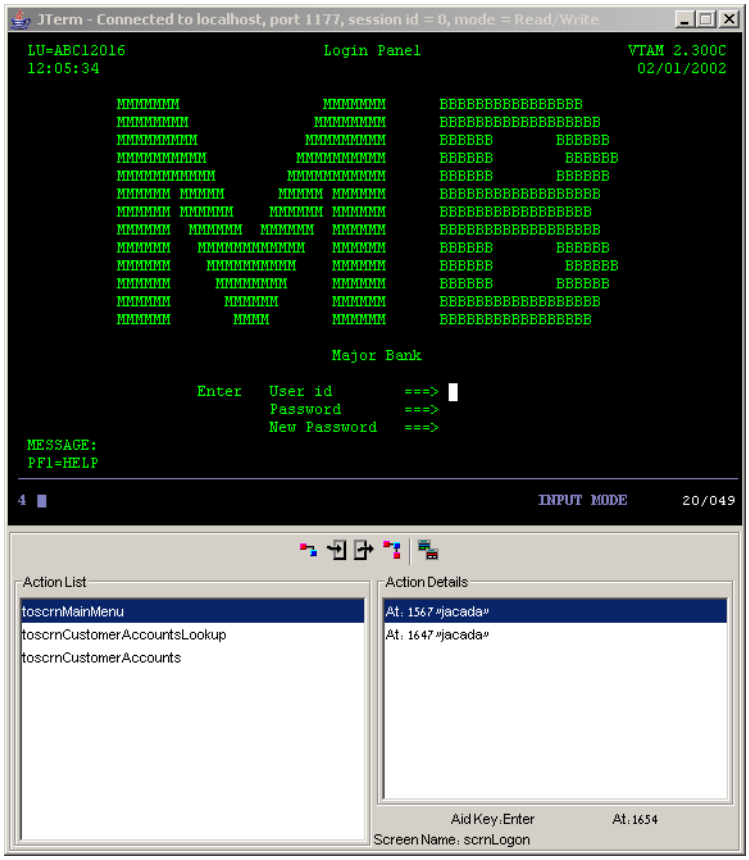


Figure 192. The Host Screen Viewer Executing a Traverse Step

4 To perform the traversals in the **Action List**, shown in Figure 192, proceed as follows:

Traversal	Action	Result
scrnLogon to scrnMainMenu	Click **Step Into**	The action details of the first action in the list are performed. In this case, typing the User Id and Password.
	Click **Step Into**	The <Enter> AID Key is sent to the host, and the traversal to the scrnMainMenu screen is performed.
scrnMainMenu to scrnCustomerAccountsLookup	Click **Step Into**	Option number **4** is entered in the **Option** field of the host screen.

Traversal	Action	Result
	Click **Step Into**	The <Enter> AID Key is sent to the host, and the traversal to the scrnCustomerAccountsLookup screen is performed.
scrnCustomerAccountsLookup to scrnCustomerAccounts	Click **Step Into**	The value contained in the **gvSSN** Global variable is entered in the **SOC SEC #** field in the host screen.
	Click **Step Into**	The <Enter> AID Key is sent to the host, and the final traversal to the destination screen, scrnCustomerAccounts, is performed.
	Click **Step** or **Step Out**	The Debug cursor moves to the next object in the method flow.

Step 5: Viewing and Editing Method Data

The **TypeDataModel Viewer** enables you to view and edit method data in order to correct any errors contained in a method's variables. The **TypeDataModel Viewer** is used while code execution is either paused at a breakpoint or stopped.

Accessing the TypeDataModel Viewer

Right-click a variable in any list of the **Debugger Variables** tab, and select one of the following options from the shortcut menu:

- **Edit Data**: Uses a shared **TypeDataModel Viewer** window. The previous variable viewed, if any, is no longer displayed in this shared window.
- **Edit Data in New Panel**: Displays a new instance of the **TypeDataModel Viewer** for the selected variable. Enables multiple instances of the **TypeDataModel Viewer** to be open simultaneously.

The **TypeDataModel Viewer** opens, displaying the selected variable:

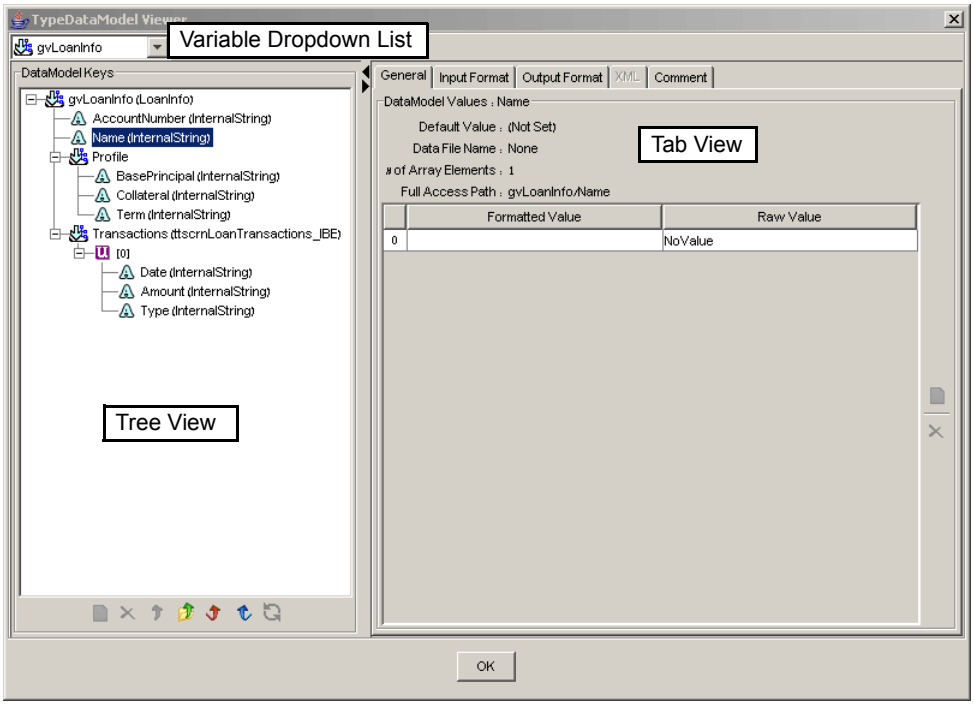


Figure 193. The TypeDataModel Viewer

A smaller version of the **TypeDataModel Viewer** is available in the **Debugger Data** tab.

TypeDataModel Viewer Components

The **TypeDataModel Viewer** consists of the following components:

- "Variable Dropdown List" on page 384
- "Tree View" on page 385
- "Tab View" on page 386

Variable Dropdown List

The **Variable** dropdown list is located in the top-left corner of the **TypeDataModel Viewer**, and enables you to select any variable in the currently selected variable category. The currently selected variable category is the **Variable** tab list from which the **TypeDataModel Viewer** was accessed.

Tree View

The Tree view is located on the left-hand side of the **TypeDataModel Viewer**, and displays the entire structure of the variable selected in the **Variable** dropdown list. It also contains a toolbar with the following buttons:

Button	Name	Function
	Add	Adds a new array element at the selected node. • If the selected node is the array itself, the **Add** action appends a new element to the end of the array. • If the selected node is an element in the array, the **Add** action inserts a new element directly before the selected element in the array.
	Remove	Removes the selected array element.
	Clear	Clears the values of the selected node and all its children.Values are set to **NO_VALUE**.
	Clear All	Clears all values from and removes any new array indexes that have been added to the structure currently displayed in the **TypeDataModel Viewer**. Values are set to **NOT_PRESENT**.
	Reset	Returns the currently displayed Type Data Model to its default state. That is, removing any new array indexes and setting all values to **NO_VALUE**.
	External TypeDataModel View	Toggles between an Input or Output variable's internal and external views.
	Reload	Reloads the default data for the Type Data Model, returning all values to their defaults.

Tab View

The Tab view is located on the right-hand side of the **TypeDataModel Viewer**, and displays the following tabs:

Tab	Description
General	Displays the details of the currently selected node in the Tree view. See “General Tab” on page 386.
Input Format and Output Format	View and edit the different formats for the inputs and outputs of certain IBE and XSD data types. See “Formatting Data Types” on page 236.
XML	Displays the content of the XML file that would be sent to the client for the selected Input or Output variable. See “XML Tab” on page 388.
Comment	Displays the comments that were written when the selected data type was created. See “Comment Tab” on page 388.

General Tab

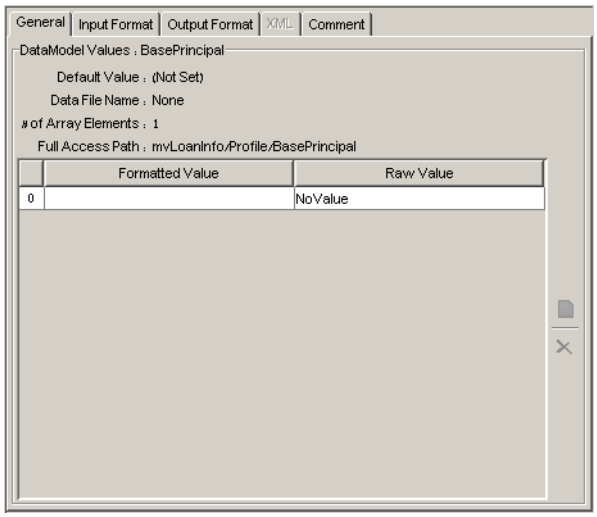


Figure 194. The TypeDataModel Viewer General Tab

The **General** tab contains the following information:

Field	Description
Default Value	The default value of the currently selected node in the tree.
Data File Name	The path and file name of the .XML file used to populate the Input variable. This field reads “None” when the Business Entity is not an XSD External Business Entity.
# of Array Elements	The number of array elements for the currently selected node. If the node contains indexes 0-4, the number of array elements is 5.
Full Access Path	The path from the root of the tree to the currently selected node.

The **General** tab also contains a table that displays the value of the currently selected node:

- The first column displays the array index of the row in the table.
- **Formatted Value**: The value after formatting, if any, has been applied. Double-click the **Formatted Value** field to edit the value of the selected node.
- **Raw Value**: The value before formatting, if any, has been applied.

XML Tab

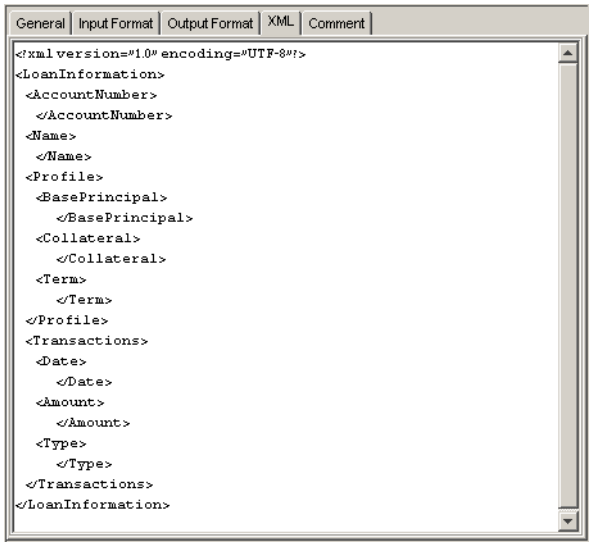


Figure 195. The TypeDataModel Viewer XML Tab

The **XML** tab displays the content of an XML file that represents the value in the selected Input or Output variable. The **XML** tab is enabled for XSD External Business Entities when the **External TypeDataModel View** button is toggled on.

Comment Tab

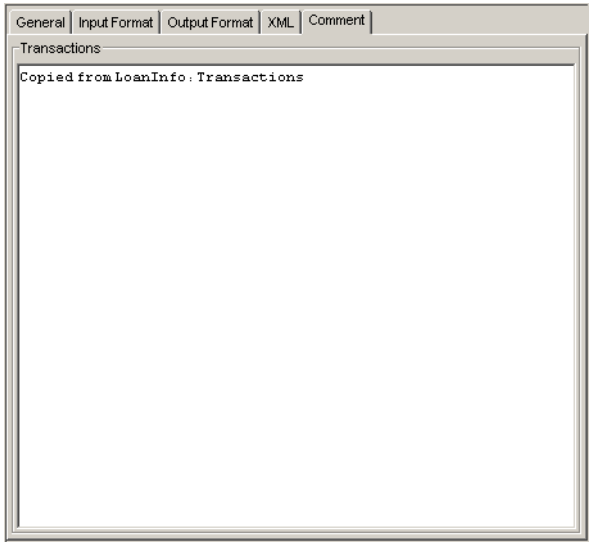


Figure 196. The TypeDataModel Viewer Comment Tab

Displays the comments that were written when the selected data type was created. When the selected data type has been copied from another data type, this tab automatically indicates the name of the source data type.

Comments can be viewed in the **TypeDataModel Viewer** but are modified in the **Business Entity Editor**.

Step 6: Watching Values Update

You can watch values as they are updated by the method execution, by adding variables to the **Debugger Watch** tab.

To watch values update:

1. Select a variable in either of the following ways:
   - From the **Variables** tab, right-click the required variable and select **Add to Watch**
   - From the **TypeDataModel Viewer**, right-click the required variable in the tree and select **Add to Watch**
2. Select the **Watch** tab.

   Your selected variable is listed in the **Watch** tab with the root node of the variable on the left, and the variable's field names, if the variable is a complex structure, in the columns to the right of the root node.
3. Click **Start**. The method is invoked, and values of the variables displayed in the **Watch** tab are updated according to the method execution.

See "Watch Tab" on page 370 for more details about the **Watch** tab.

Chapter 10. Services

webMethods JI Services represent a grouping of one or more methods. Clients invoke these service methods in order to execute the method steps contained within the methods.

Services serve as the interface between the methods and the webMethods JI environment.

This chapter covers the following topics:

- "The Services Tab" on page 391
- "Generating the Service and Test Client" on page 398
- "Test Client Code" on page 417
- "Testing the Service" on page 418
- "Deploying the Service" on page 419

The Services Tab

The **Services** tab of the Tree view displays services and their associated methods (Figure 197).

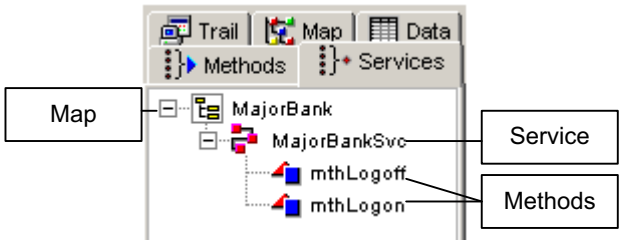


Figure 197. The Services Tab

The **Services** tab of the Tree view displays the following components:

Component	Description
Service	The name of the service.
Methods	The methods that have been added to the service.

Adding Services

To add a service to a map, select the top-level (map) component in the **Services** tab of the Tree view. With the map component selected, right-click and select **Add Service** from the shortcut menu. After the service has been added, change its name by right-clicking on the method name, selecting **Rename** from the shortcut menu, and entering the new name.

Configuring Service Properties

To configure properties for a service, select the service and edit the Properties view (Figure 198).

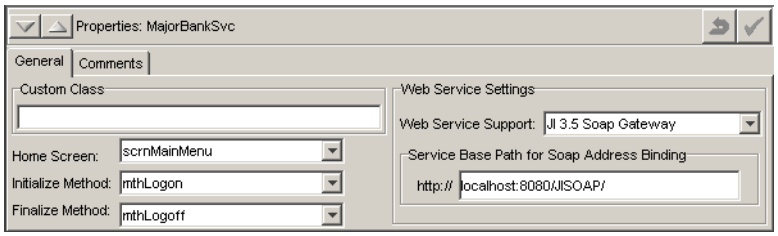


Figure 198. Service Properties View > General Tab

The Service Properties view provides the following options:

Option	Description
Custom Class	The custom class that should be used with this service. **Note:** If a custom class is created for the service, the name of the custom service class, rather than the name of the service as defined in MapMaker, should be used in the Configuration Manager when identifying the class name of the service during service configuration. Also, test clients generated by MapMaker will use the name of the custom service class when requesting the service. Alias names for any service can be created in the Configuration Manager.

Option	Description
Home Screen	When a client disconnects from this service, the service returns to this screen and waits for another client connection. For example, a legacy application might have a login screen where the user enters a username and password. Rather than having each client traverse to the login screen, the login screen can be set as the “home screen”. This enables the client to connect and automatically enter the username and password without having to execute initializing method steps to traverse to the login screen.
Initialize Method	This option identifies the method that should be run as soon as the service connects to the legacy host. For example, a legacy application might have a login screen where the user enters a username and password. An initialize method could be created that traverses to the login screen. When the service is first created, either by the Environment Manager or by a client request, the service executes the method and is at the login screen before the client invokes any methods.
Finalize Method	This option identifies the method that should be run when the service disconnects from the legacy host (such as logging off from the host).

Option	Description

Web Service Support	Choose one of the following from the drop-down list:
	• **None**: This is the default value for new services. It clears and disables the following properties:
	  • **Service Base Path for SOAP Address Binding**
	  • **WSDL**: This check box is located in the **Definitions** tab of the **Generate Code** dialog box (See "Definitions Tab" on page 403). It is disabled only if **Web Service Support** is set to **None** for ***all*** services in the map.
	• **JI 3.5 SOAP Gateway**: Provides backwards compatibility with version 3.5 SOAP Gateways. This is the default value in two cases:
	  • When a map does not have this property, since it is assumed to be a pre-existing map.
	  • When this property's value is **None** and the map is a version 3.5 converted map.
	Note: In this case, WSDL is generated using the version 3.5 procedure instead of the current version procedure.
	• **RPC**: The service is deployed into the SOAP Gateway as an RPC-style web service. The input and output parameters of the service's methods must be of either an MLM or an ExternalString type. If this requirement is not met, WSDL is not generated for the service.
	• **Document**: The service is deployed into the SOAP Gateway as a Document-style service. The input and output parameters of the service's methods must be of an XSD type. If this requirement is not met, WSDL is not generated for the service.
	Note: Each **Web Service Support** setting is compatible with specific method argument types. If any of the service's methods contains arguments that do not match the **Web Service Support** setting, the service's node in the Tree view is highlighted in red and the status bar indicates why it is invalid.

Option	Description

Service Base Path for SOAP Address Binding	Enter the location (URL) of the SOAP Gateway. If the map has no value for this property, the URL is constructed as follows:
	• The host name and port portion of the URL defaults to the value specified in the Soap Gateway's **Host:Port** field, set in the **Servers** tab of the **Properties** dialog box (see "Servers Tab" on page 105).
	• The servlet name portion of the URL depends on the **Web Service Support** setting:
	  • If **Web Service Support** is set to **JI 3.5 Soap Gateway**, the servlet portion of the URL is `/JISOAP/`.
	  • If **Web Service Support** is set to **RPC** or **Document**, the servlet portion of the URL is `/JIWSVC/services/`.
	• If the **Web Service Support** is set to **None**, **Service Base Path for Soap Address Binding** is blank and disabled.

Service Validation Checks

webMethods JI verifies that all method argument types are compatible with the service's **Web Service Support** setting (see "Configuring Service Properties" on page 392). If the two are incompatible, the service is highlighted in red in the Tree view and the status bar indicates why the service is invalid.

Figure 199 shows an example invalid service. In this case, the service contains a method that has no MLM-type input or output variables, while the service's **Web Service Support** property is set to **JI 3.5 SOAP Gateway**. Accordingly, the service name is colored yellow, and the status bar shows the following message: "**MajorBankSvc: mthConvertLoanTypes: has no MLM InVars or OutVars**".

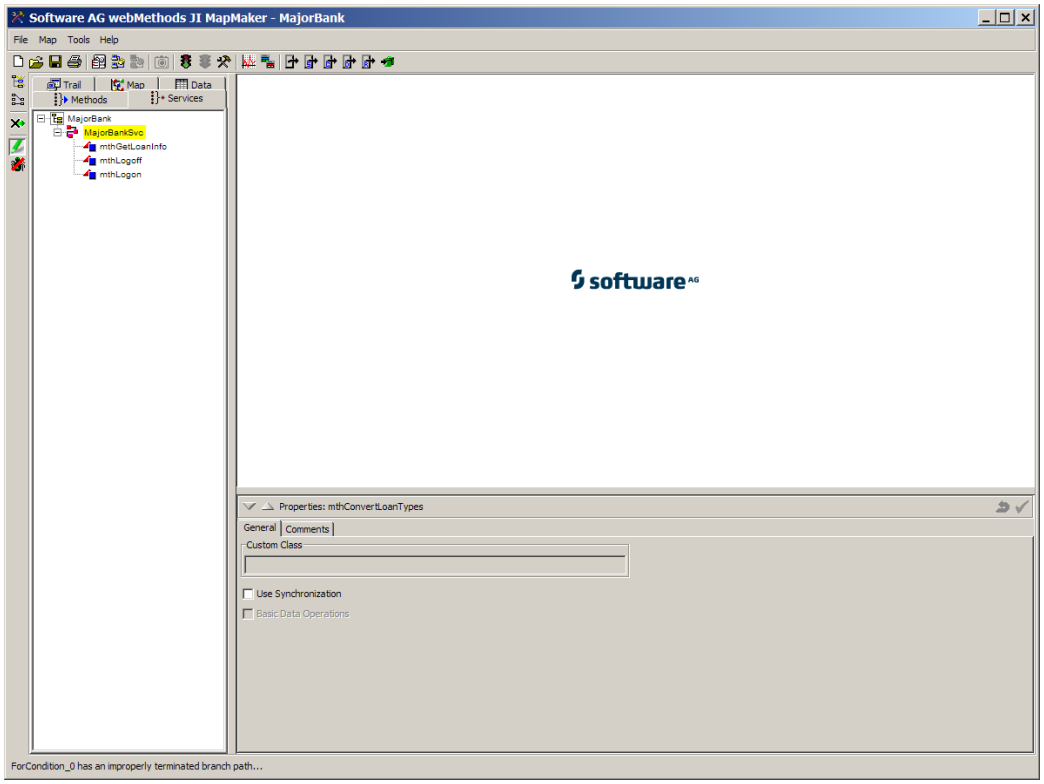


Figure 199. Example Invalid Service

Valid Method Input and Output Types for Each Web Service Support Setting

Web Service Support Setting	Valid Method Input and Output Types
None	All method argument types are valid in this case, since WSDL is not allowed to be generated.
JI 3.5 Soap Gateway	• For version 3.5 compatibility methods, all method input and output types are valid. • For current version methods, all input and output arguments must be of an MLM type.
RPC	MLM or ExternalString. **Note:** Version 3.5 compatibility methods are not allowed in RPC style services.

Web Service Support Setting	Valid Method Input and Output Types
Document	XSD **Note:** Version 3.5 compatibility methods are not allowed in Document style services.

Adding Methods to Services

To add a method, proceed as follows:

1 In the Tree view, right-click the Service node (e.g. **MajorBankSvc**, that was created in the *JI Tutorial*), and select **Add Method to Service** from the shortcut menu.

The **Choose a Method** dialog box opens (Figure 200).

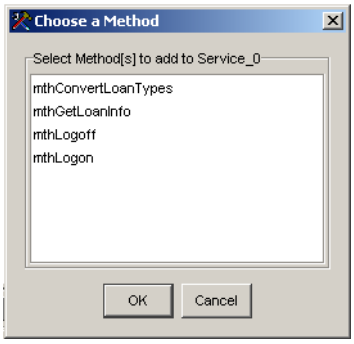


Figure 200. Choose a Method Dialog Box

2 Select all methods (select multiple methods by holding the **Ctrl** key down while selecting) and click **OK**.

Note: Methods that do not fit the service's **Web Service Support** setting (see "Configuring Service Properties" on page 392) are not displayed in this dialog box. For example, if **Web Service Support** is set to **RPC**, methods that contain an XSD input or output variable are not displayed in this dialog box.

After methods have been added to the service, the map is complete.

Note: Test clients automatically invoke all of the methods listed in the **Services** tab, except Initialize and Finalize methods. They are invoked in the order they were added to the service. To stop a test client from invoking a particular method, it must be edited.

Generating the Service and Test Client

After adding methods to the service, the next step is to generate code for the service. A test client can be generated at the same time.

To generate service and/or test client code, select **Tools** > **Generate** and select one of the options included in the sub-menu. The following options are offered:

- **Generate All Code**
- **Generate Service Code**
- **Generate Client Code**
- **Generate Definitions**
- **Generate Resource Adapter**

Note: Prior to generating code for the first time, the Java compiler must be set in the **Java** tab of the MapMaker **Properties** dialog box. Select **File** > **Properties** to verify the setting. See "Java Tab" on page 102 for details about the Java classpath property.
When regenerating code to an existing directory that already contains generated code, it may be necessary to remove outdated java and/or class files that may be associated with MapMaker objects that no longer exist because they have been deleted and/or renamed. If the map does not contain any custom code it is safe to remove all *.java and *.class files in the generated code directory.

Generating All Code

To generate both service and test client code, select **Tools** > **Generate** > **Generate All Code** to open the **Generate Code** dialog box.

Click the **Generate** button to generate the code, or click the **Restore Defaults** button to return the settings in the dialog box to their original values. Click the **Done** button to close the dialog box.

For information about the test client code, see "Test Client Code" on page 417.

There are five tabs in the **Generate Code** dialog box:

- Service tab
- Client tab
- Definitions tab
- Supplemental tab
- Resource Adapter tab

Service Tab

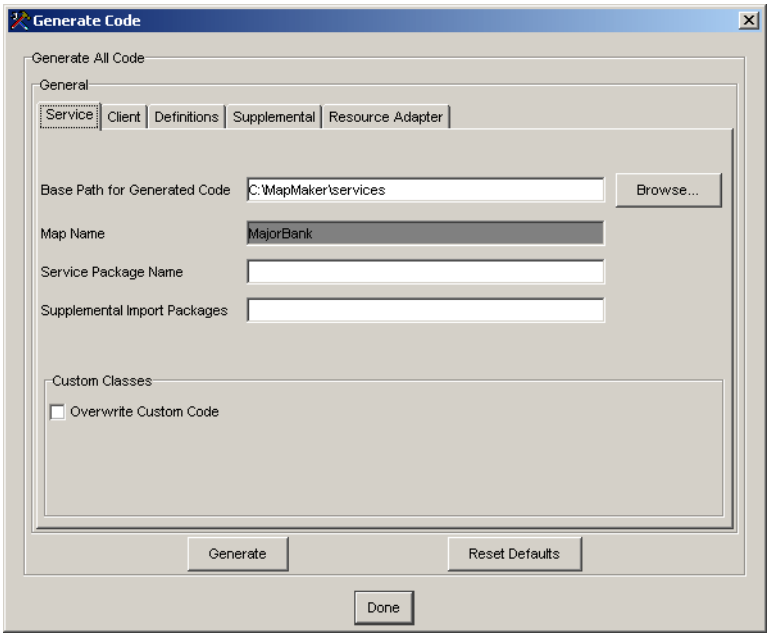


Figure 201. Generate Code Dialog Box > Service Tab

The **Service** tab provides the following options:

Option	Description
Base Path for Generated Code	The root directory (and drive letter, if applicable) where the generated code will be saved. When the code is generated, the actual location of the code is determined based on this directory plus the directory structure used in the **Service Package** option. So, for example, if the base path is */home/gen* and the package is set to "test.ea", the actual location is */home/gen/test/ea*. The default setting for this option is the output directory identified in the **Default Directories** tab of the **Properties** dialog box. See "Configuring MapMaker Properties" on page 96.
Map Name	The name of the map file. This file is completed automatically based on the name of the map, and cannot be edited.

Option	Description
Service Package Name	The name of the package to which the generated files belong. A Java package is a collection of Java class files that are grouped together to form a logical set of classes. The code generation places the Java files in directories according to package name. So do not compile two maps with the same package name, as this will cause the code generation to overwrite files.
Supplemental Import Packages	The packages that will be imported into the service. *com.jacada.mapstudio.gbbasic.** is imported by default. To import additional packages, include the packages here, using either a semicolon (;) or colon (:) to delineate each package.
Overwrite Custom Code	If set, any previously generated and modified custom code is overwritten with new stub code.

Client Tab

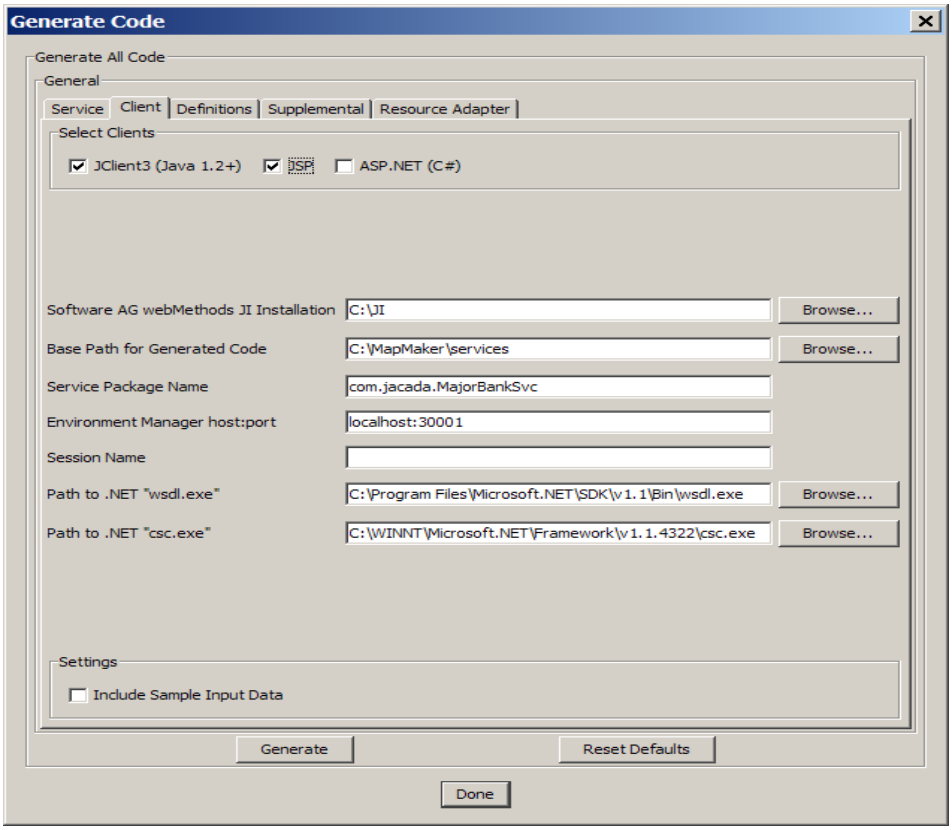


Figure 202. Generate Code Dialog Box > Client Tab

The **Client** tab provides the following options:

Option	Description
Select Clients	Here you can select one or more of the following clients to be generated: • JClient3 • JSP • ASP.NET (C#). See “ASP.NET Client Generation” on page 416.
webMethods JI Installation	The directory (and drive letter, if appropriate) where the webMethods JI software was installed.

Option	Description
Base Path for Generated Code	The root directory (and drive letter, if applicable) where the generated code will be saved. When the code is generated, the actual location of the code is determined based on this directory plus the directory structure used in the **Service Package** option. So, for example, if the base path is */home/gen* and the package is set to test.ea, the actual location is */home/gen/test/ea*. The default setting for this option is the output directory identified in the **Default Directories** tab of the **Properties** dialog box. See "Configuring MapMaker Properties" on page 96.
Service Package Name	The name of the package to which the generated files belong. A Java package is a collection of Java class files that are grouped together to form a logical set of classes. The code generation places the Java files in directories according to package name. So do not compile two maps with the same package name, as this will cause the code generation to overwrite files.
Environment Manager host:port	The DNS host name or IP address, and port number of the Environment Manager that the generated test client code should contact for access to the service.
Session Name	The session name that the client has given to this JService to enable a re-connection to the service if it still exists.
Path to .NET "wsdl.exe"	The location of the .NET framework SDK WSDL.EXE program.
Path to .NET "csc.exe"	The location of the .NET framework SDK CSC.EXE program.

Option	Description
Include Sample Input Data	If set, includes a sample of the input data.

Definitions Tab

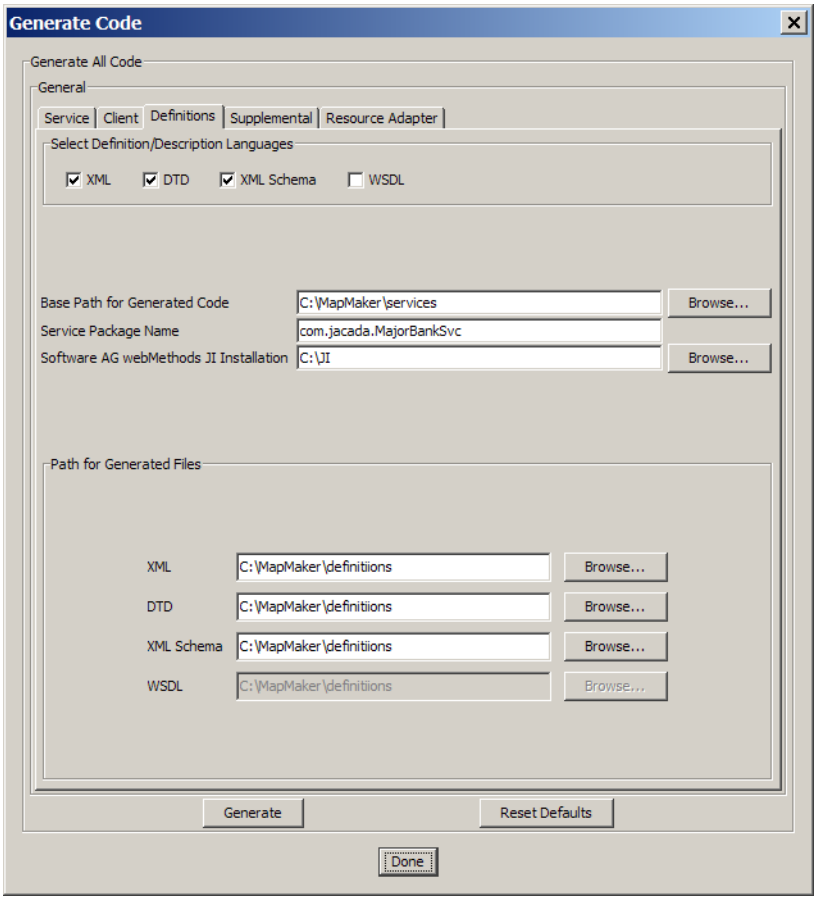


Figure 203. Generate Code Dialog Box > Definitions Tab

The **Definitions** tab provides the following options:

Option	Description
Select Definition/ Description Languages	Here you can select one or more of the following definition/description file types to be generated: • **XML** • **DTD** • **XML Schema** • **WSDL** **Note:** The **WSDL** box is disabled and cleared if **Web Service Support** is set to **None** in the Service Properties view (see "Configuring Service Properties" on page 392) of ALL services in the map.
Base Path for Generated Code	The root directory (and drive letter, if applicable) where the generated code will be saved. When the code is generated, the actual location of the code is determined based on this directory, plus the directory structure used in the **Service Package Name** option. For example, if the base path is */home/gen* and the service package name is set to test.ea, the actual location is */home/gen/test/ea*. The default setting for this option is the output directory identified in the **Default Directories** tab of the **Properties** dialog box. See "Configuring MapMaker Properties" on page 96.
Service Package Name	The name of the package to which the generated files belong. A Java package is a collection of Java class files that are grouped together to form a logical set of classes. The code generation places the Java files in directories according to package name. So do not compile two maps with the same package name, as this will cause the code generation to overwrite files.
webMethods JI Installation	The directory (and drive letter, if appropriate) the webMethods JI software was installed in.

Option	Description
Path for Generated Files	In these fields, specify the root directory (and drive letter, if applicable) where the following generated files will be saved: • XML • DTD • XML Schema • WSDL The default setting for these is set in the **Base Path for Generated Definitions** field identified in the **Default Directories** tab of the **Properties** dialog box. See "Configuring MapMaker Properties" on page 96.

Supplemental Tab

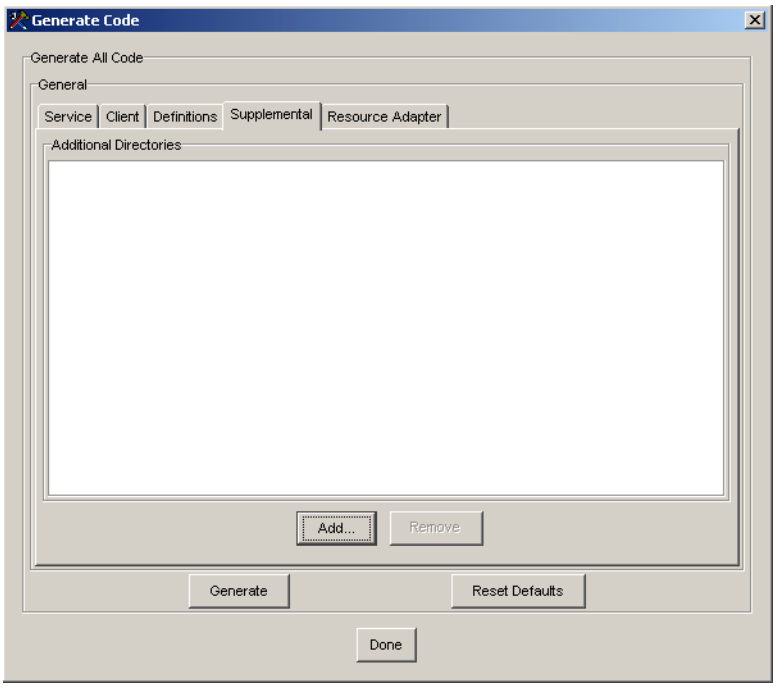


Figure 204. The Generate Code Dialog Box - Supplemental Tab

Under the **Supplemental** tab, you can list additional directories to be included in the JAR file.

- Click **Add** and browse to the directory desired, or
- Select an existing directory and click **Remove** to remove it.

Resource Adapter Tab

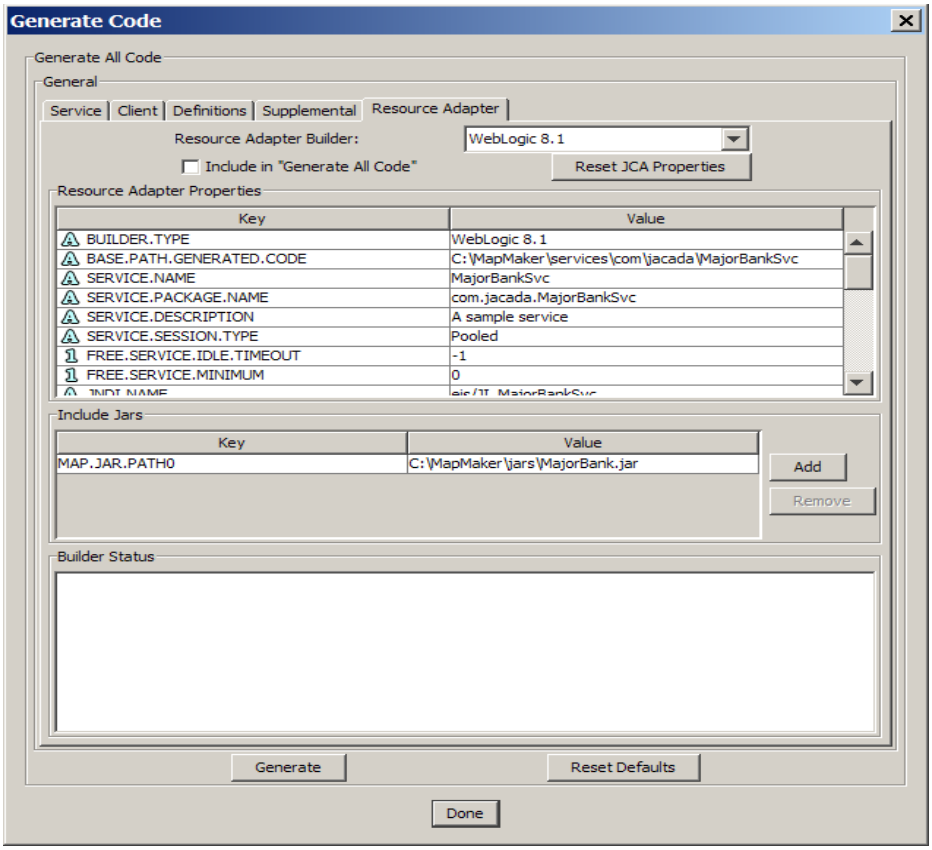


Figure 205. The Generate Code Dialog Box - Resource Adapter Tab

Use the **Resource Adapter** tab if you intend to run the webMethods JI service under an application server that supports the Java EE Connector Architecture (JCA).

See also the *webMethods JI Integration Guide*, Chapter 8, "Deploying a webMethods JI Service as a Resource Adapter" for important information about deploying a webMethods JI service under a JCA application server.

> **Note:** The **UserInteraction** method step and the **Interactive Mode onFail** feature are not supported in a Resource Adapter deployment.
> The **JClient3** step is supported; however, it requires a webMethods JI Server environment to communicate with at runtime.

To run the webMethods JI service under a JCA application server, a *Resource Adapter* must be created. The Resource Adapter resides on the application server and invokes the webMethods JI service. The webMethods JI service itself is actually tightly bound to the Resource Adapter, so from one point of view you could say that what is created is a package consisting of the Resource Adapter and the webMethods JI service.

Specifying the Application Server Type

The **Resource Adapter Builder** drop down list at the top of the **Resource Adapter** tab contains options where you specify the application server for which you are creating the service. See Figure 206.

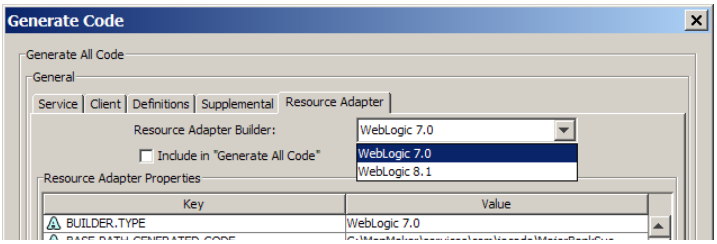


Figure 206. Selecting the Application Server for Which the Resource Adapter is to be Created

Application Server Selection Options

Option	Description
WebLogic 7.0	The webMethods JI Service is to run under WebLogic 7.0. **Note:** For WebLogic 7.0, the webMethods JI Resource Adapter is only supported under Service Patch 3, generally called Weblogic 7.0 SP3.
WebLogic 8.1	The webMethods JI Service is to run under WebLogic 8.1.

Include in "Generate All Code"

Set this checkbox if you wish the Resource Adapter to be generated when you press the **Generate** button in the **Generate Code** dialog box (**Tools** > **Generate** > **Generate All Code**).
If you do not want to create a Resource Adapter, leave the box cleared and ignore the remaining fields on the **Resource Adapter** tab.

Setting the Resource Adapter Properties

If you selected WebLogic 8.1 or WebLogic 7.0 as the application server type, the **Resource Adapter Properties** box now contains property names on the left side, and place for the property values to be entered on the right side. See Figure 207. Some property values have already been populated automatically by MapMaker.

Resource Adapter Properties

Key	Value
BUILDER.TYPE	WebLogic 7.0
BASE.PATH.GENERATED.CODE	C:\MapMaker\services\com\jacada\MajorBankSvc
SERVICE.NAME	
SERVICE.PACKAGE.NAME	com.jacada.MajorBankSvc
SERVICE.DESCRIPTION	
SERVICE.SESSION.TYPE	Pooled
FREE.SERVICE.IDLE.TIMEOUT	-1
FREE.SERVICE.MINIMUM	0
JNDI.NAME	eis/JI

Figure 207. Resource Adapter Properties

Resource Adapter properties and their values for WebLogic 8.1 and WebLogic 7.0, listed in alphabetical order.

Property Keyword	Property Value
BASE.PATH. GENERATED.CODE	Directory where MapMaker will create a directory called *<ServiceName_RAR>* that will contain *jijca.rar*, the *lib* directory, and at least one map JAR. This field is populated automatically by MapMaker to point to *<Base Path for Generated Code>* *<ServicePackageName>*. The two folder names that make up the default path are specified in the **Client** tab (**Tools > Generate > Generate All Code**).
BUILDER.TYPE	The application server that the Resource Adapter is to run under. Populated automatically by MapMaker.
HOST.ADDRESS	Enter the IP address of the host connection or a defined DNS name that represents the IP address.
HOST.NAME	Enter a name of your choice to represent the host machine.

Property Keyword	Property Value
HOST.PORT	A 4-digit number representing the host port used for connections with the Resource Adapter. Default port is 23.
HOST.TN3270.NAME	The LUName used for the host connection. Can be left blank if no particular LU is being used.
JNDI.NAME	Enter a name for the Resource Adapter. For WebLogic, it must begin with the string "`eis/`". For example, `eis/myra`. This is the name that your application will use to refer to the Resource Adapter. Default: `eis/JI_<YourServiceName>`
LICENSE.KEY	Taken from the webMethods JI license ID defined locally on your machine.
LICENSE.NAME	Taken from the webMethods JI license ID defined locally on your machine.
MAPS.JARS	The JAR file containing the webMethods JI map definitions. This JAR file was created in the directory for output JAR files as defined in webMethods JI's properties. Its name is the name of your map with the .jar file extension; for example: `mysvc.jar`.
SERVICE. DESCRIPTION	Enter an optional description of the service
SERVICE.LOGGING. ENABLED	True = Service logging is enabled. False = Service logging is inactive. Default: false
SERVICE.LOGGING. FILENAME	Specify a full pathname for the Service Logging file. Example: *C:\temp\mylogfile.log* Default: *<install directory>\<ServiceName>.log*

Property Keyword	Property Value
SERVICE.LOGGING. INSTANCE.DIRECTORY	Specifies the directory where each individual Service Instance log file will be written. Example: *C:\temp* The actual file name for the Service Instance log files should not be specified here, and is fixed as <*ServiceName_n*>, where "n" is a numeric suffix that starts at 0 for the first service instance and is incremented by 1 for each new service instance .
SERVICE.LOGGING. INSTANCE.LEVEL	An integer from 0 to 9 that controls how detailed the Service Instance log files will be. Note that the log level codes used for this property are different from those used for Service Logging. 0 = no Service Instance logging 9 = most detailed logging, up to 100+KB per request.

Property Keyword	Property Value
SERVICE.LOGGING. LEVEL	An integer from 0 to 6 that controls how detailed the Service Logging log file will be. Note that the log level codes used for this property are different from those used for Service Instance logging. The output for a "lower" log level includes the output for all "higher" levels; for example, the output for log level 3 (warning messages) also includes level 4 output (error messages) and level 5 output (fatal error messages). 0 = Debug-level logging, output is very detailed. 1 = At present, there is no difference between the log output for log level 0 and log level 1. 2 = Informational messages, mostly dealing with connections, sessions, and connection management 3 = Warning messages. The resource adapter outputs few warning messages. 4 = Error messages. These deal with exceptions that occurred in processing. 5 = Fatal error messages. There are no fatal error messages at this time 6 = No logging
SERVICE.NAME	This field is a drop-down list that is populated automatically by MapMaker with the name of your service, as shown on the **Service** tab in the Tree view (*not* the **Service** tab in the **Generate Code** dialog box). If your webMethods JI map consists of more than one service, each of the services appear in this drop-down list. If you want each service to be separately invokable, you must generate a separate RAR for each, changing your selection in the SERVICE.NAME field for each RAR generation.

Property Keyword	Property Value
SERVICE.PACKAGE.NAME	This field is populated automatically by MapMaker according to the service package name entered in the **Service** tab of the **Generate** dialog box.
WEBLOGIC.POOLED.CAPACITY.INCREMENT	Specifies the maximum number of additional managed connections that the WebLogic Server attempts to obtain during resizing of the connection pool. Default: 1.
WEBLOGIC.POOLED.CONNECTION.INACTIVE.TIMEOUT.SECONDS	*This property applies to WebLogic 8.1 only.* Specifies the amount of time in seconds that a connection handle can remain idle before becoming eligible for termination. This property prevents leaks in cases where the application did not close a connection after completing usage. Idle connections are terminated only when the connection pool becomes full and a new connection request is about to fail because of this. Default: 0.
WEBLOGIC.POOLED.CONNECTION.MAXIDLE.TIME	*This property applies to WebLogic 7.0 only.* Specifies the amount of time in seconds that a connection handle can remain idle before becoming eligible for termination. This property prevents leaks in cases where the application did not close a connection after completing usage. Idle connections are terminated only when the connection pool becomes full and a new connection request is about to fail because of this. Default: 0.
WEBLOGIC.POOLED.CONNECTION.PROFILING.ENABLED	If this property is set to true, connections can be monitored via the WebLogic console, including leaked and idle connections. The monitoring of connections in this way can be useful in identifying components that fail to close connections. See the WebLogic Server documentation for information on how to monitor connections via the WebLogic console. Default: false.

Property Keyword	Property Value
WEBLOGIC.POOLED. INITIAL.CAPACITY	The initial number of managed connections which WebLogic attempts to obtain at startup. Default: 1. Software AG recommends against setting this property to zero.
WEBLOGIC.POOLED. MAXIMUM.CAPACITY	The maximum number of managed connections which WebLogic allows. Requests for new managed connections beyond this limit results in a Resource Allocation Exception being returned to the caller. Default: 10.
WEBLOGIC.POOLED. SHRINK.PERIOD. MINUTES	*This property applies to WebLogic 7.0 only.* Specifies the amount of time in minutes that the connection pool manager waits between attempts to reclaim unused managed connections. Default: 15.
WEBLOGIC.POOLED. SHRINK.FREQUENCY. SECONDS	*This property applies to WebLogic 8.1 only.* Specifies the amount of time in seconds that the connection pool manager waits between attempts to reclaim unused managed connections. Default: 900 seconds (15 minutes).
WEBLOGIC.POOLED. SHRINKING.ENABLED	Specifies whether or not unused managed connections in the connection pool should be reclaimed, to reduce system resource requirements. Default: true.

Generating a Resource Adapter

To create the RAR and JAR files that make up the webMethods JI Resource Adapter for your service, press the **Generate** button.

For instructions on how to deploy the Resource Adapter, see "Deploying the webMethods JI Service to your Application Server" on page 237 of the *webMethods JI Integration Guide.*

The Reset JCA Properties Button

There may be situations where you have copied a webMethods JI Map from another environment into your current environment and are having problems generating the Resource Adapter. In such a case, the problem could be an incompatibility between certain properties of the old environment and your current environment. Specifically, the Map may be remembering file paths from its old environment, paths that do not show up in the **Resource Adapter Properties** list in the **Resource Adapter** tab.

To reset all of these "remembered paths" and other JCA properties to the defaults for your current installation, click the **Reset JCA Properties** button.

Generating Service Code

To generate only service code, select **Tools** > **Generate** > **Generate Service Code** to open the **Service** tab of the **Generate Code** dialog box.

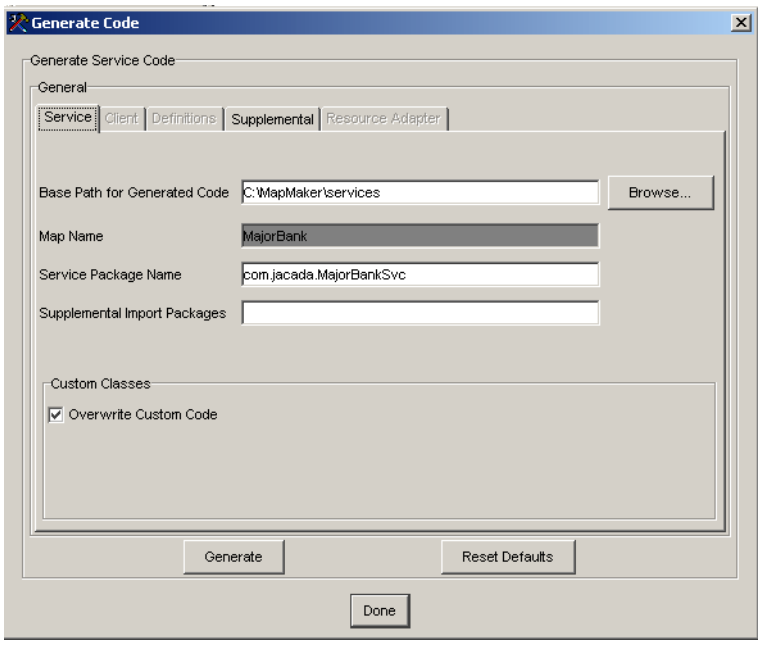


Figure 208. Generate Code Dialog Box > Service Tab

Only two of the tabs are available when selecting **Generate Service Code**:

- **Service** tab
- **Supplemental** tab

> **Note:** These tabs' properties are the same as those available when generating all code.

Generating Client Code

To generate only test client code, select **Tools** > **Generate** > **Generate Client Code** to open the **Generate Code** dialog box.

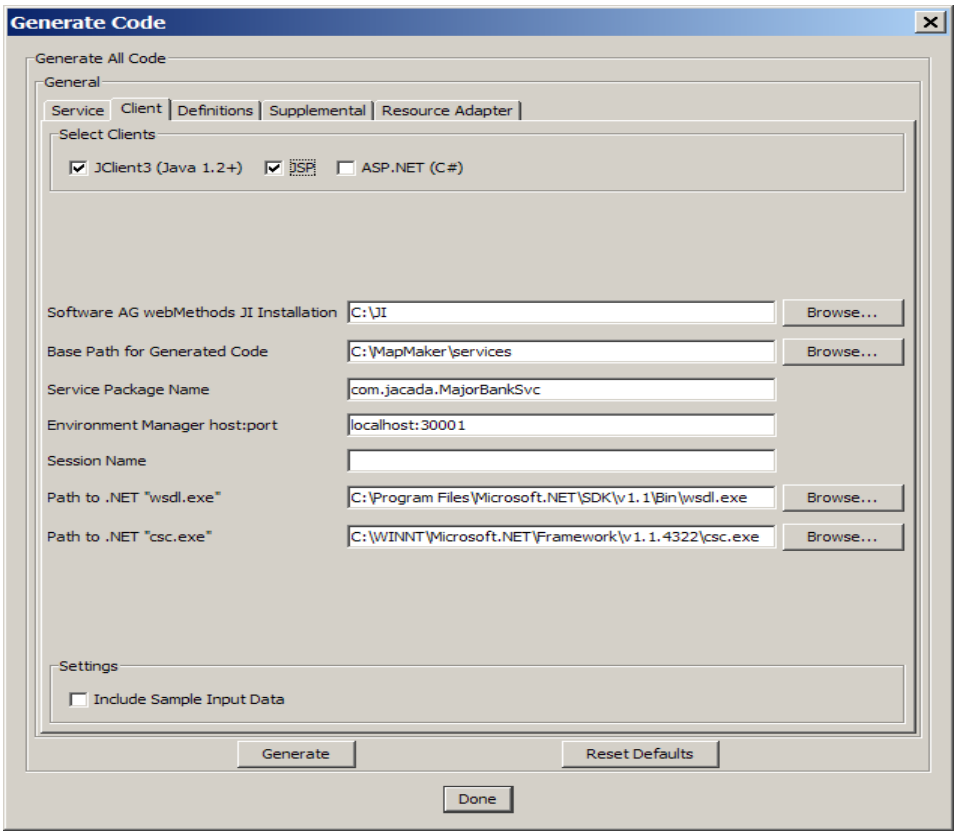


Figure 209. Generate Code Dialog Box > Client Tab

Only the Client tab is available when selecting **Generate Client Code**. See "Client Tab" on page 401 for more details.

Generating Definition/Description Files

To generate only the definition file, select **Tools** > **Generate** > **Generate Definitions**. The **Generate Code** dialog box opens, displaying the **Definitions** tab (all other tabs are disabled). For a description of the **Definitions** tab, see "Definitions Tab" on page 403.

Click the **Generate** button to generate the Definition files, or click the **Reset Defaults** button to return the settings in the dialog box to their original values. Click the **Done** button to close the dialog box.

Custom Code Conversion Utility

A custom code conversion utility is included with this version of webMethods JI. The utility is activated when the **Overwrite Custom Code** check box is set in the **Generate Service Code** dialog box and in the **Generate Code** dialog boxes's **Service** tab.

The utility converts custom code generated in previous versions of webMethods JI to the current version level. The conversion process is not totally comprehensive, due to incompatibility issues with regard to some methods and classes. If an incompatibility is detected during the compilation of the service code, the error is entered in a generated log file.

ASP.NET Client Generation

ASP.NET client generation requires the Microsoft .NET Framework Software Development Kit (SDK) version 1.1. In addition, an ASP.NET client requires the service it invokes to be deployed as a web service, and that your application server is running.

To generate an ASP.NET client:

1. Select the service in the MapMaker **Services** tab.
2. In the MapMaker Properties view, select either **RPC** or **Document** in the **Web Service Support** dropdown list, and specify the location of the WSDL file in the field below. For more information, see "Configuring Service Properties" on page 392.
3. Deploy the service, as usual. For more information, see "Deploying the Map and/or Custom EAServiceBean" on page 419.
4. Generate the WSDL code by completing the fields in the **Definitions** tab of the **Generate Code** dialog box, setting the **WSDL** checkbox. For more information, see "Definitions Tab" on page 403.
5. Deploy the service as a web service. For more information, see "Deploying the Service as a Web Service" on page 422.
6. Generate the client code by completing the fields in the **Client** tab of the **Generate Code** dialog box, setting the **ASP.NET (C#)** checkbox. For more information, see "Client Tab" on page 401.

ASP.NET clients are generated as .aspx files, and are located in the *clients* directory of the path specified in step 6 above. One .aspx file is generated for each method in the service, and one .dll file is generated per service. The .dll file is located in the *clients/bin* directory of the same path.

Run the ASP.NET clients by deploying them to a Web Server with .NET support, such as Microsoft's IIS. For more information about ASP.NET client implementation, see the *webMethods JI Client Developer Guide*.

Test Client Code

Note: This section does not apply to webMethods JI services generated as resource adapters. For testing and deployment of the webMethods JI service as a resource adapter, see Chapter 8 - "Deploying a webMethods JI Service as a Resource Adapter" on page 213 of the webMethods JI Integration Guide.

When the **Generate All Code** or **Generate Client Code** options are selected, test clients are created that can be used to test the generated service code and can serve as the basis for clients. The following test clients are created:

- *<service_name>_jclient3.java*: A Java client that must be compiled using the webMethods JI JClient3 library. Makefiles are generated to aid in compilation. For more information, see "Compiling the JClient3 Test Client" on page 417.

Note: The JClient3 library is recommended, because it has the most features.

- *<service_name>_<method_name>.jsp*: A JSP client that must be deployed to a servlet engine, such as Tomcat.
- *<service_name>_<method_name>.aspx*: An ASP.NET client is created for each method in the service. ASP.NET clients must be deployed to a Microsoft Web Server, such as IIS.

Compiling the JClient3 Test Client

Examples for compiling the JClient3 test client:

UNIX:

```
make -f <service_name>_Makefile.jclient3
```

Windows:

```
nmake /f <service_name>_nmakefile.jclient3
```

Compiling the JClient3 Test Client Without Using the Makefile

Examples for compiling the JClient3 test client:

UNIX:

```
<JDK_path>/bin/javac -classpath <JI_install_dir>/lib/jclient3.jar
<svcname>_jclient3.java
```

Windows:

```
<JDK_path>\bin\javac -classpath <JI_install_dir>\lib\ jclient3.jar
<svcname>_jclient3.java
```

Where `<JDK_path>` represents the path (and drive letter, if appropriate) to the installation of the Java Development Kit (JDK) and `<JI_install_dir>` represents the path (and drive letter, if appropriate) to the installation of webMethods JI, and `<svcname>` represents the name of the service as used in MapMaker.

Testing the Service

Note: This section does not apply to webMethods JI Services generated as Resource Adapters. For testing and deployment of the webMethods JI Service as a Resource Adapter, see Chapter 8 - "Deploying a webMethods JI Service as a Resource Adapter" on page 213 of the webMethods JI Integration Guide.

To test the service, the service must be configured in the webMethods JI environment. Configuration settings vary depending on whether the file will be deployed in the database or saved on the local file system. Typically, a service cannot be tested until it has been deployed, because it must have a Service Master. The automatically generated test clients execute the service using the service name as configured in the **Services** tab of the Tree view (or the name of the custom service class, if a custom class was included). Therefore, either this name should be used as the name of the Service Master in the Configuration Manager during service configuration, or the client code will have to be edited to correctly request the name of the service as it is configured in the Service Master record. For information about configuring the service, see *"Service Masters" on page 489*.

Note: Test clients invoke ALL methods defined under the **Service** tab, except for Initialize and Finalize methods.

If inputs are included in the service, it is necessary to edit the client code to include the inputs.

When the service is configured in the webMethods JI environment, execute the client at the command line by using one of the JClient3 test client method.

JClient3 Test Client

UNIX:

```
<JDK_path>/bin/java -classpath .:<JI_install_dir>/lib/jclient3.jar
<svcname>_jclient3
```

Windows:

```
<JDK_path>\bin\java -classpath .;<JI_install_dir>\lib\jclient3.jar
<svcname>_jclient3
```

Deploying the Service

Note: This section does not apply to webMethods JI services generated as Resource Adapters. For testing and deployment of the webMethods JI service as a Resource Adapter, see Chapter 8 - "Deploying a webMethods JI Service as a Resource Adapter" on page 213 of the webMethods JI Integration Guide.

This section describes:

- “Deploying the Map and/or Custom EAServiceBean” on page 419
- “Deploying the Service as a Web Service” on page 422

Deploying the Map and/or Custom EAServiceBean

webMethods JI service maps can be deployed to the Resource Database, so that the JAR file that contains the map and associated class files is saved in the database. There are a number of benefits to deploying maps to the Resource Database. Some of these benefits include: easier configuration, cross-platform compatibility, and simpler redundancy. While webMethods JI maps and service class files can also be stored on the local file system, it is recommended that the service be deployed into the database.

Custom EAServiceBeans can also be deployed as JAR files into the Resource Database.

To perform any of these operations, select **Tools** > **Deploy Map**.

This opens the **Deploy to Resource Server** dialog box (Figure 210).

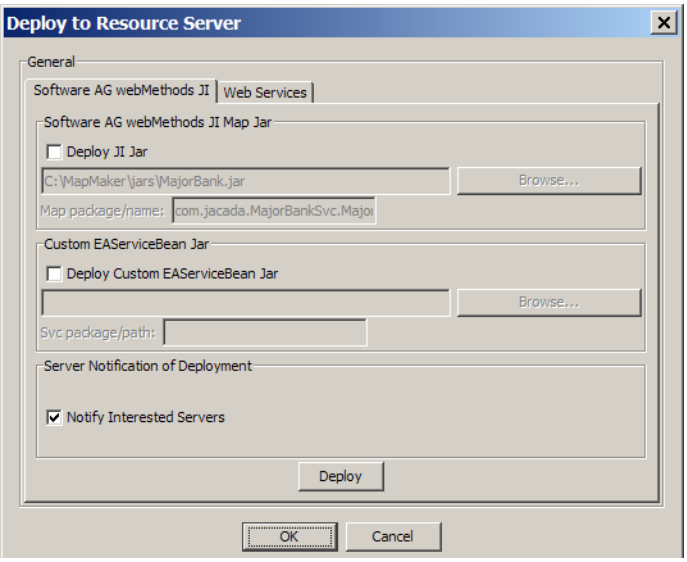


Figure 210. The Deploy to Resource Server Dialog Box > webMethods JI Tab

The **Deploy to Resource Server** dialog box includes two tabs:

- The webMethods JI tab, which allows you to deploy the map and custom Service Bean to the webMethods JI Resource Server.
- The **Web Services** tab, which allows you to deploy the webMethods JI service into the SOAP Gateway.

The webMethods JI Tab

The webMethods JI tab provides the following options:

Option	Description
Deploy webMethods JI Jar	To deploy the generated service code, set the **Deploy webMethods JI Jar** check box. By default, the JAR for the current map is selected. To select another JAR, click the **Browse** button and locate the JAR file.
Map package/name	The name of the map, including the package name. This field's default is based on information included in either the **Generate Code** or **Generate Service Code** dialog box.

Option	Description
Deploy Custom EAServiceBean Jar	If the EAServiceBean class for this service has been customized, set this check box to deploy the custom class to the database. The custom Service Bean and all accompanying files must be included in a JAR file. Click the **Browse** button and locate the JAR file.
Svc package/path	The path to the extended EAServiceBean, including the package name.
Notify Interested Servers	This causes Environment Managers to reload services using the Maps deployed here.

After completing the information in the **Deploy to Resource Server** dialog box, click the **Deploy** button.

Selecting the Resource Server for Deployment

If the **Enable Multicast Resource Location** check box is set in the **Servers** tab of the **Properties** dialog box, (see "Configuring MapMaker Properties" on page 96), the **Choose a Resource Server** dialog box opens presenting all Resource Servers that have been located using multicasting. Select the Resource Server that serves the database into which this service should be deployed and click the **OK** button.

Selecting the Database for Deployment

Next, the **Choose a Database** dialog box opens (Figure 211), listing all the Resource Databases being served by the Resource Server. If the **Enable Multicast Resource Location** check box is not set, the host and port of the Resource Server must be identified in the **Servers** tab of the **Properties** dialog box (see "Configuring MapMaker Properties" on page 96). If this is the case, the **Choose a Database** dialog opens automatically, bypassing the **Choose a Resource Server** dialog box.

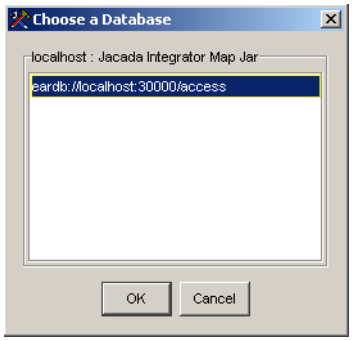


Figure 211. The Choose a Database Dialog Box

Select the appropriate Resource Database and click the **OK** button.

If a previous version of the service has already been deployed, a prompt appears to determine whether the existing version of the service in the database should be overwritten, or if a new version should be deployed and the version number incremented. Select the appropriate option from the dialog box and click **OK**.

Note: If a new service or new version of an existing service is deployed, the new version will not be used by any services that are currently running in the environment. One of the following should be done prior to using the new version:

— Set the **Notify Interested Servers** checkbox
— Restart the Environment Manager
— Kill any JClusters with instances of the service that are running, or
— Make a change to the Resource Database in the Configuration Manager and then select **File** > **Save and Apply** to force the Resource Database to be reapplied.

If a Service Master record does not exist for the service being deployed by the Map, a default Service Master is created.

Deploying the Service as a Web Service

This section describes the procedure for deploying webMethods JI services as web services. The procedure involves deploying services to the SOAP Gateway, so that the services are available to SOAP clients as web services. This is done in the **Deploy to Resource Server** dialog box's **Web Services** tab.

In order to deploy a web service, the following conditions must be met:

- The WSDL must be generated for the service. See page 398.
- The servlet engine must be started.
- Users who are deploying services as web services, must be defined as Administrators in the servlet engine installation.

Defining Users as Administrators in Tomcat

In the Tomcat servlet engine, users are defined as Administrators by editing the *<TOMCAT_HOME>\conf\tomcat-users.xml* file. This file contains parameters for which the values must be edited as follows:

Parameter	Value
role rolename	admin
user username	admin_username
password	admin_password
roles	admin

The values of the username and password parameters can be defined, as required.

If any of the listed parameters are missing from the *tomcat-users.xml* file, they must be added.

The Web Services Tab

The **Web Services** tab enables you to deploy services to the SOAP Gateway, so that the services are available to SOAP clients as web services.

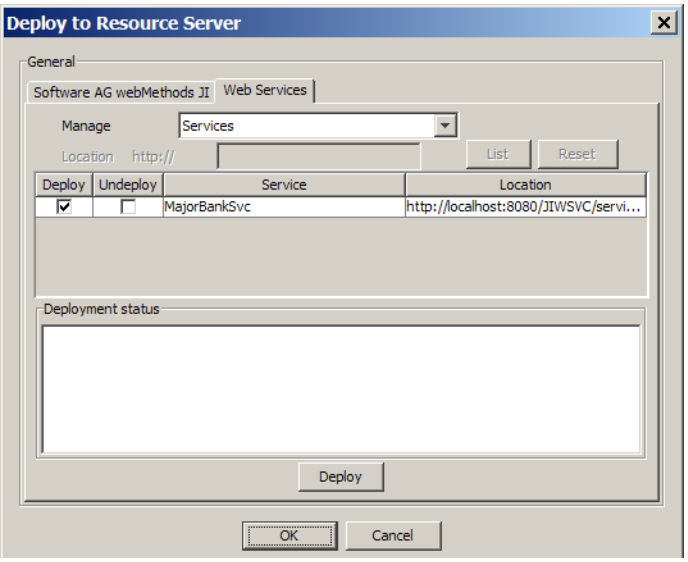


Figure 212. Deploy to Resource Server Dialog Box > Web Services Tab

To deploy a service to the SOAP Gateway:

1 Complete the fields in the **Web Services** tab. The **Web Services** tab provides the following options:

Option	Description
Manage	Select the type of deployment from the drop down list: • **Services**: Lists all web services in the current map. • **Deployed Web Services**: Allows you to manage (i.e. display, deploy and undeploy) web services in any SOAP Gateway, running either locally or on a remote system.
Location	This field is enabled only if **Deployed Web Services** is selected in the **Manage** drop down list, and allows you to specify the host and port of the system where the remote SOAP Gateway is running. • **List** queries the SOAP Gateway running on the specified system for all deployed services, and lists them in the table. • **Reset** restores the original host and port entered.

Option	Description
Services table	Each service in the map is represented by a line in this table, consisting of the following columns: • **Deploy** - A checkbox allowing you to set/clear the services to be deployed as web services to the SOAP Gateway. By default, all services' checkboxes are set. • **Undeploy** - A checkbox allowing you to set/clear all services that have already been deployed to the SOAP Gateway and are to be undeployed from it. • **Service** - the name of the service to be deployed to the SOAP Gateway. • **Location** - The location of the web service, specified in the **Service Base Path for Soap Address Binding** property in the service's Properties view (see "Configuring Service Properties" on page 392).
Deployment Status	Displays a text message indicating the results of the deployment: • If the deployment succeeds, this text area displays detailed information used during the deployment. • If the deployment fails, this text area displays an error message.
Deploy/Undeploy	Performs the following: • Deploys each service whose **Deploy** checkbox is set into the SOAP Gateway. • Undeploys each service whose **Undeploy** checkbox is set. After the deployment/undeployment completes, the **Deployment Status** text area shows the operation's results. The **Deploy/Undeploy** button is enabled only if at least one of the **Deploy**/**Undeploy** check boxes in the table is set.

2 Click **Deploy**.

A dialog box prompting you for your user ID and password is displayed.

3 Enter the username and password values defined in your servlet engine.

Chapter 11. Customizing MapMaker-Generated Service Code

Introduction to Custom Classes

Although MapMaker was designed to handle most situations encountered when interacting with legacy host applications, there are occasions when it is necessary for the developer to "extend" some components in the webMethods JI service code. As a result, webMethods JI includes two Java packages for customizing the service code.

These packages are:

- com.jacada.ea.ejb.server
- com.jacada.mapstudio.gbbasic

This chapter describes some situations encountered that require custom coding. It will describe the areas in MapMaker that allow custom classes to be added, and a description of important architectural details regarding the functions of webMethods JI services. At the end of the chapter are a number of code examples that illustrate the preferred way of accessing Global Variable data.

JavaDoc

A JavaDoc for webMethods JI Custom Classes is included with your webMethods JI installation. The JavaDoc, which can be opened in any Web browser, is located at *<JI_install_dir>/html/javadoc.html*. Classes available for customizing webMethods JI services are included in the GBBasic package, com.jacada.mapstudio.gbbasic. Another class, EAServiceBean, is used to extend the top level interface between the JService and the underlying service code, is in the package com.jacada.ea.ejb.server.

The GBBasic and EAServiceBean reference is also available in the *webMethods JI API Function Reference* book.

Using Threads with Custom Code

For performance reasons, access to GBBasic objects is not synchronized. If custom code is developed that accesses any GBBasic objects and uses threading, any required synchronization must be handled appropriately by the developer.

Custom Classes -- Examples of Common Uses

This section provides a brief description of three scenarios in which custom classes may be necessary.

Example One: Pre and/or Post Processing

Custom coding can also be used for pre and/or post processing in webMethods JI services. Pre-processing refers to computation that takes place prior to executing methods that connect to the legacy host. Some examples of pre-processing are:

- **Bounds checking:** The client might pass an account number into the service as a method input. If you know the number of characters required in the account number, you could use custom coding to verify that the account number is the appropriate length prior to executing the method that inputs the account number into the legacy application.
- **Verifying and filtering input in other ways:** For example, you might want to verify that an account number contains numeric characters only, or that input is capitalized.
- **Passing in a host name from the client:** It may be necessary to determine the host machine or LU connection name based on the client. A client can be designed to pass in a value that can be used to determine which host or LU that the client should use.

Post-processing allows the user to manipulate data after it has been extracted from the legacy application. Examples of post-processing include:

- **Verifying that output is in the proper format to be processed by the client application:** For example, a client may not be able to correctly handle data with a negative value. In this case, an error message could be returned to the client indicating that the value is a negative number.
- **Adding a message to the output:** For example, the host application might provide an error code when the application is in an error state. Using post-processing, the error code could be converted into a descriptive text message. This is done by reading the text message from a file that describes each error code. The text message can then be returned to the client.

A number of the code examples at the end of this chapter use pre or post processing. See, for example, “Client Host Selection” on page 436 and “Processing Extracted Data and Inputting the Processed Data” on page 437.

Example Two: Accessing External Data Sources

Often, clients require access to multiple data sources. They may need information from multiple databases in addition to data from legacy applications. Custom coding allows services to access data from data sources external to webMethods JI. This external data can be included in the service message that is returned to the client.

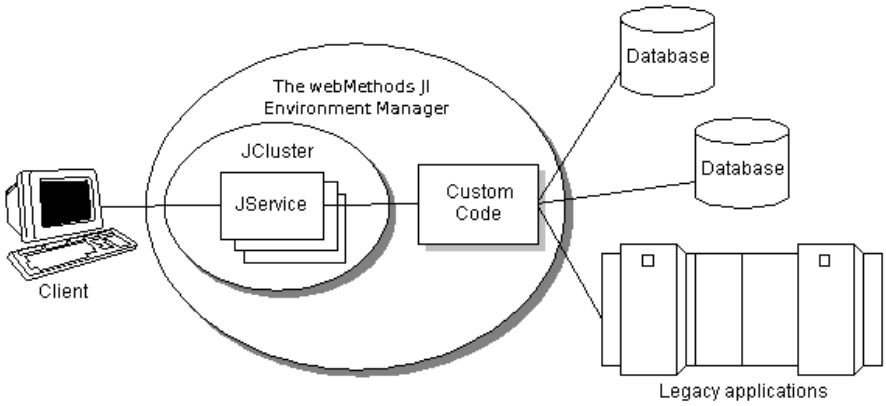


Figure 213. Accessing External Data Sources

For an example of the coding required to access external data sources, see "JDBC Access" on page 435.

Example Three: Branching

The webMethods JI service may need to be able to "branch" to different methods depending on input from the client. For example, clients might receive a different sequence of screens from the legacy application depending on the machine used to log onto the legacy host.

In this case, multiple methods should be developed that will handle each possible screen sequence. The client can then pass a check value to the service (for example, the system name of the client's machine). The service then executes a method that is written using custom code, which reads the client's check value and compares it to the value contained on the application screen.

After verifying the client's check data, the method determines which additional method to branch to, depending on the client's system name.

This is a description of the methods flow:

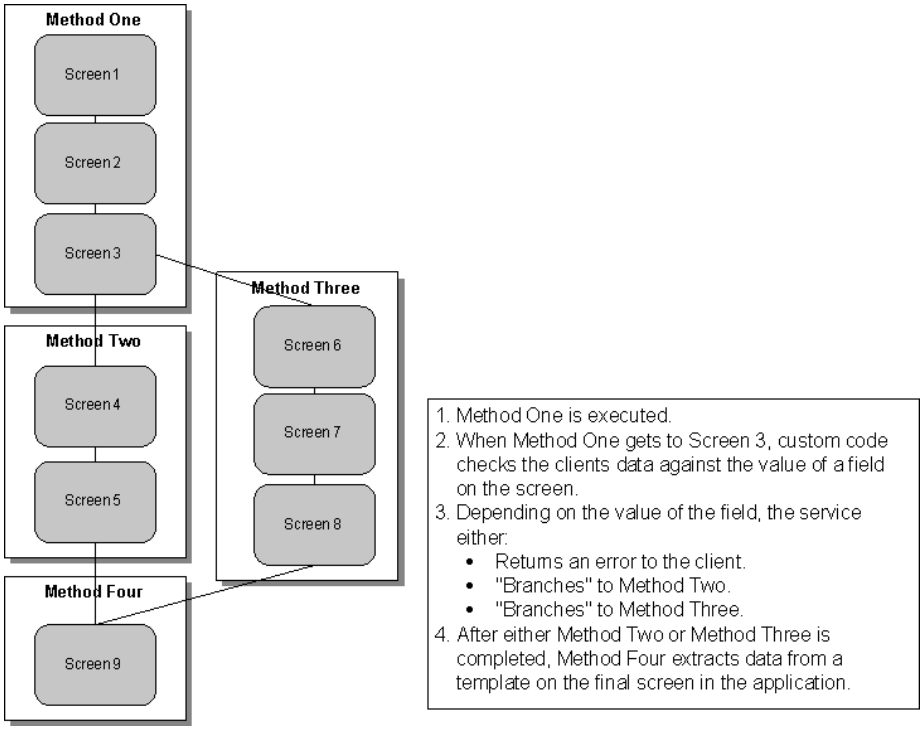


Figure 214. Branching Methods Flow

An example of custom code used to solve a problem similar to this is provided at the end of this chapter. See “Branching” on page 436.

Defining Custom Classes in MapMaker

For more information about defining Custom Classes within MapMaker, see *“Custom Classes” on page 112*.

Extending the EAServiceBean

In addition to the components in MapMaker, the EAServiceBean can also be extended using custom coding. The EAServiceBean is the top level service interface, providing the communication between JServices running in the Environment Manager’s server infrastructure and the service code. The EAServiceBean can be extended to create additional methods that are not included in the GBBasic library.

Deploying Extended EAServiceBeans

Deploying into the Resource Database

When custom Service Beans are created, they can be deployed into the webMethods JI Resource Database from within MapMaker at the same time that the map code is deployed. For further information, see *"Deploying the Map and/or Custom EAServiceBean" on page 419*. The custom service bean must be stored in a JAR file to be deployed to the Resource Database.

Using Extended EAServiceBeans from the Local File System

For information about configuring customized service beans on the local file system (rather than deploying them into the Resource Database), see *"Service Masters" on page 489*.

Example Code for Extended EAServiceBean

Example code for customizing the EAServiceBean is provided at the end of this chapter. See "Echo Test/Ping" on page 436.

Customizable Components in MapMaker

The following components in MapMaker allow for customization using Custom Classes. Examples of situations that might require custom coding are provided for some components. However, you should remember that custom coding within webMethods JI is extremely flexible. Many of these examples could be solved by extending other components. These are provided as generic examples only.

- **Maps:** The GBMap class is the central class that provides access to components in the maps (screens, fields, templates, methods, etc.). This class also provides the host connection for other GBBasic classes. One example for extending the GBMap class would be to select the host connection based on client input. For a code example of this, see "Client Host Selection" on page 436.

Note: If a custom class is created for a map, the name of the custom map class, rather than the name of the map as defined in MapMaker, should be used in the Configuration Manager when identifying the class name of the map during service configuration.

- **Global Actions:** Global Actions use the GBAction class to extend actions. Global Actions could be extended to check the current screen, to decide if the Global Action should be performed.
- **Screens:** The GBScreen class represents a screen that includes the actions, tags, templates and the formatted field layout and/or tags for recognition within the presentation space.
- **Actions:** One example for why you might want to extend the GBAction class would be to limit the number of times the client is allowed to attempt to log in to the host. A custom class can be created to allow the client to attempt to log in three times. If login fails, ten minutes must pass before another log in attempt is allowed. For an example of code that demonstrates this, see "Limiting Login Attempts" on page 436.
- **Data Templates:** The GBDataTemplate class represents a set of data fields. One example for extending GBDataTemplates would be to perform some field calculations to be used to modify the values of the fields that are passed back to the client. This example could also apply to customizing table templates or data fields.
- **Table Templates:** The GBDataTableTemplate class is a sub-class of the GBDataTemplate class and inherits those methods. This class represents sets of data fields contained in table templates.
- **Data Fields:** The GBDataField class represents the fields in data templates and table templates.
- **Methods:** The GBMethod class represents the method in the map. You might extend the GBMethod class to check values returned from the data templates and table templates by the GBMethodStepFetch.

Note: If a custom class is created for a method, the name of the custom method class, rather than the name of the method as defined in MapMaker, should be used by clients when invoking the method.

- **Traverse Method Steps:** The GBMethodStepTraverse class represents a Traverse method step.
- **Perform Method Steps:** The GBMethodStepPerform class represents a Perform method step.
- **Fetch Method Steps:** The GBMethodStepFetch class represents a Fetch method step. For a code example for extending the GBMethodStepFetch class, see "Processing Extracted Data and Inputting the Processed Data" on page 437.
- **Write Steps:** The GBMethodStepWrite class represents a Write method step.
- **User Interaction Steps:** The GBMethodStepUserInteraction class represents a User Interaction step.
- **Invoke Steps:** The GBMethodStepInvoke class represents an Invoke step.
- **JClient3 Steps:** The GBMethodStepJClient3 class represents a JClient3 step.
- **Thread Steps:** The GBMethodStepThread class represents a Thread step.

- **Thread Join Steps**: The GBMethodStepThreadJoin class represents a ThreadJoin step.
- **Set Steps**: The GBMethodStepSet class represents a Set step.
- **IfCondition Steps**: The BasicIfConditional class represents an IfConditional step.
- **IfElseCondition Steps**: The BasicIfElseConditional class represents an IfElseConditional step.
- **ForCondition Steps**: The BasicForConditional class represents a ForConditional step.
- **WhileCondition Steps**: The BasicWhileConditional class represents a WhileConditional step.
- **DoWhileCondition Steps**: The BasicDoWhileConditional class represents a DoWhileConditional step.
- **Services:** The GBService class represents a webMethods JI Service. This class can be used to check data returned from the method and modify the data or invoke another method based on the value of the data.

Note: If a custom class is created for a service, that name rather than the MapMaker name, should be used in the Configuration Manager to identify the service during service configuration.

Test clients generated by MapMaker use the name of the custom service class when requesting the service.

Template Data Architecture

The following diagram outlines the data structure used/referenced by Fetch, Write, and Set steps. The data structures are:

- Map - java.util.Map
- List - java.util.List

For a detailed description of the process that creates these data structures and returns them to the client, see the steps below:

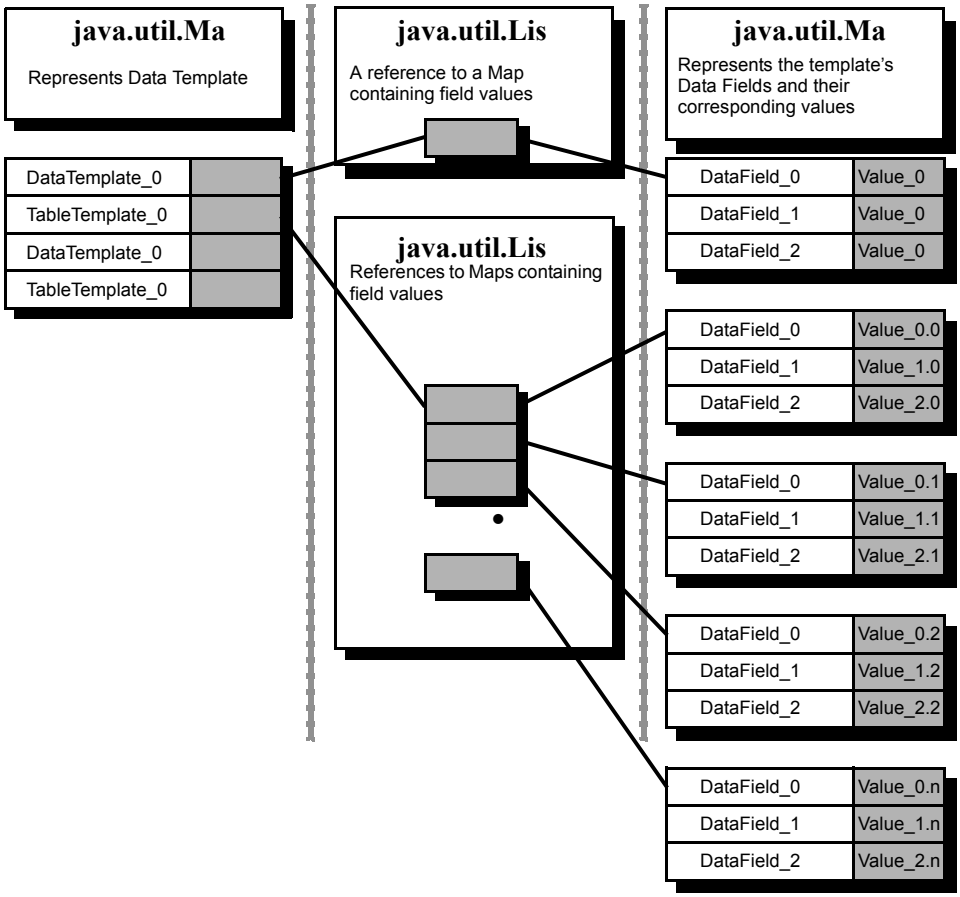


Figure 215. Template Data Architecture

The following steps provide an overview of Fetched Data Returned:

1 If the fetched data template is:

- A table template, the fetchAllRecords(fetchNames) method creates multiple java.util.Maps, with one hashtable for each row in the table on the legacy application. The keys in the hashtables represent the names of each data field in the table template, and the value for each key is the information displayed in that field on the legacy screen.

> **Note:** fetchAllRecords(fetchNames) also creates a java.util.List, and as each java.util.Map is created, a value is added to the java.util.List that points to the hashtable.

- A data template, the fetchRecord(fetchName) method creates one java.util.Map. The keys in this hashtable represent the names of each data field in the data template, and the value for each key is the information displayed in that field on the legacy screen.

2. The fetchRecord(fetchName) or fetchAllRecords(fetchNames) function returns its data to the GBMethodStepFetch.invoke function.
   - If the template was a table template, the java.util.List is returned to the GBMethodStepFetch.invoke function.
   - If the template was a data template, the java.util.Map is returned to the GBMethodStepFetch.invoke function, which creates a java.util.List. The single value in this list points to the java.util.Map.
3. The java.util.List created in Step 2 is returned to the GBMethod.invoke.
4. The GBMethod.invoke function creates a java.util.Map. It adds each template name as a key in this hashtable with the corresponding value pointing to the appropriate java.util.List.

   At this point (and at any point beyond this step), additional values can be added to this hashtable using custom coding. These values will be included in the service message that the JService passes to the client (as described in the following steps). These values can be any type that is supported by webMethods JI (String, Integer, Float, or Long).
5. The method.invoke's template hashtable is then passed through the GBService and the EAServiceBean to the JService.
6. The JService creates a service message containing the field names and values from the data field java.util.Map and passes this service message back to the client.

Example Code

Code examples for a number of situations that might require custom coding are included with the webMethods JI installation. Code examples are located in the *<JI_install_dir>/examples/mapmaker/com/Jacada/examples* webMethods JI installation directory. A brief summary of each of the code examples is provided here.

JDBC Access

This code example shows how to access data from a both legacy host using webMethods JI and a database using JDBC. This example uses the following Java code file(s), located in the *db* sub-directory:

```
CustomDBAction.java
CustomDBMethod.java
UserDB.java
UserDetails.java
```

Branching

This example demonstrates how to use custom code to govern manual "branching" among methods in the service. This example uses the following Java code file(s), located in the *branch* sub-directory:

`CustomBranchMethod.java`

Echo Test/Ping

This example demonstrates how to extend the EAServerBean (com.jacada.ea.ejb.server.EAServiceBean) to create additional (non-GBBasic) methods. This example uses the following Java code file(s), located in the *sbean* sub-directory:

`GenericDebugServiceBean.java`

`MTEAServiceBean.java`

Limiting Login Attempts

This example demonstrates how to customize an action to limit the number of login attempts that a client is allowed to make. After making three unsuccessful login attempts, the action waits for ten minutes before allowing additional login attempts. This example uses the following Java code file(s), located in the *gbbasic* sub-directory:

`CustomAction.java`

Custom Output Variables

This example demonstrates how to customize output variables that are used for input into fields on the legacy screen. This example uses the following Java code file(s), located in the *gbbasic* sub-directory:

`ccGetAllLines.java`

Client Host Selection

This example shows how to use information from the client to select the legacy host to which the service will connect. This example uses the following Java code file(s), located in the *gbbasic* sub-directory:

`Custom_HostConnectMethod.java`

`Custom_HostConnectService.java`

Processing Extracted Data and Inputting the Processed Data

This example demonstrates how to extract data from the legacy screen, process the data to compute a value, and input the value back into the legacy application. This example uses the following Java code file(s), located in the *gbbasic* sub-directory:

`CustomFetch.java`

Using Custom Code to Access Variable Data

Overview

This section contains examples of custom code showing the preferred way of accessing Global Variable data in webMethods JI.

> **Note:** Method Variable data can be accessed in a similar way, by calling methods on the GBMethod instead of the GBMap.

The examples include:

1. Getting and Setting the value of a Global Variable.
   a) Getting the value from a top level InternalString.
   b) Getting the value from a top level InternalNumber.
   c) Setting the value of a top level InternalString.
   d) Setting the value of a top level InternalNumber.
   e) Setting the value of an InternalString within an IBE.
   f) Setting the value of an InternalString array within an IBE.
   g) Setting the value of an InternalString sub-element in an IBE array.
2. Adding a structure to an “isArray” structure within an IBE.
3. Determining the webMethods JIwebMethods JI type of an element in a BE.
4. Determining if an element in a BE is an array or not.
5. Copying data from structure to structure.

Conventions Used in These Examples

- It is assumed that the sample code is part of an extended method step. This is to simplify the acquisition of the `GBMap` reference. In other extended classes, the method of obtaining the `GBMap` is a little different.
- `try/catch` clauses are omitted to simplify the example code.
- Assignments should be done through the `GBMap`'s `setValue` and `doAssignment` methods. The `GBMap` takes care of locking the `TypeDataModel` for writing if there are thread steps involved in the method. See the javadoc for `GBMap` for more detail about these methods.

These are the BE structures used by the Global Variables for these examples:

> **Note:** All sub-structures are private structures.

```
gvSourceStringVar (InternalString)

gvSourceNumberVar (InternalNumber)

gvSourceVar (SourceIBEType)
- SourceString (InternalString)
- SourceNumber (InternalNumber)
- SourceStringStructure0
    - SourceSubString0 (InternalString)
- SourceStringStructure1
    - SourceArrayString1 (InternalString) [isArray=true]
- SourceStringArrayStructure0 [isArray=true]
    - SourceSubString1 (InternalString)
- SourceArrayStructure0 [isArray=true]
    - SourceSubStructure0
        - SourceStringField (InternalString)
        - SourceNumberField (InternalNumber)

gvTargetStringVar (InternalString)

gvTargetNumberVar (InternalNumber)

gvTargetVar (TargetIBEType)
- TargetString (InternalString)
- TargetNumber (InternalNumber)
- TargetStringStructure0
    - TargetSubString0 (InternalString)
```

```
- TargetStringStructure1
    - TargetArrayString1 (InternalString) [isArray=true]
- TargetStringArrayStructure0 [isArray=true]
    - TargetSubString1 (InternalString)
- TargetArrayStructure0 [isArray=true]
    - TargetSubStructure0
        - TargetStringField (InternalString)
        - TargetNumberField (InternalNumber)
```

Code Example 1a - Get Value from Top Level InternalString

```
/* -------- START Get value from Simple TypeDataModel -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we
// need to get the data we want.
JITypeDefinition srcVar =
(JITypeDefinition)gbMap.getGlobalVar("gvSourceStringVar");

// Get the TypeDataModel of the Global Variable. The TypeDataModel
// contains the data for the Global Variable.
TypeDataModel srcTDM = srcVar.getTypeDataModel();

// Values are [generally] wrapped in a JITypeValueObject. This
// wrapper provides access to the JIType associated with the value.
// The cases where the value is NOT wrapped in a JITypeValueObject
// are when an array of values is returned. In that case, it
// is an array of JITypeValueObjects.
JITypeValueObject valObj =
(JITypeValueObject)gbMap.getValue(this, srcTDM, null);

// we can cast directly to String since the type of the
// JITypeDefinition is InternalString. If we looked up the
// value for "gvSourceNumberVar", we could have casted the
// contained value to java.lang.Number.
java.lang.String stringValue = (java.lang.String)valObj.getValue();

// There also exist convenience methods on GBMap to simplify
```

```
// getting values from the GlobalVariables, these will be used
// throughout the rest of these examples where applicable.
stringValue = (String)gbMap.getGlobalVarValue(this,
"gvSourceStringVar");

// To always get a String value, use the following method.
stringValue = gbMap.getGlobalVarValueAsString(this,
"gvSourceStringVar");
/* -------- END Get value from Simple TypeDataModel -------- */
```

Code Example 1b - Get Value from Top Level InternalNumber

```
/* -------- START Get value from Structured TypeDataModel -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceVar");

// Get the TypeDataModel of the Global Variable. The TypeDataModel
// contains the data for the Global Variable.
TypeDataModel srcTDM = srcVar.getTypeDataModel();

// Values are [generally] wrapped in a JITypeValueObject. This
// wrapper provides access to the JIType associated with the value.
// The cases where the value is NOT wrapped in a JITypeValueObject
// are when an array of values is returned. In that case, it is
// an array of JITypeValueObjects.
JITypeValueObject valObj =
    (JITypeValueObject)gbMap.getValue(this,
        srcTDM,
        "gvSourceVar/SourceString");
```

```
// we can cast directly to String since the type of the
// JITypeDefinition "gvSourceVar/SourceString" is InternalString.
// If we looked up the value for "gvSourceVar/SourceNumber", we
// could have casted the contained value to java.lang.Number.
java.lang.String stringValue = (java.lang.String)valObj.getValue();
/* -------- END Get value from Structured TypeDataModel -------- */
```

Code Example 1c - Set Value of Top Level InteralString

```
/* -------- START Set Value top-level InternalString -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Source Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceStringVar");

// Get the Target Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetStringVar");

if (true) {
    // Copy the value from the source GV to the target GV
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        null,
        gbMap.getGlobalVarValue(this, "gvSourceStringVar");

    /*
    * or...
    * This depends on the Type Mappings between the source JIType
    * and the target JIType. This is also a pseudo-atomic operation
    * in that any Thread locking of the source TypeDataModel and
    * the target TypeDataModel is done before any data is
    * read from the source. Using the above setValue call with the
```

```
    * getValue call doesn't lock both the source and the target
    * before any work is done.
    *
    gbMap.doAssignment(
        this,
        srcVar.getTypeDataModel(),
        null,
        tgtVar.getTypeDataModel(),
        null);
    *
    */
    /*
    * or... use the Global Variable convenience method.
    gbMap.doGlobalVarAssignment(this, "gvSourceStringVar",
"gvTargetStringVar");
    *
    */
}
else {
    // Set the value of the target GV with a value NOT from another
    // TypeDataModel... The target TypeDataModel is locked
    // (if necessary) before the value is set.
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        null,
        "Some Value...");
    /*
     * or...
    gbMap.setGlobalVarValue(this, "gvTargetStringVar", “Some
Value...”);
     *
     */
}
/* -------- END Set Value top-level InternalString -------- */
```

Code Example 1d - Set Value of Top Level InternalNumber

```
/* -------- START Set Value top-level InternalNumber -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Source Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceNumberVar");

// Get the Target Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetNumberVar");

if (true) {
    // Copy the value from the source GV to the target GV
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        null,
        gbMap.getGlobalVarValue(this, "gvSourceNumberVar"));

    /*
    * or...
    * This depends on the Type Mappings between the source JIType
    * and the target JIType. This is also a pseudo-atomic operation
    * in that any Thread locking of the source TypeDataModel and
    * the target TypeDataModel is done before any data is
    * read from the source. Using the above setValue call with the
    * getValue call doesn't lock both the source and the target
    * before any work is done.
    *
    gbMap.doAssignment(
        this,
        srcVar.getTypeDataModel(),
        null,
        tgtVar.getTypeDataModel(),
```

```
        null);
    *
    */
    /*
    * or... use the Global Variable convenience method.
    gbMap.doGlobalVarAssignment(this, "gvSourceStringVar",
"gvTargetStringVar");
    *
    */
}
else {
    // Set the value of the target GV with a value NOT from another
    // TypeDataModel... The target TypeDataModel is locked
    // (if necessary) before the value is set.
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        null,
        new Integer(42)); // Values must be Objects, so
                          // wrap int values in Integers
    /*
    * or...
    gbMap.setGlobalVarValue(this, "gvTargetStringVar", new Integer
(42));
    *
    */
}
/* -------- END Set Value top-level InternalString -------- */
```

Code Example 1e - Set Value of InternalString Within an IBE

```
/* -------- START Set Value InternalString within IBE -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Source Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceVar");
```

```
// Get the Target Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetVar");

if (true) {
    // Copy the value from the source GV to the target GV
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetString",
        gbMap.getValue(this,
            srcVar.getTypeDataModel(),
            "gvSourceVar/SourceString"));

    /*
    * or... This may be easier if you already have a reference
    * to the srcTD and tgtTD...
    * This depends on the Type Mappings between the source JIType
    * and the target JIType. This is also a pseudo-atomic operation
    * in that any Thread locking of the source TypeDataModel and
    * the target TypeDataModel is done before any data is
    * read from the source. Using the above setValue call with the
    * getValue call doesn't lock both the source and the target
    * before any work is done.
    *
    TypeFactory tf = gbMap.getTypeFactory();
    JITypeDefinitionFactory tdf = tf.getTypeDefinitionFactory();

    JIStructureType srcStructType = (JIStructureType)
srcVar.getJIType();
    JITypeDefinition srcTD = tdf.createJITypeDefinition(srcVar,
srcStructType.getTypeDefinition("SourceString"));

    JIStructureType tgtStructType = (JIStructureType)
tgtVar.getJIType();
    JITypeDefinition tgtTD = tdf.createJITypeDefinition(tgtVar,
tgtStructType.getTypeDefinition("TargetString"));

    gbMap.doAssignment(
```

```
        this,
        srcVar.getTypeDataModel(),
        srcTD,
        tgtVar.getTypeDataModel(),
        tgtTD);
    */
}
else {
    // Set the value of the target GV with a value NOT from another
    // TypeDataModel... The target TypeDataModel is locked
    // (if necessary) before the value is set.
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetString",
        "Some Value...");
}
/* -------- END Set Value InternalString within IBE -------- */
```

Code Example 1f - Set Value of InternalString Array in an IBE

```
/* -------- START Set Value InternalString... -------- */
/* -------- isArray=true within IBE -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetVar");

// Setting array values is done by specifying an array index in the
// JITypeDefinition lookup name... If no index is supplied, “0”
// is assumed to be the index.
for(int i = 0; i < 42; i++) {
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetStringStructure1/TargetArrayString1[" + i + "]",
```

```
        "SET For Index : " + i);
}

// Adding values to an array only requires telling the GBMap to use
// the "ADD" setType. This will add the value to the last array in
// the lookup name (in this case, TargetArrayString1 is
// the last array). If an index is supplied, the ADD acts as a
// pre-insert (adds the value at the supplied index, and pushes
// all values after that index down).
for(int i = 0; i < 42; i++) {
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetStringStructure1/TargetArrayString1",
        "ADD For Index : " + i,
        com.jacada.ji.type.Util.ADD);
}
/* -------- END Set Value InternalString... -------- */
/* -------- isArray=true within IBE -------- */
```

Code Example 1g - Set the Value of an InternalString Sub-element in an IBE Array

```
/* -------- START Set Value InternalString... -------- */
/* -------- within a isArray=true IBE -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetVar");

// Setting array values is done by specifying an array index in the
// JITypeDefinition lookup name... If no index is supplied, “0”
// is assumed to be the index.
for(int i = 0; i < 42; i++) {
    gbMap.setValue(
        this, // The caller...
```

```
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetStringArrayStructure0[" + i +
        "]/TargetSubString1",
        "SET For Index : " + i);
}

// Adding values to an array only requires telling the GBMap to use
// the "ADD" setType. This will add the value to the last array
// in the lookup name (in this case, TargetStringArrayStructure0
// is the last array). If an index is supplied, the ADD acts as a
// pre-insert (adds the value at the supplied index, and pushes
// all values after that index down).
for(int i = 0; i < 42; i++) {
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetStringArrayStructure0/TargetSubString1",
        "ADD For Index : " + i,
        com.jacada.ji.type.Util.ADD);
}
/* -------- END Set Value InternalString... -------- */
/* -------- within a isArray=true IBE -------- */
```

Code Example 2 - Add a Structure to an Array Within an IBE

```
/* -------- START Set Value add Structure... -------- */
/* -------- within a isArray=true IBE -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetVar");

// Setting array values is done by specifying an array index in the
// JITypeDefinition lookup name...
for(int i = 0; i < 42; i++) {
    gbMap.setValue(
```

```
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetArrayStructure0[" + i + "]/
        TargetSubStructure0/TargetStringField",
        "SET For Index : " + i);
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetArrayStructure0[" + i + "]/
TargetSubStructure0/TargetNumberField",
        new Integer(i));
}
// Adding values to an array only requires telling the GBMap to use
// the "ADD" setType. This will add the value to the last array
// in the lookup name (in this case, TargetStringArrayStructure0
// is the last array) If an array index is specified within the
// JITypeDefinition lookup name, the ADD will act as an insert,
// inserting the value at the specified index.
// for each new "i", an empty [NO_VALUE] structure is created.
// When adding to a structure like this, the NO_VALUE values are
// replaced in the order they are found (index count from 0)
// with the passed value. So, 2 separate for loops could be used to
// produce identical results.
for(int i = 0; i < 42; i++) {
    gbMap.setValue(
        this, // The caller...
        tgtVar.getTypeDataModel(),
        "gvTargetVar/TargetArrayStructure0/TargetSubStructure0/
TargetStringField",
        "SET For Index : " + i,
        com.jacada.ji.type.Util.ADD);

// This setValue provides an index for the path and uses
// com.jacada.ji.type.Util.SET as the assignment type. This
// will cause the number value that is added to be placed at the
// same level as the string value from the above setValue.
// If com.jacada.ji.type.Util.ADD was used, another empty
// structure would be created with NO_VALUE for the
// "...../TargetStringField" of the structure, and an Integer
// with the value of "i" for "...../TargetNumberField"
gbMap.setValue(
    this, // The caller...
```

```
    tgtVar.getTypeDataModel(),
    "gvTargetVar/TargetArrayStructure0[" + i +
    "]/TargetSubStructure0/TargetNumberField",
    new Integer(i),
    com.jacada.ji.type.Util.SET);
}
/* -------- END Set Value add Structure... -------- */
/* -------- within a isArray=true IBE -------- */
```

Code Example 3 - Determine the Type of an Element in a BE

```
/* -------- START Determine the JIType of an element in a BE ----- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Source Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceVar");

// Now, we want to determine the JIType of
// "SourceStringStructure0/SourceSubString0"
// Get the JIStructureType of the Source Global Variable.
// This will be the JIStructureType "SourceIBEType"
JIStructureType sourceStructType = srcVar.getJIType();

// Get the JITypeDefinition for the path
// "SourceStringStructure0/SourceSubString0"
String stringPath = "SourceStringStructure0/SourceSubString0";
JITypeDefinition path =
    sourceStructType.getTypeDefinition(stringPath);

// Get the JIType of the path
// This will be the Simple JIType "InternalString"
JIType pathType = path.getJIType();

// Do any work that is necessary
switch(pathType.getTypeIdentifier().intValue()) {
    case JIType.INTERNAL_STRING.intValue():
```

```
    // Do specific work for an InternalString
    break;
    case JIType.INTERNAL_NUMBER.intValue():
    // Do specific work for an InternalNumber
    break;
// case <SOME OTHER TYPE>:
//  break;
default:
  break;
}
/* -------- END Determine the JIType of an element in a BE ----- */
```

Code Example 4 - Determine Whether an Element in a BE is an Array or Not

```
/* ------- START Determine if an element in a BE is an array ------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Source Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceVar");

// Now, we want to determine the if
// "SourceStringStructure0/SourceSubString0" is an array

// Get the JIStructureType of the Source Global Variable.
// This will be the JIStructureType "SourceIBEType"
JIStructureType sourceStructType = srcVar.getJIType();

// Get the JITypeDefinition for the path
// "SourceStringStructure0/SourceSubString0"
String stringPath = "SourceStringStructure0/SourceSubString0";
JITypeDefinition path =
    sourceStructType.getTypeDefinition(stringPath);

// Check if the path is an array
// The AbstractJITypeDefinition (which all JITypeDefinitions extend),
```

```
// implements com.jacada.types.ArrayObject.
com.jacada.types.ArrayObject ao = (com.jacada.types.ArrayObject)path;
boolean isArray = ao.isArray();

// Do work as necessary
if (isArray) {
    // Do some stuff
}
else {
    // Do some other stuff
}
/* -------- END Determine if an element in a BE is an array ----- */
```

Code Example 5 - Copy Data from Structure to Structure

```
/* -------- START Structure to Structure -------- */
// Get the GBMap from the GBMethod
GBMap gbMap = getMethod().getMap();

// Get the Source Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition srcVar = (JITypeDefinition)gbMap.getGlobalVar
("gvSourceVar");

// Get the Target Global Variable from the GBMap. Global Variables
// implement JITypeDefinition, which supplies everything we need
// to get the data we want.
JITypeDefinition tgtVar = (JITypeDefinition)gbMap.getGlobalVar
("gvTargetVar");

// Copy the values from the source GV to the target GV
// The doAssignment method relies on the Type Mapping defined
// between the source JIStructureType and the target JIStructureType
```

```
gbMap.doAssignment(
    this,
    srcVar.getTypeDataModel(),
    null,
    tgtVar.getTypeDataModel(),
    null);
/* -------- END Structure to Structure -------- */
```

Value Classes for setValue Calls

For each of the webMethods JI types, there is a set of regular Java classes that are valid to assign as the value for a JITypeDefinition of that webMethods JI type. When doing assignments using Internal Type to Internal Type, see "Data Type Conversion Rules" on page 252.

- `InternalString`:
  - Any object can be assigned to an `InternalString`. Internally, the `toString()` method is called on the object before it is assigned.
  - `InternalNumber`
  - `java.lang.Number`
  - `java.lang.String` (with formatting conversion)
  - `java.util.Date`
- `InternalDate`:
  - `java.util.Date`
  - `java.lang.String` (with formatting conversion)
  - `java.lang.Number`
- `InternalBoolean`:
  - `java.lang.Boolean`
  - `java.lang.String` (with formatting conversion)
  - `java.lang.Number` (with formatting conversion)
- `InternalObject`:
  - Any Object
- `InternalBinary`:
  - An array of bytes (`byte []`)
  - `java.lang.String` (`String.getBytes()` is called)

Chapter 12. Configuration Manager

The Configuration Manager is a Java-based application included with webMethods JI that is used to configure components in the webMethods JI environment.

Starting the Configuration Manager

The Configuration Manager can be started from the command line in both UNIX and Windows.

> **Note:** If Windows shortcuts were added to the **Start** menu during installation, the Configuration Manager can be started by selecting the shortcuts created during webMethods JI installation.

Command Line

The Configuration Manager can be started from the command line by executing the following command at the command line:

UNIX:

```
cd <JI_install_dir>/bin
./ea_cfgmgr
```

Windows:

```
cd <JI_install_dir>\bin
.\ea_cfgmgr
```

> **Note:** This executable uses a LAX file to specify the runtime behavior of the application. An example of this could be using a different Java Runtime Environment (JRE) than the one selected during webMethods JI installation, it may be necessary to edit the ea_cfgmgr.lax file. For information about editing LAX files, see Chapter 7 - "LAX Files" on page 177 of the Supplemental Reference Guide.

The Configuration Manager Interface

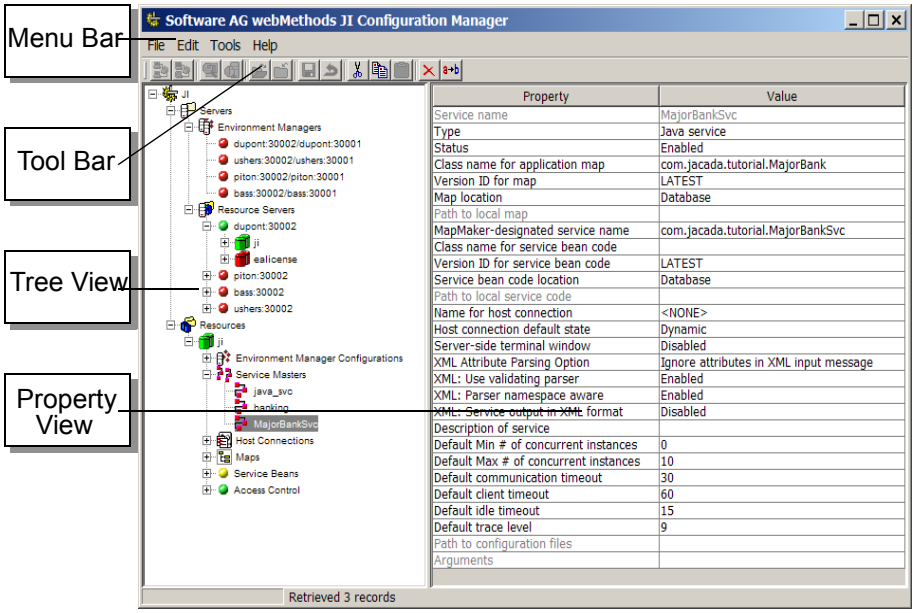


Figure 216. The Configure Manager Interface

Configuration Manager Views

The Configuration Manager interface consists of two primary panes, or views:

- "Tree View" on page 457
- "Property View" on page 457

When using Windows and the CDE/Motif look and feel, a "+/-" symbol before each component in the tree identifies if the node can be expanded to reveal additional, lower-level nodes. A "+" indicates that the node can be expanded, a "-" indicates that the node is already expanded. The absence of both indicates that the node cannot be expanded.

> **Note:** The Metal look and feel uses a root node handle in place of the "+/-" symbols.

Tree View

The left pane of the Configuration Manager interface is the Tree view. This is a graphical representation of hierarchical information, containing “Root” components that can be expanded to expose underlying sub-components (also called “nodes,” or “branches”). These sub-components often can be expanded to reveal their own sub-components, creating a hierarchical tree format.

Property View

The right pane of the Configuration Manager interface is the Property view. This view is used to display the properties and values for various configuration parameters. For example, when you are editing a Resource Database, the Property view displays the properties for the database and their corresponding values. The values of each record can be edited in this view.

Configuration Manager Menus

The Configuration Manager menu bar contains the following menus:

- “File Menu” on page 458
- “Edit Menu” on page 460
- “Tools Menu” on page 461
- “Help Menu” on page 461

Shortcut menus are available throughout the Configuration Manager by right-clicking on a component in the Tree view, or by right-clicking anywhere within the Property view.

The following sections detail the options available in each menu.

File Menu

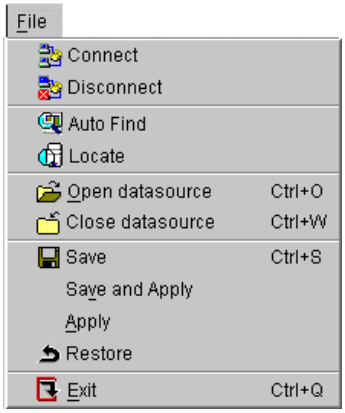


Figure 217. The File Menu

To access the **File** menu using the mouse, click **File** on the menu bar. To use the keyboard, press **Alt + F** and the key that corresponds to the underlined letter of the command.

The **File** menu offers the following options:

Option	Command
Connect	Connects to the selected Resource Server or Environment Manager. Available only when an unconnected Resource Server or Environment Manager is selected in the Tree view.
Disconnect	Disconnects from the selected Resource Server or Environment Manager. Available only when a connected Resource Server or Environment Manager is selected in the Tree view.
Auto Find	Automatically finds all Environment Managers and/or Resource Servers within the sub-net. This option uses the multicast technology, described in “Managing the webMethods JI Environment” on page 29.

Option	Command
Locate	Opens the **Enter Host/Point of server** dialog box to locate a specific Environment Manager or Resource Server. Enter the host and port of a known Environment Manager or Resource Server in this dialog box. **Note:** This is the only way to locate an Environment Manager or Resource Server on other sub-nets without proxy servers.
Open datasource (**Ctrl + O**)	Opens the selected datasource. When a database is open, the Resources node in the Tree view can be expanded to reveal its sub-components.
Close datasourse (**Ctrl + W**)	Closes the selected datasource.
Save (**Ctrl + S)**	Saves changes made to the database properties.
Save and Apply	Saves the changes made to the database, and applies them to any currently running environments.
Apply	Applies the changes made to the database to any running environments.
Restore	Cancels the changes that have been made since the last save.
Exit (**Ctrl + Q**)	Closes the Configuration Manager.

Edit Menu

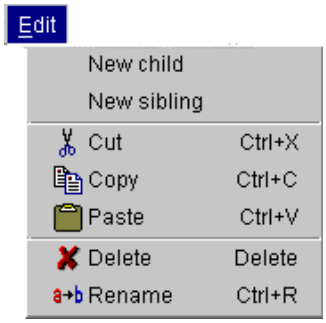


Figure 218. The Edit Menu

To access the **Edit** menu using the mouse, click **Edit** on the menu bar. To use the keyboard, press **Alt + E** and press the key that corresponds to the command's underlined letter. The **Edit** menu offers the following options:

Option	Command
New child	Creates a new component in the tree, subservient to the selected component. **Note:** The name of this menu option varies depending on what item is highlighted in the Tree View.
New sibling	Creates a new component in the tree, at the same level in the tree as the selected component. **Note:** The name of this menu option varies depending on what item is highlighted in the Tree View.
Cut (**Ctrl + X**)	Cuts a selected item into the clipboard and deletes the item.
Copy (**Ctrl + C**)	Copies a selected item to clipboard while leaving the original intact.
Paste (**Ctrl + V**)	Pastes a cut or copied item from the clipboard.
Delete (**Delete**)	Deletes a selected item.

Option	Command
Rename (**Ctrl + R**)	Allows you to rename a selected item.

Tools Menu

Figure 219. The Tools Menu

To access the **Tools** menu, click **Tools** on the menu bar. To use the keyboard, press **Alt + T** and the key that corresponds to the underlined letter of the command.

The **Tools** menu offers the following print options:

Option	Command
Print Table	Selecting this option prints the table shown in the Properties view. The **Print** dialog box opens. Select the proper parameters from the available options. **Note:** The printed table may span across multiple pages.
Print Tree	Selecting this option prints the Tree view. The **Print** dialog box opens. Select the proper parameters from the available options. **Note:** The printed Tree view may span across multiple pages.

Help Menu

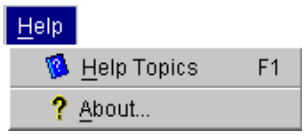


Figure 220. The Help Menu

To access the **Help** menu using the mouse, click **Help** in the menu bar. To use the keyboard, press **Alt + H** and the key that corresponds to the underlined letter of the command.

The **Help** menu offers the following options:

Option	Command
Help Topics	Opens a window containing help for the Configuration Manager.
About	Opens a window listing the version of the Configuration Manager and other information.

Shortcut Menus

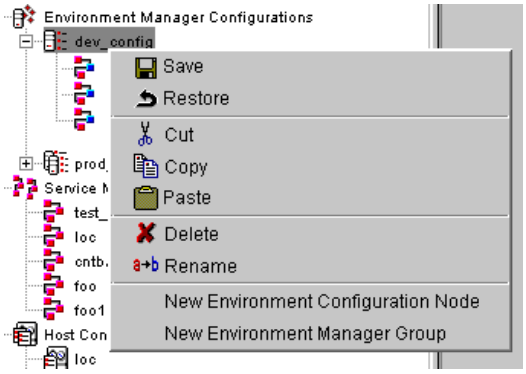


Figure 221. A Shortcut Menu

Shortcut menus are available in many places throughout the Configuration Manager user interface. To open a shortcut menu, right-click on an item in the Tree view or the Property view.

Shortcut menus are context sensitive. They only present options that apply in the current situation. As a result, shortcut menus are not always available, and the options that they present vary.

Configuration Manager Toolbar

Figure 222. The Configuration Toolbar

The Configuration Manager toolbar contains shortcuts to a number of operations included in the menus. The following toolbar buttons are available:

Connect: Connects to an Environment Manager or Resource Server. Available only when a disconnected Environment Manager or Resource Server is selected in the Tree view.

Disconnect: Disconnects from the selected Environment Manager or Resource Server.

Auto Find: Automatically finds all Environment Managers and/or Resource Servers within your sub-net.

Locate: Opens the **Enter Host/Post of server** dialog box, where the host and port of a known Environment Manager or Resource Server can be entered.

Open: Opens the selected database. When a database is open, the Resources node in the Tree view can be expanded to reveal its sub-components.

Close: Closes the selected database.

Save: Saves changes made to the database records.

Restore: Cancels the changes that have been made since to the last save.

Cut: Cuts the selected item into the clipboard and deletes the item.

Copy: Copies the selected item into the clipboard.

Paste: Pastes the cut or copied item from the clipboard.

Delete: Deletes the selected item.

Rename: Allows you to rename the selected item.

Getting Help from Within Configuration Manager

To open a **Help** window, select **Help** > **Help Topics**. A window displays the Help file for the Configuration Manager.

Configuring Environment Managers

About Environment Managers

The Environment Manager is a process that runs within the webMethods JI runtime environment. It acts as the central management and control point for webMethods JI server-side processing.

The Environment Manager performs the following tasks:

- Provides a central access point for clients to communicate with the webMethods JI environment
- Connects clients to the appropriate JClusters
- Load balancing
- Provides a central monitoring point for the entire webMethods JI environment
- Process management, including starting and stopping JClusters

Note: Environment Managers are configured from within the Configuration Manager.

Locating Environment Managers

Before Environment Managers can be configured, the Configuration Manager must locate them.

This is done in one of two ways:

- Automatic location, using multicast technology.
- Manual location, specifying the host and port of the Environment Managers.

Automatically Locating Environment Managers

When the Configuration Manager is first opened, it uses multicast technology to locate any Environment Managers that are currently running on the local sub-net. All Environment Managers that were automatically located will be displayed as subservient to the Environment Managers node in the Tree view.

After the Configuration Managers has been opened, use the following auto location procedure to query all available Environment Managers running in your local sub-net:

1. Select the Environment Manager node in the Tree view.
2. Select **File** > **Auto Find** or click the **Auto Find** button ( ).

   This allows you to automatically locate any Environment Managers in your local sub-net that have been started after the Configuration Manager was started.

> **Note:** If the proxy server is enabled, Environment Managers in other sub-nets are located as well.

3. When the Configuration Manager has located an Environment Manager, connect to the server using the **Connect** button ( ).

For more information about the multicast technology used by webMethods JI, see Chapter 2 - "Managing the webMethods JI Environment" on page 29.

Manually Locating Environment Managers

Specific Environment Managers started after the last automatic location was performed, or those not running on the local subnet, can be located without using the automatic location method. To do this, identify the host and port that the Environment Manager is running on.

To manually locate an Environment Manager:

1. Expand the Servers node in the Tree view.
2. Select the Environment Manager's sub-component.
3. Select **File** > **Locate** or click the **Locate** button ( ).

   This opens the **Enter Host/Port of server** dialog box, where the host and RMI port of the Environment Manager can be entered:

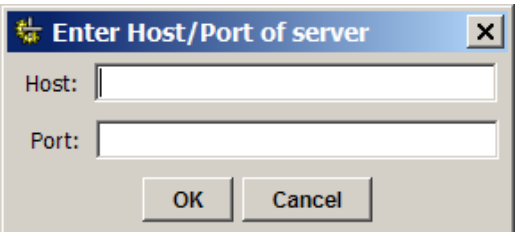


Figure 223. The Enter Host/Port of server dialog box

The **Enter Host/Port of server** dialog box offers the following options:

Option	Description
Host	The host name or IP address of the host on which the Environment Manager is running.
Port	The RMI port number of the Environment Manager.

Note: After locating a server using this method, there is an automatic connection to the Environment Manager. The Configuration Manager then maintains this connection.

Environment Manager Configuration Properties

After one or more Environment Managers are located, the Environment Managers node in the Tree view has sub-components, indicated by the "+" next to the node in the Tree view. Click the "+" to expand the tree if it is not already expanded.

One or more Environment Managers are listed, using the following format:

```
<host>:<RMI_port>/<host>:<server_port>.
```

To edit the configuration properties of an Environment Manager listed in the tree:

1. Select the Environment Manager.
2. Select **File** > **Connect**, or right-click on the Environment Mangers node and select **Connect** from the shortcut menu.

   Once connected, the properties and their corresponding values for the Environment Manager are listed in the Property view.

Note: The properties listed here are also found in the Environment Manager's configuration file, <envmgr_filename>.cfg. Changes made in the Configuration Manager to the Environment Manager's configuration properties are saved in this file.

Additional properties may be configured by modifying the envmgr.cfg file located in the <ji_install_dir>/config directory. See "The Environment Manager Configuration File" on page 131 of the webMethods JI Supplemental Reference Guide for more information on these and other parameters.

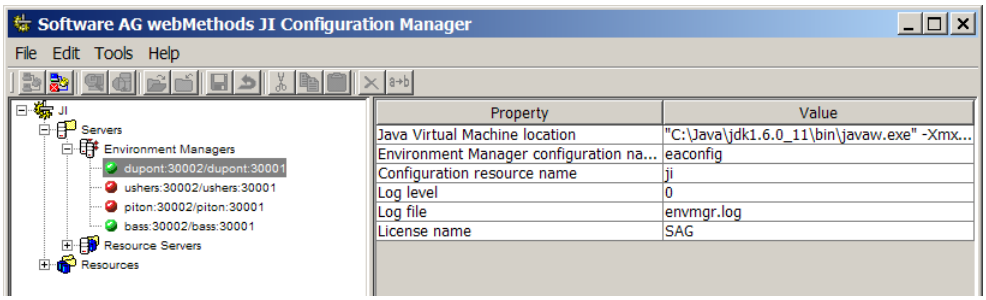


Figure 224. Environment Managers

Note: Environment Managers in a connected state have a green icon; those in a disconnected state have red icons.

The following properties can be configured for the Environment Manager:

Property	Description
Java Virtual Machine location	The path to the Java Virtual Machine (JVM) that the Environment Manager will use. **Note:** If the Java Runtime Environment (JRE) provided with webMethods JI installation was installed and is being used, this property defaults to *./jre/bin/java*.

Property	Description
Environment Manager configuration name	The name of the Environment Manager load balancing configuration as it appears in the Resource Database. This parameter must be set to an existing configuration in the Resource Database otherwise the Environment Manager fails to start. Default: `eaconfig`
Configuration resource name	The name of the Resource Database that contains configuration information for the Environment Manager. The location of the database is defined in the Resource Server configuration (see "Configuring Resource Servers" on page 470). The value of this parameter must correspond to the name of the appropriate resource configured for the Resource Server. Default: `eardb://ea`
Log level	Indicates the amount of information that is logged into the Environment Manager's log file. Valid entries are `0` through `9`, with `0` representing no logging and `9` representing the most verbose logging. **Note:** Setting the log level to 0 turns logging off. Default: `0`

Property	Description
Log file	The name of the Environment Manager's log file. Optionally, this parameter can also set the path to the log file. By default, the location of the log file is *<JI_install_dir>/logs*. Additional valid entries are: • `System.out` • `System.err` **Note:** (These values are mapped to a file or system console window as specified by `lax.stderr.redirect` or `lax.stdout.redirect` parameters within the `ea_envmgr.lax` file) If this parameter is present with no value and the value of the `loglevel` parameter is greater than `0`, logging is sent to `System.out`. Default: `envmgr.log`
License name	If the License name for this Environment Manager is something other than the default name (`SoftwareAG`) or `EmergencyTemporaryKey`, the correct name should be entered here. See "Adding and Editing a License Key" on page 507 for information about the License name.

To change the value of a property, click on the value cell that corresponds to the appropriate property.

Note: Some of the cells contain drop-down lists that become visible once a cell has been selected and some require text entry.

Configuring Resource Servers

About Resource Servers

Resource servers, also called resource providers, are processes that run in the webMethods JI runtime server environment. They act as centralized resource providers, facilitating access to webMethods JI server-side data (such as information stored in the webMethods JI Resource Database). The Configuration Manager can be used to configure Resource Databases and files that are accessed via the Resource Servers.

Locating Resource Servers

In order to edit Resource Databases and other resource files from within the Configuration Manager interface, the Configuration Manager must first locate the Resource Server that provides access to the database and/or files. This is done in one of two ways:

- Automatic location, using multicast technology
- Manual location, specifying the host and port of the Resource Server

Automatically Locating Resource Servers

When the Configuration Manager is first opened, it uses multicast technology to locate any Resource Servers that are currently running on your local sub-net. All automatically located Resource Servers will be displayed as subservient to the Resource Servers node in the Tree view.

After the Configuration Manager has been opened, use the following auto location to query for all available Resource Servers running in the local sub-net:

1. Select the Resource Servers node in the Tree view.
2. Select **File** > **Auto Find** or click the **Auto Find** button ( ).

Note: This locates Resource Servers in the local sub-net if they were started after the Configuration Manager. For more information about the multicast technology used by webMethods JI, see Chapter 2 - "Managing the webMethods JI Environment" on page 29.

Manually Locating Resource Servers

Individual Resource Servers can be located without using the automatic location method by identifying the host and port on which the Resource Server is running.

> **Note:** Resource Servers that are not in the local subnet, or a specific Resource Server started after the Configuration Manager performed the last automatic location, can be located in the same manner.

To manually locate a Resource Server:

1. Expand the Servers node in the Tree view.
2. Select the Resource Servers sub-component.
3. Select **File** > **Locate** or click the **Locate** button ( ).

This opens the **Enter Host/Port of server** dialog box, where the host and port of the Resource Server are identified:

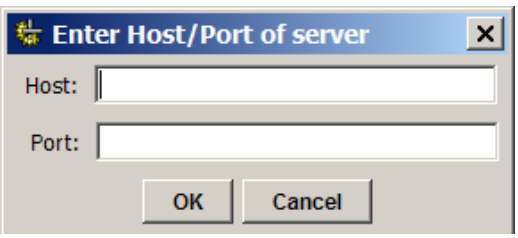


Figure 225. The Enter Host/Port of Server Dialog Box

The **Enter Host/Port of server** dialog offers the following options:

Option	Description
Host	The host name or IP address of the host on which the Resource Server is running.
Port	The port number of the Resource Server.

After one or more Resource Servers are located using either method described above, the Resource Server component in the Tree view has sub-components, indicated by the "+" next to the component in the tree. Click on the "+" to expand the tree. One or more Resource Servers are listed, using the format `<host>:<RMI_port>`. Each Resource Server sub-component can also be expanded to view the list of all the Resource Databases and other resources served by the Resource Server.

To view resources served by the Resource Server using multicasting, select the Resource Server and click the **Connect** button ( ).

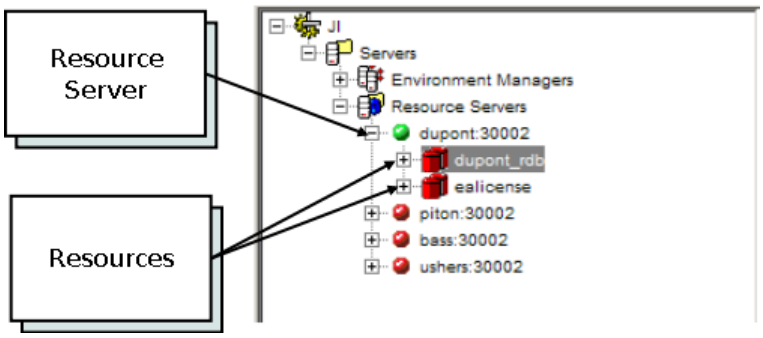


Figure 226. Resource Server

Resource Server Configuration Properties

To edit the configuration properties for one of the Resource Servers listed in the tree, select the Resource Server and select **File** > **Connect**, or right-click and select **Connect** from the shortcut menu. Once connected, the properties and their corresponding values for the Resource Server are listed in the Property view.

> **Note:** The properties listed here are also found in the Resource Server configuration file, ressvr.cfg. Changes made to the Resource Server configuration properties are saved in this file.

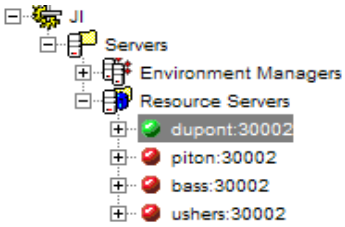


Figure 227. HOST3:30002 is connected

> **Note:** Resource Servers that are in a connected state have a green icon; those in a disconnected state have a red icon.

The following properties can be configured for the Resource Server:

Property	Description
Log file	The name of the Resource Server log file. Optionally, this parameter can also set the path to the log file. By default, the location of the log file is *<JI_install_dir>/logs*. Additional valid entries are: • `System.out` • `System.err` **Note:** If no value is entered for this parameter, and the value of the `loglevel` parameter is greater than `0`, logging is sent to `system.out`. Default: `ressvr.log`
Log level	The level that information is logged to the Resource Server log file. Valid entries are `0` through `9`, with `0` representing no logging and `9` representing the most verbose logging. Setting the Log level to `0` turns logging off. Default: `0`
Maximum logbuffer size	The maximum number of lines that are buffered internally before being written to the file.
RMI port #	The port number on which the Resource Server RMI (Remote Method Invocation) registry is running. If the Resource Server is running on the same machine as an Environment Manager, this port number can be the same for both applications. **Note:** If the same RMI port is used for the Environment Manager and the Resource Server and the Resource Server goes down, the Environment Manager has to be restarted.

Property	Description
Multicast proxy hub address	The host name or IP number and port of the hub proxy server, running on another sub-net, with which this server will communicate. The format is `host:port`. When this parameter is set, this Resource Server serves as a "satellite" proxy server and communicates via RMI (Remote Method Invocation) with the hub proxy server identified here. For more information about proxy server configuration, see "Proxy Server" on page 76.
Alternate multicast proxy hub address	A comma-separated list of host names or IP addresses and ports of alternate hub proxy servers. The format is `host1:port1, host2:port2, hostn:portn`. For more information about proxy server configuration, see "Proxy Server" on page 76.

Note: To change the value of a property, click on its value. Some cells contain drop-down lists that become visible once the cell has been selected. Some cells require text entry.

Configuring Resources

About Resources

Individual resources controlled by the selected Resource Server are listed subservient to it in the Server node. Resources can include Resource Databases, license files, and other information served by the Resource Server. To edit a resource listed under the Resource Server sub-component, open the resource. After a Resource Database has been opened, the configurable records in the database are listed in the Resources node of the Configuration tree.

After a Resource Server has been located, the Resource Server component in the Tree view has one or more resource sub-components, indicated by the "+" next to the Resource Server component in the Configuration tree. Click on the "+" to expand the tree. Subservient resources are listed. The following is a description of the various types of resources.

Resource Databases

The webMethods JI Resource Database, implemented using the H2 Database Engine, is used to store runtime environment configuration information, generated maps, and runtime Java service code.

> **Note:** For C services developed using previous versions of webMethods JI, the Resource Database stores Protocol Agent definitions, screen definitions, and service metadata.

For information about configuring databases, see "Configuring Databases" on page 477.

JI License Files

webMethods JI license files are controlled by the Resource Server and are configured from within the Configuration Manager as well. For information about editing license files, see "Licensing" on page 504.

Adding Resources to the Configuration Manager Tree

To add resources to the Configuration Manager tree:

1. Select the Resource Server sub-component for the new resource.
2. Right-click and select **New Data Source** from the shortcut menu.

> **Note:** **New Data Source** can also be selected from the **Edit** menu, and you can select another resource in the tree and select **New Data Source**.

After adding a new resource, a text entry box is added to the Configuration Manager tree. Enter a name for the new resource in this text field. To rename a resource, select the resource and select **Edit** > **Rename** and enter the new name. After adding a resource, change the resource configuration parameters to the appropriate settings.

> **Note:** If a data source is added, the Resource Server must be restarted before the change takes effect.

Resource Configuration Parameters

After creating the resource, identify its location. Select the resource name in the Tree view, and the properties for the resource are listed in the Property view.

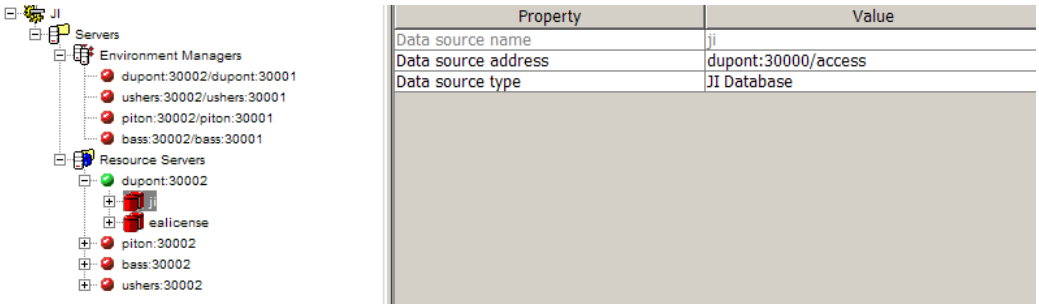


Figure 228. Resource Configuration Parameters

The following properties can be configured for resources:

Property	Description
Data source name	The name of the resource. This property cannot be edited.

Data source address	The URL of the resource. Allowable values include:
	• For a Resource Database: `<host>:<port>/<database_name>`.
	• For a license file: `<directory>/<filename>.txt`. Examples:
	  • UNIX: `/opt/ea/ealicense.txt`.
	  • Windows: `C:\ea\ealicense.txt`.
	Note: If the license file is in its default location, *<JI_install_dir>/config*, you do not need to enter the path to the file

Data source type	The type of data source identified by this resource. Valid options are JI Database and JI License file.

To change the value of a property, click on the value cell that corresponds to the appropriate property and enter the new value. For information about Licensing, see “Licensing” on page 504.

Configuring Databases

The Configuration Manager provides the ability to configure various webMethods JI server components within the Resource Database. The configurable resources include:

- Environment Managers
- Service Masters
- Host Connections

In addition, the maps and custom EAServiceBeans that have been deployed to the database can be viewed and deleted.

Opening Resource Databases for Editing

Open the database to edit the settings for resources. To open a Resource Database, expand the Resource Server component in the Tree view. One or more databases are listed as subservient to the Resource Server.

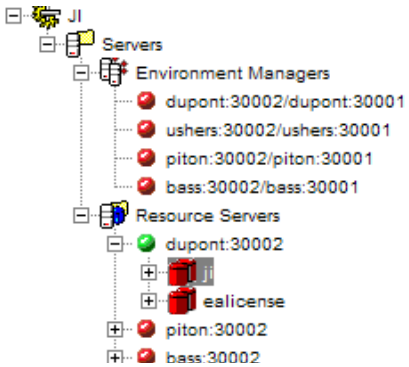


Figure 229. Opening Resource Databases for Editing

Select the datasource in the Resource Server tree and select **File** > **Open datasource** or click the **Open** button ( ).

> **Note:** Databases in an open state have a green icon; those in a closed state have a red icon.

After the Resource Database is opened, the Resources node in the Configuration Manager tree expands to included the database and all of its sub-components. All open databases are listed under the Resources node.

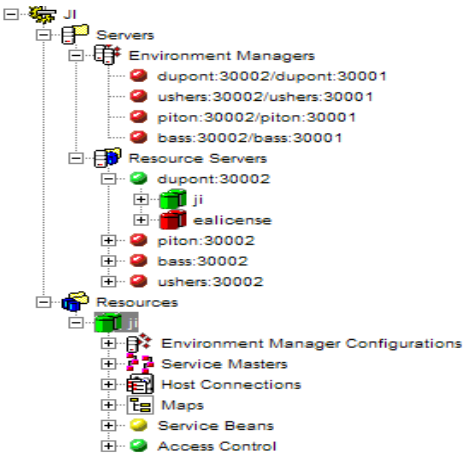


Figure 230. The Expanded Resources Node

Each sub-component and its configurable items are described below.

Environment Manager Configurations

The Environment Manager Configurations sub-component in the Resources node is used to configure load balancing, JCluster behavior, Environment Manager log settings, and service instances. The Environment Manager tree consists of the following information:

- **Configurations:** One or more configurations can be established to manage load balancing, logging, JCluster and service behavior, and other information. Multiple Environment Managers can subscribe to the same configuration. This simplifies configuration of identical Environment Managers. When an individual instance of an Environment Manager is launched, its configuration is identified in the Environment Manager's configuration file using the parameter `configname=<configname>`. This parameter can be overridden at the command line using the `-f <configname>` command line option. Using either method, the name entered for `<configname>` must correspond to an existing Environment Manager configuration in the Resource Database, or the Environment Manager will fail to launch.
- **Service Detail:** Subservient to each configuration, individual services can be configured to override the Service Master information.

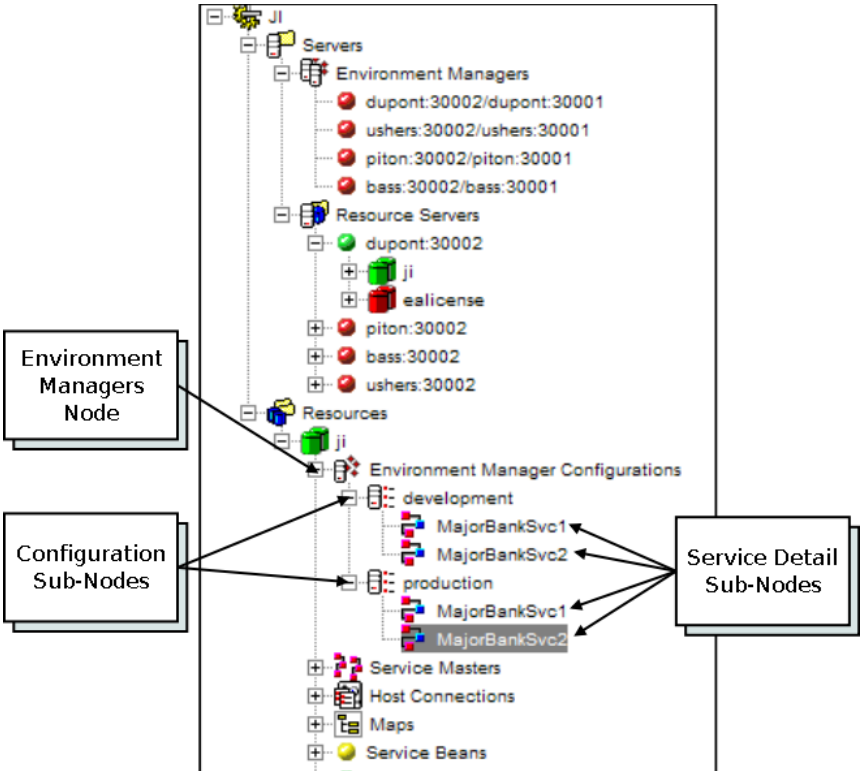


Figure 231. Service Detail

Each of these sub-components is described in detail below.

Environment Manager and JCluster Configuration

Configurations are used to set parameters for JCluster management, load balancing options, and other information. This configuration information is then stored in the Resource Database and is used at runtime by Environment Managers for load balancing purposes. A group could include high-performance and low-performance machines. A configuration could be set for the high-performance machines, with a higher maximum number of JCluster processes and other information. A second configuration could then be set for the low-performance machines with a lower maximum number of JClusters. This would allow for better fine-tuning of load balancing options.

Configurations also include service detail information, used to configure how individual services will behave within Environment Managers. This is configured in the Service Detail sub-components. For more information about service detail information, see “Service Details” on page 486.

The configuration for each individual Environment Manager is defined in the Environment Manager’s text-based configuration file. This file can be edited manually to identify the configuration and other information. It can also be edited from within the Configuration Manager interface if the Environment

Manager is running. See “Configuring Environment Managers” on page 464 for information about editing this file from within the Configuration Manager interface.

Adding Configurations

Configurations are added to the Resources tree by highlighting the Environment Manager Configurations sub-component, right-clicking, and selecting **New Environment Configuration Node**. Enter the name for the new configuration. This adds the new configuration to the Resource Database.

> **Note:** Configuration names must be unique across your entire network.

Copying and Pasting Configurations

Configurations can be copied and pasted to minimize the amount of effort required to create each configuration. One configuration could be created for high-performance machines and their service details. Rather than having to re-create all the service details for each additional configuration, copy the first configuration and paste a copy of it into the appropriate group. After pasting it, make any necessary changes to the configuration and the service details.

To copy a configuration, right click the configuration in the Tree view and select **Edit** > **Copy**. To paste, right click the appropriate Group node and select **Edit** > **Paste**. A new configuration is included in the Tree view. Enter a unique name for the new configuration.

Load Balancing Groups

Multiple Environment Managers can be grouped together for load balancing purposes. For example, a group of Environment Managers can be established that encompasses all production Environment Managers running on the network. This way, the load can be balanced among all production servers.

At the same time, another group of Environment Managers (or a single Environment Manager) can be included in a separate group for development, training, or testing purposes. These separate Environment Managers can be running on the same network but will not be included in the load balancing for the production servers. This allows for multiple load balancing options within the same network. For more information about load balancing, see Chapter 2 - "Managing the webMethods JI Environment" on page 29.

The group for each individual Environment Manager is defined in its subcomponent of the Servers node (see “Configuring Environment Managers” on page 464). It can also be passed at the command line when the Environment Manager is started. A default group name is provided for each Environment Manager configuration.

Setting Properties for Configurations

To set properties for configurations, highlight the configuration in the Tree view. The Property view displays a list of properties and their values.

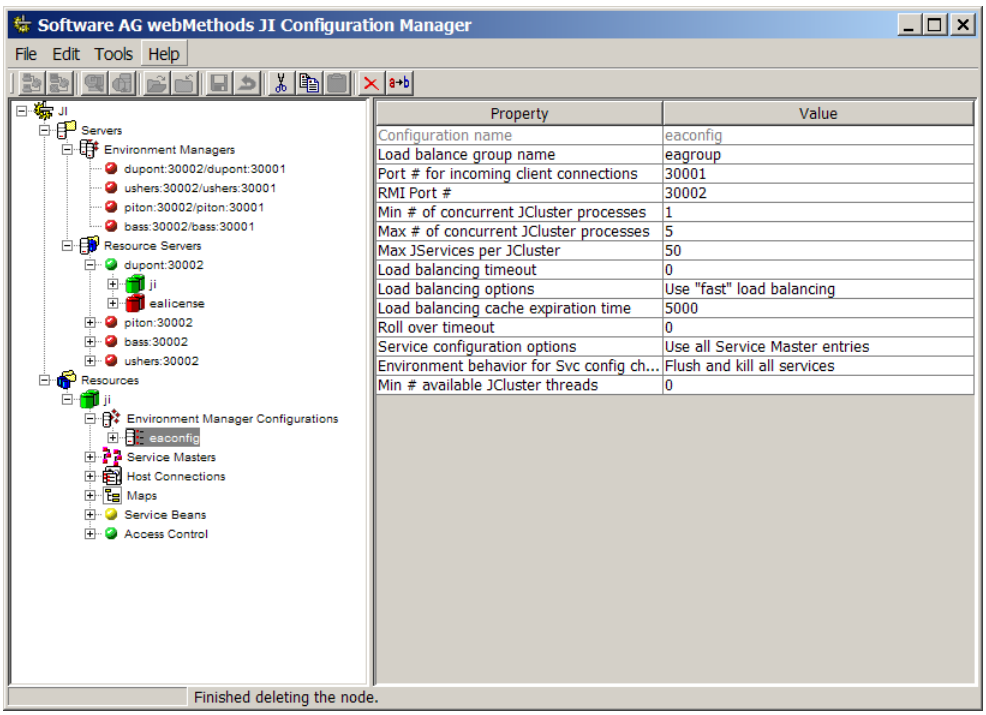


Figure 232. Setting Properties for Configurations

The Property view displays a list of properties and their values:

Property	Value
Configuration name	The name of the configuration. This field cannot be edited.
Load balance group name	The name of the Environment Manager's load balancing group. See "Load Balancing Groups" on page 480 for more details about load balancing groups.
Port # for incoming client connections	The port number the Environment Manager uses to listen for client Requests for Services.

Property	Value
RMI Port #	The port number on which the Environment Manager's RMI (Remote Method Invocation) registry is running. If the Environment Manager is running on the same machine as a Resource Server, this port number can be the same for both applications. **Note:** If the same RMI port is used for both, and the Resource Server goes down, the Environment Manager has to be restarted.
Min # of concurrent JCluster processes	The minimum number of JCluster processes that are started when the Environment Manager is started. Default: 1
Max # of concurrent JCluster processes	The maximum number of JCluster processes that can be started by the Environment Manager. Default: 5
Max JServices per JCluster	The maximum number of JServices that are allowed within each JCluster. When the maximum number of JServices are running within a JCluster, a new JCluster is started when the Environment Manager receives the next Request for Service. **Note:** When the maximum number of JCluster processes has been reached, no new JClusters are started. Default: 50

Property	Value
Load balancing timeout	The amount of time, in milliseconds, the Environment Manager waits for load balancing requests to return on the multicast channel. When this timeout has been reached, the Environment Manager uses its load balancing information, along with the information retrieved from any or all of the other Environment Managers in the load balancing group, to route the client's Request for Service to the appropriate JCluster. **Note:** This property allows for load balancing tuning. A larger value, such as 2000 milliseconds, provides for better load balancing. A smaller number, such as 100 milliseconds, provides faster performance for the client's Request for Service, but provides less precise load balancing. A value of 0 (zero) disables load balancing between other Environment Managers. Default: 0 milliseconds (Load balancing disabled)
Load balancing options	This property provides a drop-down box with two options. Click in the **Value** field to select the appropriate option. • **Use "fast" load balancing:** After a load balancing request, the Environment Manager uses information from the first respondent to determine load balancing. • **Use "accurate" load balancing:** After a load balancing request, the Environment Manager waits the full Load Balancing Timeout duration, and then uses information from all respondents to determine load balancing. Default: Use "fast" load balancing
Load balancing cache expiration time	The amount of time, in milliseconds, that load balancing information is cached by the Environment Manager. Default: 5000 milliseconds

Property	Value
Roll over timeout	Maximum amount of time, in milliseconds, to wait for services to expire before they are killed while a JCluster is being shut down. Default: 0
Service configuration options	This drop-down box contains two options. Click in the **Value** field to select the appropriate option. • **Use all Service Master entries:** All services configured in the Service Masters node are available for Environment Managers using this configuration. **Note:** If this option is selected, the latest version of the JService is used. • **Use configured service entries for this environment:** Only the services specifically configured for this configuration are available. Default: Use all Service Master entries

Property	Value
Environment behavior for Svc config changes	Determines how the Environment Manager behaves when there have been changes to the Service Master or Service Detail records in the Resource Database. • **Flush and kill all services:** Instructs the Environment Manager to refresh all configuration data and identify all instances of affected services. Those instances that do not have clients currently connected are killed, those that do have clients are killed upon client disconnect. • **Flush only:** Instructs the Environment Manager to refresh all configuration data but do nothing to currently running services. • **Flush and restart applicable JClusters:** Instructs the Environment Manager to refresh all configuration data, and restart any JClusters that are configured to use the service. When the JCluster is stopped, it either waits until no clients are connected or waits for its configured roll over timeout. • **Do nothing:** Instructs the Environment Manager to do nothing after changes have been made. If this option is selected, the Environment Manager must be restarted before any changes take effect. **Note:** If changes are made to the Resource Database while your webMethods JI environment is running, the changes can be applied by selecting **File** > **Apply** or right-clicking and selecting **Apply** from the shortcut menu. Default: Flush and kill all services
Min # of JCluster Threads	Defines the minimum number of threads available in the pool. These threads are always kept running even after the JServices which use them go away, allowing multiple JService instances to reuse a single thread. The thread pool grows UP TO the minimum, the minimum value does not automatically start “minimum” threads.

Service Details

Service details allow configurations to override the JService configurations, defined in the Service Masters node. This allows the Service Masters to be configured to provide "default" configurations for each service. These settings can then be overridden for certain configurations. For example, a number of configurations based on the quality of the host machines may be included in the configurations. If most of the host machines are high-performance, use Service Masters to configure the services. Then override these settings for the configuration that applies to the lower-performance machines by using service details.

For more information about the Service Masters node, see "Service Masters" on page 489.

Adding Service Details

Service Details are added to an Environment Manager configuration tree by highlighting the Environment Manager configuration's node, right-clicking, and selecting **New Service Detail record**. From the **Choose a MasterService** dialog box, select the name of the service master that provides the default configuration for this service detail.

Copying and Pasting Service Details

Service Details can be copied and pasted to minimize the amount of effort required to create each service detail. After creating a service detail, copy and paste it into the same configuration and rename it, or paste it into another configuration.

To copy a service detail, highlight the service detail in the Tree view and select **Edit** > **Copy** or click the **Copy** button ( ).

To paste, highlight the appropriate configuration node and select **Edit** > **Paste** or click the **Paste** button ( ). A new service detail is included in the Tree view. Enter the correct service name for the new service detail, if appropriate.

Setting Properties for Service Details

To set properties for service details, highlight the configuration in the Tree view.

The Property view displays a list of properties and their values:

Property	Value
Service name	The name of the service. This field is cannot be edited.

Property	Value
Configuration name	The name of the configuration in which the service detail is included. This field is not editable.
Status	Indicates if the service is enabled or disabled. To change the setting, click in the **Value** field and select the appropriate option. Default: Enabled
Server-side terminal window	If set to Enabled, instructs the service to generate a terminal (debug) window that displays the service's interaction with the legacy application. The debug window displays the host machine the Environment Manager is running on. **Note:** For UNIX, if the $DISPLAY environment variables are set, the display is sent to another machine. The behavior of this option varies depending on the setting for the Service Master's Host Connection Default State and the Service Detail's Minimum Number of Concurrent Instances records. If the Service Detail's Minimum Number of Concurrent Instances is set to any non-zero number, the following occurs based on the setting of the Service Master's Host Connection Default State record: • If the Host Connection Default State is set to Dynamic, the Server-side terminal window opens when the service makes a connection to the host. • If the Host Connection Default State is set to Pooled, the terminal window opens when the service starts. **Note:** Server-side terminal windows consume large amounts of CPU power on the server and should only be used for development or debug purposes, and should generally not be used for production services. Defaults to the setting of the Service Master

Property	Value
Min # of concurrent instances	The minimum number of instances of this service that are started when the Environment Manager is started. Defaults to the setting of the Service Master
Max # of concurrent instances	The maximum number of instances of this service that can be started by an Environment Manager. Defaults to the setting of the Service Master
Communications timeout	The amount of time, in seconds, that the service waits for a response from the host before generating an error. Defaults to the setting of the Service Master
Client timeout	The amount of time, in seconds, that the service waits after a client invokes a method before the service sends a message to the client indicating that the service is not able to complete its method invoke request. When this value is set to zero, the `EA_TIMEOUT` setting in the client code is used. When a value other than zero is used the `EA_TIMEOUT` setting is not used. Setting both the default client timeout and the `EA_TIMEOUT` to zero causes the client to wait indefinitely. Defaults to the setting of the Service Master
Idle timeout	The amount of time, in seconds, that the service runs without activity. Because the Environment Manager polls the service for activity every five seconds, the actual timeout may be up to five seconds longer than the number entered here. **Note:** `0` indicates that the service goes away as soon as the client disconnects, `-1` or any negative number indicates that the service never goes away. Defaults to the setting of the Service Master

Property	Value
Trace level	The default level that information for this service are logged to the log file. Valid entries are `0` through `9`, with `0` representing no logging and `9` representing the most verbose logging. **Note:** Setting the loglevel to `0` turns logging off. Defaults to the setting of the Service Master

Service Masters

The Service Master sub-component in the Resources node is used to configure individual JServices. A JService is a thread that runs within the JCluster process. JServices serve as interfaces to webMethods JI Java services that were developed in MapMaker or C services developed using previous versions of webMethods JI.

During deployment, MapMaker creates (and populates) a Service Master Record if one does not exist. A number of the settings configured for Service Masters can be overridden by the Service Details (defined in the Environment Managers node), described above. See “Service Details” on page 486 for more information.

Adding Service Masters

To add Service Masters to the Configuration tree, select the Service Masters node, right-click, and select **New Service Master Record**. Enter the name of the service.

Copying and Pasting Service Masters

Service masters can be copied, pasted and named to minimize the amount of effort required to create each Service Master.

To copy and paste a Service Master:

1. Select the Service Master in the Tree view.
2. Select **Edit** > **Copy** or click the **Copy** button ( ).
3. Select the Service Masters node.
4. Select **Edit** > **Paste** or click the **Copy** button ( ).

   A new Service Master is included in the Tree view. Enter the correct service name for the new Service Master.

Setting Properties for Service Masters

To set properties for Service Masters, select the configuration in the Tree view.

The Property view displays a list of properties and their values:

Property	Value
Service name	The Service Master's name. This field cannot be edited.
Type	This is a drop-down box containing two options. Click in the **Value** field to select the appropriate option. • **Java service:** Select this option if the service is developed in MapMaker. • **C service:** This option is provided for C services used in conjunction with the compatibility kit, included with webMethods JI to provide compatibility with webMethods JI 2.x services. Default: Java service
Status	Indicates if the service is enabled or disabled. To change the setting, click in the **Value** field and select the appropriate option. Default: Enabled
Class name for application map	The full class name for the map file generated by MapMaker. This is an editable drop-down box containing class names for map files generated by MapMaker. Select one of the class names shown, or type in the class name in the format of: *package.map*, for example, *com.ea.map_name*. This name is limited to 128 characters. **Note:** This property applies to Java services only. If a custom class was created for the map, the name of the custom map class, rather than the name of the map as defined in MapMaker, should be used for this record.

Property	Value
Version ID for map	The version number of the map, identified during deployment in MapMaker. Available options are `LATEST` or the version number. **Note:** This property applies to only Java services when the map is deployed in the database.
Map location	This property provides three options, listed in a drop-down box. Click in the **Value** field to select the appropriate option. • **Database:** Indicates that the map is stored in the database. • **Local file:** Indicates that the map is stored in a file on the local file system. • **Classpath:** Indicates that the map is stored in a location identified in the class path used when the Environment Manager is launched. **Note:** This property applies to Java services only. Default: Database
Path to local map	The path to the map on the local file system. This path is limited to 255 characters. **Note:** This property applies only to Java services when Map location is set to Local file.
MapMaker-designated service name	The full class name of the service as entered in the **Services** tab of the Tree view in MapMaker. This is an editable drop-down box containing classnames generated by MapMaker. Select one of the classnames shown, or type in the class name in the format of: *package.service,* for example, *com.ea.service_name*. This name is limited to 50 characters. **Note:** This property applies to Java services only. If a custom class was created for the service, the name of the custom service class, rather than the name of the service as defined in MapMaker, should be used for this record.

Property	Value
Class name for service bean code	The full class name of custom code that has been used to extend the webMethods JI service bean (EAServiceBean) code. This is an editable drop-down box containing EAServiceBean custom code class names generated by MapMaker. Select one of the class names shown, or type in the class name in the format of: *package.serviceBean*, for example, *com.ea.serviceBean_name*. This name is limited to 128 characters. **Note:** This property applies to Java services only.
Version ID for service bean code	The version number of the custom EAServiceBean, identified during deployment in MapMaker. Available options are `LATEST` or the version number. **Note:** This property applies to Java services only, and only when custom code has been used to extend the EAServiceBean and is deployed in the database.
Service bean code location	This drop-down box contains three options. Click in the **Value** field to select the appropriate option. • **Classpath:** Indicates that the service code is stored in a location identified in the class path used when the Environment Manger is launched. • **Database:** Indicates that the service code is stored in the database. • **Local file:** Indicates that the service code is stored in a file on the local file system. **Note:** This property applies to Java services only, and only when custom code has been used to extend the EAServiceBean. Default: Database

Property	Value
Path to local service code	The path to the service code on the local file system. This path is limited to 255 characters. **Note:** This property applies to "C" services or to Java services when Service code location is set to Local file and custom code has been used to extend the EAServiceBean.
Name for host connection	The name of the host connection. Select the name of a configured host connection. See "Host Connections" on page 498. **Note:** This option applies to Java services only.
Host connection default state	Determines whether the host connection is Dynamic or Pooled. • **Dynamic:** A host connection is created when the client connects to the service. When the client disconnects, the service automatically disconnects from the host but remains idle until the idle timeout has passed. • **Pooled:** The host connection is created when the JService is started by the Environment Manager. The host connection closes when services exit. Default: Dynamic

Property	Value
Server-side terminal window	If set to Enabled, instructs the service to generate a terminal (debug) window that displays the service's interaction with the legacy application. The debug window displays on the host machine on which the Environment Manager is running. **Note:** In UNIX, the $DISPLAY environment variable sends the display to another machine. The behavior of this option varies depending on the setting for the Host Connection Default State and Default Minimum Number of Concurrent Instances records. If Default Minimum Number of Concurrent Instances is set to any non-zero number, the following occurs based on the setting of the Host Connection Default State record: • If Host Connection Default State is set to Dynamic, the Server-side terminal window opens when the service makes a connection to the host. • If the Host Connection Default State is set to Pooled, the terminal window opens when the service starts. **Note:** Server-side terminal windows consume large amounts of CPU power on the server and should only be used for development or debug purposes, and should generally not be used for production services. Default: Disabled

Property	Value
XML AttrParsingOptions	As the XML parser traverses through the input XML data and generates the Map/List for the service method, it uses this property to determine how to handle XML attributes, and then generate the resulting Map/List structure accordingly. • Ignore all attributes in the XML input message. Any attributes encountered while parsing the XML input message are ignored and not included in the resulting Map/List. • Parse attributes into fields in the resulting Map. Attributes encountered while parsing are included as fields in the resulting Map/List. These attributes are treated as sub-elements of the XML tag to which they belong. • Parse attributes into fields in a separate Map. Attributes encountered while parsing are included as fields in a separate Map. The name of this map is “EA_XML_ATTR”. The service developer may choose to write custom code to access this attribute data. **Note:** This option applies to Java services only. Default: Ignore all attributes in the XML input message
XML: Use Validating Parser	Specifies whether the XML parser should act as a validating or non-validating parser. • Enabled - Parser is validating • Disabled - Parser is non-validating **Note:** This option applies to Java services only. Default: Enabled

Property	Value
XML: Parser Namespace Aware	Specifies whether the XML parser should be aware of name spacing while parsing. The XML namespace mechanism extends the XML data model to allow element type names and attribute names to be qualified with a Uniform Resource Identifier (URI). Thus, a document that describes the title of a person can use `title` qualified by one URI, and a document that describes the title of books can use `title` qualified by another URI. When this option is disabled, duplicate element type names or attribute names in an XML document result in a potential conflict. • Enabled - Parser is name-space aware • Disabled - Parser is not name-space aware **Note:** This option applies to Java services only. Default: Enabled
XML: Service Output in XML Format	If there is no EA_XML field in the output map from the service, this property is used to determine if the output should be in XML format. • Enabled - The entire output map is converted into XML and sent back in one field, called EA_XML • Disabled - The entire map is sent back to the client unchanged **Note:** This option applies to Java services only. Default: Disabled
Description of service	Description of the service. The description is limited to 60 characters.
Default Min # of concurrent instances	The minimum number of instances of this service that are started when the Environment Manager is started. Default: 0
Default Max # of concurrent instances	The maximum number of instances of this service that can be started by an Environment Manager. Default: 10

Property	Value
Default communication timeout	The amount of time, in seconds, that the service waits for a response from the host before generating an error. Default: 30
Default client timeout	The amount of time, in seconds, a client waits for the service to respond to a service method invoke. When this value is set to zero, the `EA_TIMEOUT` setting in the client code is used. When a value other than zero is used, the `EA_TIMEOUT` setting is not used. **Note:** Setting both the default client timeout and the `EA_TIMEOUT` to zero causes the client to wait indefinitely. Default: 60
Default idle timeout	The amount of time, in seconds, that the service runs without activity. Because the Environment Manager polls the service for activity every five seconds, the actual timeout may be up to five seconds longer than the number entered here. **Note:** `0` indicates that the service goes away as soon as the client disconnects, `-1` or any negative number indicates that the service never goes away. Default: `-1`
Default trace Level	The default level that information for this service is logged to the log file. Valid entries are `0` through `9`, with `0` representing no logging and `9` representing the most verbose logging. **Note:** Setting the loglevel to `0` turns logging off. Default: `0`
Path to configuration files	The path to the Compatibility Kit configuration files. If no value is set the server uses *<JI_install_dir>/config*. This value is limited to 60 characters. **Note:** This option applies to C services only.

Property	Value
Arguments	Any arguments that should be passed to the service when it is started. This value is limited to 60 characters. **Note:** This option applies to C services only.

Host Connections

The Host Connections sub-component in the Resources node is used to configure individual host connections. This provides dynamic changes of host connections. For example, if the IP address or name of the telnet server changes, the host connection information in the Host Connection node can be changed without making any changes to the service code.

Note: The host connections apply to Java services only. For "C" services from previous versions of webMethods JI, the host connection is defined in the Resource Editor, included with the Compatibility Kit.

For more information about the Compatibility Kit, see the webMethods JI Compatibility Kit User Guide.

Adding Host Connections

Host connections are added to the Configuration tree by selecting the Host Connections node, right-clicking, and selecting **New Host Connection record**. Enter the name of the host connection. This adds the new host connection to Configuration tree.

Setting Properties for Host Connections

To set properties for host connections, select the configuration in the Tree view. The Property view displays a list of properties and their values.

Property	Value
Symbolic name for host connection	The name of the host connection. This field is not editable.

Property	Value
Host name (or IP address)	The IP address or name of the TN3270, TN5250, or Character Mode (Telnet) server.
Port number	The port number or name of the TN3270, TN5250, or Character Mode (Telnet) server.
TN3270E connect name	TN3270 connections: The Device Name sent to the TN3270E server. TN5250 connections: The Session name sent to the TN5250 server. Character Mode (Telnet) connections: The Terminal Type String sent to the Telnet server. **Note:** For Character Mode (Telnet) connections, changing the Terminal Type String is not recommended and may cause the service to work incorrectly.

Deployed Maps and Custom EAServiceBeans

The Resources tree also provides the ability to view and delete webMethods JI maps and custom EAServiceBeans deployed to the Resource Database. Maps and EAServiceBeans can be cut or copied from one database and pasted into another. These maps and custom EAServiceBeans are deployed as JAR files from within the MapMaker user interface (see "Deploying the Map and/or Custom EAServiceBean" on page 419).

Deployed Maps

Viewing Deployed Maps

To view the Maps that have been deployed to the Resource Database, expand the Maps node. The full class name of all Maps deployed to the database are listed under this node. Each Map can be expanded to display the versions of the Map that have been deployed.

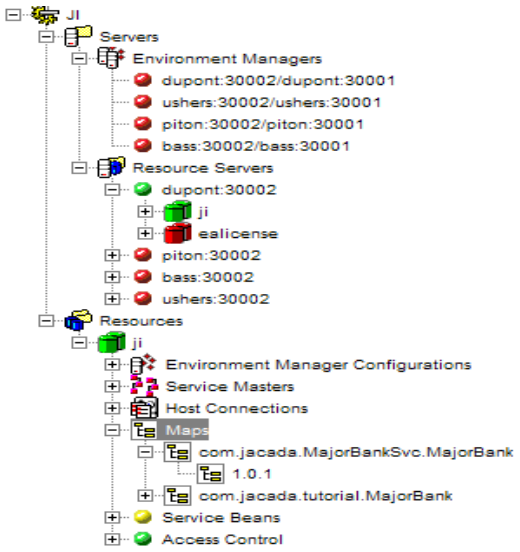


Figure 233. Viewing Deployed Maps

Deleting Maps

There are two ways to delete maps that have been deployed to the Resource Database:

- Deleting all versions of the map.
- Deleting only one version of the map.

To delete all versions of the map:

1. Highlight the map in the Tree view.
2. Right-click, and select **Delete** from the shortcut menu.

   A message indicates that all versions of the map will be deleted.
3. Select **OK**.

> **Note:** If there are any Service Masters that use this map, a message warns you of this fact. To proceed with deleting the map, select **OK**, otherwise, select **Cancel**. The Service Master will still exist but will not be able to execute because its map no longer exists in the database.

To delete only one version of the map:

1. Highlight the version node subordinate to the map.
2. Right-click, and select **Delete** from the shortcut menu.

Note: If there are any Service Masters that use this map, a message warns you of this fact. Because the Service Master does not necessarily know which version of the map it uses, this warning message appears whenever the map is used by a Service Master, regardless of whether or not the version is being used. To proceed with deleting the map, select **OK**, otherwise, select **Cancel**. The Service Master will still exist but will not be able to execute because its map no longer exists in the database.

Copying or Cutting Maps and Pasting into Another Database

Maps can also be copied and pasted from one database into another.

To cut or copy a map:

1. Highlight the map.
2. Select **Edit** > **Cut** or **Edit** > **Copy**, or right-click and select **Cut** or **Copy** from the shortcut menu.
3. To paste the map into another database, highlight the Map node under that database.
4. Select **Edit** > **Paste**, or right-click and select **Paste** from the shortcut menu.

Note: This pastes all versions of the map into the second database.

To paste Map versions:

A single version of a map can be copied or cut and pasted into another database. To do this, there must already be a deployed map in the second database with the same name as the map from which the single version is cut or copied.

1. Highlight the version node of the map.
2. Select **Edit** > **Cut** or **Edit** > **Copy**, or right-click and select **Cut** or **Copy** from the shortcut menu.
3. Highlight the map node in the second database that has the same name as the map node in the first database.
4. Select **Edit** > **Paste**, or right-click and select **Paste** from the shortcut menu.

Note: This pastes the single version of the map into the second database.

Deployed Custom EAServiceBeans

Viewing Deployed Custom EAServiceBeans

To view custom EAServiceBeans that have been deployed to the Resource Database, expand the Service Beans node. The full class name of all custom EAServiceBeans deployed to the database are listed under this node. Each service bean can be expanded to display the versions of the service bean that have been deployed.

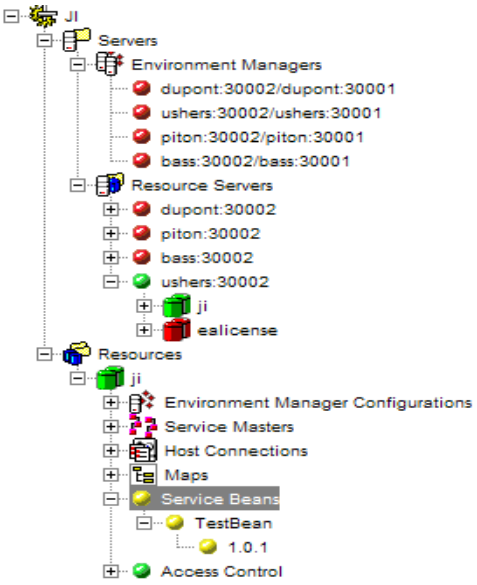


Figure 234. Viewing Deployed Custom EAServiceBeans

Deleting Custom EAServiceBeans

There are two ways to delete service beans that have been deployed to the Resource Database:

- Deleting all versions of the Service Bean.
- Deleting only one version of the Service Bean.

To delete all versions of the Service Bean:

1. Highlight the service bean in the Tree view, right-click, and select **Delete** from the shortcut menu.

   A message indicates that all versions of the service bean will be deleted.
2. Click **OK**.

> **Note:** If there are any Service Masters that use this service bean, a message appears warning of this fact. To proceed with deleting the service bean, click **OK**, otherwise, click **Cancel**. The Service Master will still exist but will not be able to execute because its service bean no longer exists in the database.

To delete only one version of the Service Bean:

Highlight the version node subordinate to the map, right-click, and select **Delete**.

> **Note:** If there are any Service Masters that use this version of the map, a message appears warning you of this fact. To proceed with deleting the map, click **OK**, otherwise, click **Cancel**. The Service Master will still exist but will not be able to execute because its map no longer exists in the database.

Copying or Cutting Service Beans and Pasting into Another Database

Service beans can also be copied and pasted from one database into another.

To cut or copy and paste a service bean:

1. Highlight the service bean.
2. Select **Edit** > **Cut** or **Edit** > **Copy**, or right-click and select **Cut** or **Copy**.
3. Highlight the Service Beans node under that database.
4. Select **Edit** > **Paste**, or right-click and select **Paste** from the shortcut menu.

> **Note:** This pastes all versions of the service bean into the second database.

To paste Service Bean versions:

You can also cut or copy a single version of a service bean and paste the version into another database. To do this, there must already be a deployed service bean in the second database with the same name as the service bean from which the single version is cut or copied.

1. Highlight the version node for the service bean.
2. Select **Edit** > **Cut** or **Edit** > **Copy**, or right-click and select **Cut** or **Copy** from the shortcut menu.
3. Highlight the service bean node in the second database that has the same name as the service bean node in the first database.
4. Select **Edit** > **Paste**, or right-click and select **Paste** from the shortcut menu.

Note: This pastes the single version of the service bean into the second database.

Licensing

This sections describes:

- "The License Key" on page 504
- "License Properties" on page 504
- "The License File (license.txt)" on page 506
- "Adding and Editing a License Key" on page 507
- "Checking the Status of webMethods JI Licenses" on page 510
- "License Expiration Email Notification" on page 511
- "Duplicate Licenses" on page 513
- "License Compatibility" on page 514

The License Key

The webMethods JI license key contains the following information:

1. License Name
2. Major Version
3. Minor Version
4. Expiration Date and time
5. One to five licenses - each license contains a combination of webMethods JI features required by a specific customer. The available features include JI_SERVER, JI_SERVICE, JI_CLIENT, JI_HOST and JI_MAPMAKER. The empty slots are indicated by the value NO_LICENSE. For information on the various license features, see "License Properties" on page 504.

License Properties

Each of the licenses included in the license key contains the following information:

- "Feature Type" on page 505
- "License Count" on page 506
- "Expire Flag" on page 506

Feature Type

This property determines the maximum number of instances that a specific type of feature may have across *all* active webMethods JI runtimes. The license feature type can have one of the following values:

- JI_SERVER - determines the maximum number of servers on a network, permitted to run a webMethods JI Environment Manager.
- JI_SERVICE - determines the maximum number of concurrent Services.
- JI_CLIENT - determines the maximum number of concurrent run-time clients that are external to the webMethods JI Server and are connected to active services.

Note: Although JClient3 method steps are webMethods JI Clients, they are not counted because they are "internal" clients within the webMethods JI Server.

- JI_HOST - determines the maximum number of concurrent host sessions.

Note: JI_HOST licenses are not used by Compatibility Kit customers.

- JI_MAPMAKER - allows an unlimited number of MapMaker GUIs to run simultaneously.

In addition to these five standard features, two more feature types may be used:

- NO_LICENSE - indicates that this license key slot has not been filled.
- JI_DEMO - allows you to get acquainted with webMethods JI through a composite of three features: JI_SERVER (one license), JI_MAPMAKER (unlimited licenses) and JI_SERVICE (as many licenses as there are JI_DEMO licenses, which range from one to five).

Note: The number of licenses allowed by JI_DEMO is overridden by the number of licenses allowed by non-demo features.

License Priority

JI_SERVICE, JI_CLIENT, and JI_HOST are not used concurrently. Instead, the license checking code searches for one valid license in the following order:

1. JI_SERVICE
2. JI_CLIENT
3. JI_HOST

Once the code finds the license with the highest priority available, it stops searching for licenses that have a lower priority. The code uses the information in the license that takes precedence, and automatically sets the other two licenses to "unlimited".

For example, if JI_SERVICE is not defined and JI_CLIENT and JI_HOST are both defined, JI_CLIENT has the highest priority available. Accordingly, the JI_CLIENT information is used while JI_SERVICE and JI_HOST are set to "unlimited".

License Count

The license count determines how many instances of this feature may be used concurrently. This number is either a numeric value, or the string "unlimited".

Note: webMethods JI licenses are not counted per Server instance, but across all active webMethods JI runtimes. For example, if you are running two webMethods JI Servers and your license is for 100 Services, you can only run 100 total Services between the two Servers (and not 100 Services per-Server).

Expire Flag

webMethods JI licenses are renewed once a year. The expire flag indicates whether or not the license should honor the expiration date.

The License File (license.txt)

The webMethods JI licensing information is located in a special license file. The file's default name is *license.txt* and its default location is *<JI_install_dir>/config*.

The *license.txt* file is created the first time a license is entered during the installation or via the Configuration Manager. The file is served by a Resource Server and is edited using the Configuration Manager. The configuration manager always shows the same license keys as the *license.txt* file.

Note: Do not edit the *license.txt* file using any tool other than the Configuration Manager.

To edit or add license keys within the license file, you must first connect to the Resource Server (see "Locating Resource Servers" on page 470).

Note: Licensing information also appears in the envmgr.cfg configuration file, which specifies the name of the license entered during the webMethods JI installation. If several licenses have been added using the Configuration Manager, the license name specified in envmgr.cfg determines which license is used by that Environment Manager at runtime.

Adding and Editing a License Key

After connecting to the Resource Server, a default license resource named `ealicense` is listed as subordinate to that Resource Server (Figure 235).

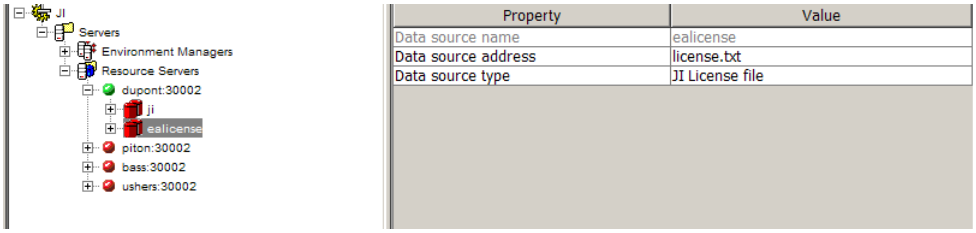


Figure 235. Default License Resource > ealicense

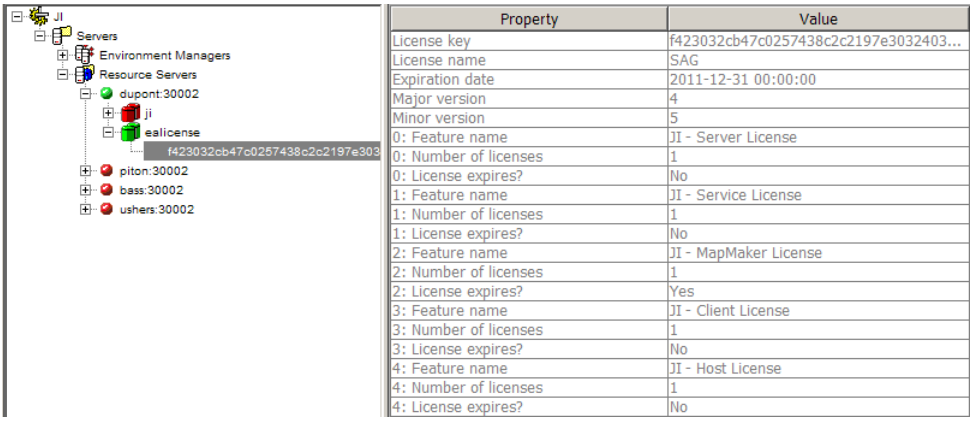
The Properties view displays the values of the following license resource properties:

Property	Description
Data source name	The name of the resource. This value cannot be edited.
Data source address	The location of the license file: *<directory>/<filename>.txt.* Examples: • UNIX: */opt/JI/config/license.txt.* • Windows: *C:\JI\config\license.txt.* **Note:** If the license file is in its default location, *<JI_install_dir>/config*, you do not need to enter the path to the file.
Data source type	The type of data source identified by this resource.

To view the license properties, right click the license node in the Tree view and select **Open datasource** from the shortcut menu.

The license key nodes are displayed in the Tree view. Selecting one of the license nodes displays its properties in the Properties view (Figure 236).

> **Note:** Open license data source nodes have a green icon; those that are closed have a red icon.



Property	Value
License key	f423032cb47c0257438c2c2197e3032403...
License name	SAG
Expiration date	2011-12-31 00:00:00
Major version	4
Minor version	5
0: Feature name	JI - Server License
0: Number of licenses	1
0: License expires?	No
1: Feature name	JI - Service License
1: Number of licenses	1
1: License expires?	No
2: Feature name	JI - MapMaker License
2: Number of licenses	1
2: License expires?	Yes
3: Feature name	JI - Client License
3: Number of licenses	1
3: License expires?	No
4: Feature name	JI - Host License
4: Number of licenses	1
4: License expires?	No

Figure 236. License Key Node Properties

If the license name is something other than the default name (`SoftwareAG`) or `EmergencyTemporaryKey`, it is necessary to change the license parameter for the Environment Manager to match the license name entered here.

> **Note:** A license name is not configurable, as it is associated with the Software AG-provided license key.

If the Environment Manager is not running, the Configuration Manager parameters for the Environment Manager are not editable. The value of the License name parameter needs to be changed directly in the Environment Manager's configuration file. For information about editing this file, see Chapter 12 - "Configuration Manager" on page 455.

If the license key is invalid, the Configuration Manager displays an error message and the license key is rejected.

Adding an Additional License Key Node

To add an additional License Key node, proceed as follows:

1 Right-click the license node in the Tree view, and select **New License Key Node** from the shortcut menu.

The **Enter Key/Name for license** dialog box (Figure 237) opens.

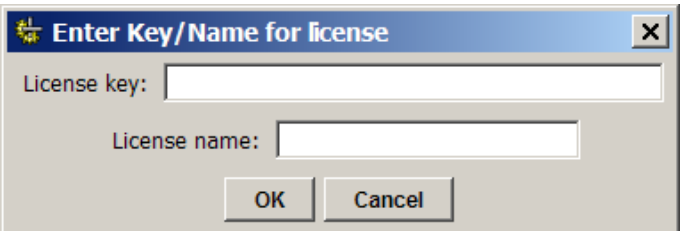


Figure 237. The Enter Key/Name for license dialog box

2 Enter the unique **License Key** and **License Name** that were provided with the installation or by a webMethods JI sales or support representative.

3 Click the **OK** button. The key's properties are listed in the Properties view (Figure 236). These properties are not editable.

Adding an Additional License Data Source

To add a license datasource to a Resource Server, proceed as follows:

1 Right-click on the Resource Server node and select **New Data Source** from the shortcut menu

2 Enter a name for the datasource (for example, `licenseNew`).

3 In the Properties view, enter the following license details:

- Specify the name of the license in the **Data Source Name** field.
- Specify the license location in the **Data source address** field.
- Select **JI License file** as the license type from the **Data source type** drop-down list.

4 Right-click and select **Save** from the shortcut menu, or select **Save** from the **File** menu.

5 To add license keys, see "Adding and Editing a License Key" on page 507.

Note: If a license data source is added, the Resource Server must be restarted before the change takes effect.

Checking the Status of webMethods JI Licenses

The following table describes useful commands for checking license statuses.

Command	Function	Comment
ea_start	Checks for expired licenses	When the `ea_start` command starts the Resource Server, it also checks for licenses that have either expired or are about to expire in 30 days or less (and specifies the number of days or hours left). **Note:** ea_start always shows the expiration status of the JI_DEMO license (e.g. "WARNING: Feature JI_DEMO, name PHOTON expires in 149 days.").
ea_admin status	Counts service, host and client licenses	• When checking for available licenses, make sure that no webMethods JI services have been pre-started. To identify the names of such pre-started services, run the command `ea_admin jclusters -e`. • For `ea_admin status` to display correct information after deleting or adding licenses using the configuration manager, you must restart the environment manager.

ea_admin licenseStatus	Lists all licenses in the Resource Server.	The licenses are listed in the following format:
		`<feature Name><License Name>`
		`<Expiration><Count>`
		
		• Expiration: the number of hours until expiration, or "N" if the license never expires.
		• Count: the number of licenses that can run simultaneously, or "unlimited" if there is no limit.

Command	Function	Comment
ea_admin jclusters	Describes all JClusters and the services they contain.	

License Expiration Email Notification

The Resource Server can be configured to send a daily email, listing licenses that are about to expire in 30 days or less. This section describes the email notification's format, and the email properties you are required to define in the Resource Server Configuration File (*ressvr.cfg*).

Email Format

This license expiration email notification appears in the following format:

Email Field	Contents
Subject	"webMethods JI License Expiration"

From Address	The email sender's address can be one of the following:
	• If the **config.fromUser** value is defined in the *ressvr.cfg* file, it appears in the **From** field.
	• If the **config.fromUser** value is not defined, <user_name>@<IP_Address> is used instead, where:
	  • <user_name> is the name of the user that started the Resource Server.
	  • <IP_Address> is the numerical IP address of the machine on which the Resource Server is running.

From Name	"Resource Server"

Email Field	Contents

Body	The email body consists of the following elements:
	• "Resource Server Location: <location>", where <location> is the host and port of the resource server in a <host>:<port> format.
	• "Resource Server Version: <version>".
	• The following information is provided for each license that expires in 30 days or less:
	  • "Feature <feature name>, name <license name> expires in N days", *Or*
	  • "Feature <feature name>, name <license name> is expired".
	• The following information is provided for each license that has already expired:
	  • If <feature name> is JI_DEMO, the text "Your webMethods JI server will stop functioning" is appended.
	  • If <feature name> is JI_MAPMAKER, the text "MapMaker will switch into Read Only mode. This will not affect your webMethods JI server." is appended.
	• The license key itself is also appended to the message, along with a breakdown of the key's contents.

To Address, To Name	The email recipient's address and name are configured in *ressvr.cfg* using the following parameters: • Recipient address: **config.mailHost** • Recipient name: **config.mailUser**. **Note:** To enable email notification, both **config.mailHost** and **config.mailUser** must be defined

Resource Server Configuration File (ressvr.cfg) Email Definitions

To following license expiration email properties must be defined in the Resource Server Configuration File (*ressvr.cfg*):

Parameter	Description
config.mailHost	The host name or IP address of the machine to which email should be sent. If the mail server is not listening on the default SMTP port (25), the format "<host>:<port>" may be used. **Note:** The mail host may not be the same as the company name.
config.mailUser	The email address (for example john_smith@mycompany.com). This value may contain multiple destination addresses, separated by commas.
config.fromUser	The email sender's address (for example john_smith@mycompany.com). If this value is not configured, a default value of <user_name>@<IP_Address> is used, where • <user_name> is the name of the user that started the Resource Server. • <IP_Address> is the numerical IP address of the machine on which the Resource Server is running.
config.licenseCheckTime	The hour at which the Resource Server checks for impending license expirations. The valid values are 0 through 23 (0 = midnight, 23 = 11:00 PM). **Note:** This parameter is optional. If it is not defined, a default value of 7 (7:00 AM) is used.

Duplicate Licenses

The exact same license, with the same name and key, cannot be added more than once into the Configuration Manager. However, the Configuration Manager allows you to add multiple licenses that share the same name as long as they do not have an identical key. Such licenses are known as duplicate licenses.

If duplicate licenses have duplicate features (for example, two licenses with the same name both have a JI_SERVER feature), the Resource Server uses the first feature listed in the *license.txt* file and deletes all duplicate features following it.

Note: If a license displayed in the Configuration Manager has no features, it means that all of its features were duplicates deleted by the Resource Server.

The Configuration Manager always adds new licenses at the beginning of the list, so that if there are duplicate features the most updated license takes precedence.

License Compatibility

webMethods JI versions 4.X and 3.X can run on the same subnet, with the following limitations:

1. A 4.X Resource Server and a 4.X Configuration Manager utility cannot use 3.X licenses.
2. A 3.X Resource Server and a 3.X Configuration Manager utility cannot use 4.X licenses.
3. A 4.X Resource Server cannot serve licenses to 3.X environments.
4. A 3.X Resource Server cannot serve licenses to 4.X environments.

Chapter 13. System Monitor

The System Monitor is a Java-based application included with webMethods JI that is used to monitor and administrate server components in the webMethods JI environment. The following is a list of the monitored information that the System Monitor provides:

- **Environment Manager Groups**: Lists all Environment Manager groups running in your local subnet.
- **Environment Managers**: Lists all Environment Managers running within each group. Also lists the number of currently running JClusters, total number of JServices and clients, and other information.
- **JClusters**: Lists all running JClusters controlled by each Environment Manager, number of services currently running, maximum services, and other information.
- **JServices**: Lists the start time, status, client information, trace level, and other information associated with JServices.

The System Monitor also allows events to be monitored; such as the starting and stopping of JClusters and JServices. Actions like stopping the environment, starting and stopping JClusters, and other actions associated with the environment can be performed with the System Monitor.

Starting the System Monitor

The System Monitor can be started from the command line in both UNIX and Windows. Under Windows, the System Monitor can be started using shortcuts, if shortcuts were created during webMethods JI installation.

For example, if shortcuts were added to the **Start** menu during installation, the System Monitor can be started by selecting this shortcut from the appropriate program menu in the **Start** menu.

Command Line

Starting the System Monitor from the command line can be accomplished by executing the following command:

UNIX:

```
cd <JI_install_dir>/bin
./ea_sysmon
```

Windows:

```
cd <JI_install_dir>\bin
.\ea_sysmon
```

> **Note:** This executable uses a LAX file to specify the runtime behavior of the application. If certain changes occur to your system, (for example if you want to use a Java Runtime Environment [JRE] different from the one selected during webMethods JI installation) it may be necessary to edit the ea_sysmon.lax file. For information about editing LAX files, see Chapter 7 - "LAX Files" on page 177 of the Supplemental Reference Guide.

The System Monitor Interface

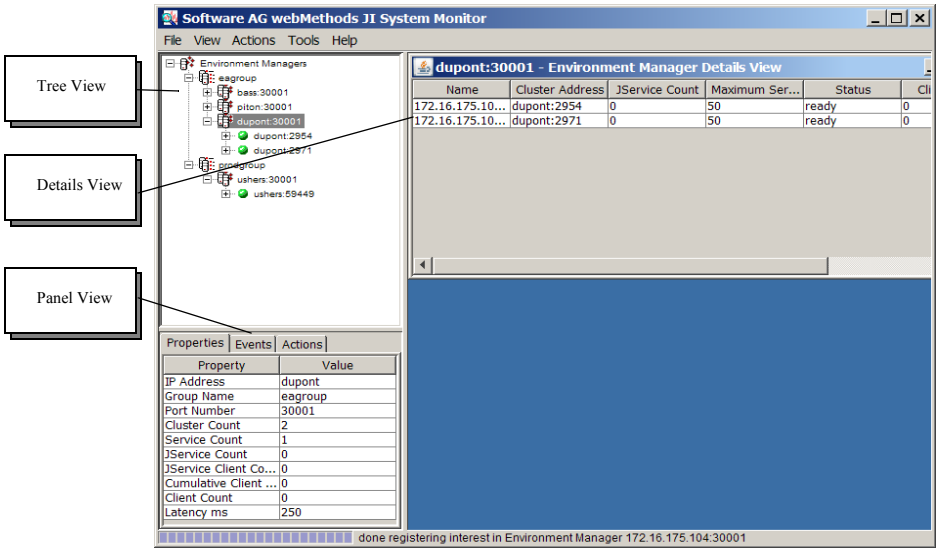


Figure 238. The System Monitor Interface

System Monitor Views

The System Monitor interface consists of three primary panes, or views:

- “Tree View” on page 517
- “Panel View” on page 517
- “Details View” on page 518

Tree View

The upper-left pane of the System Monitor interface is the Tree view. The Tree view is a graphical representation of hierarchical information, consisting of "root" components that can be expanded to expose underlying sub-components (also called "nodes," or "branches"). In many instances sub-components can be expanded as well. This reveals their own sub-components, creating a hierarchical tree format.

A "+/-" symbol before each component in the tree identifies whether or not the node can be expanded to reveal additional nodes. A "+" indicates that the node can be expanded, a "-" indicates that the node is already expanded. The absence of either a "+" or "-" indicates that the node cannot be expanded.

Note: When using the Motif look and feel, a root node handle is used in place of the "+/-" symbols.

Panel View

The Panel view, located in the lower left portion of the System Monitor interface, contains three views, represented by tabs at the top of the view. The three tabbed views are:

- **Properties**: Displays the properties for the component highlighted in the Tree view. Individual component properties and their descriptions are discussed later in this chapter.
- **Events**: Allows you to select the events that will be logged to the **Event Log** window in the Details view. Only events associated with the component highlighted in the Tree view are available in the **Events** tab. These events can only be selected if an **Event Log** window is open and active in the Details view.

  To select events, open an **Event Log** window by selecting **View** > **New Event Log**. With the **Event Log** window active in the Details view, click on the event listed in the **Events** tab to be logged. A check mark appears next to the selected event. Repeat for all appropriate events. When a selected event occurs, information about that event is logged to the **Event Log** window.

  Individual components and their available events are discussed later in this chapter.
- **Actions**: Provides access to the actions associated with the component that is highlighted in the Tree view. To perform an action, click on the action in the **Actions** tab. Individual components and their available actions are discussed later in this chapter.

Details View

The right pane of the System Monitor interface is the Details view. Multiple windows can be displayed in the Details view; only one window is active at any time.

The Details view is used to display four types of windows:

- **Event Log**: Displays the events associated with the component that is selected in the Tree view. The events that are displayed are selected in the **Events** tab of the Panel view.

> **Note:** Multiple events can be monitored at the same time.

- **Details View**: Displays detailed information about the component selected in the Tree view.

> **Note:** The Details view is not available for JServices. When one of these nodes is selected in the Tree view and the Details view is active, the Details view is blank.

- **Clients View:** Displays a list of clients and the associated information for the component that is selected in the Tree view.

> **Note:** The Clients view is not available for JServices or Environment Manager groups. When one of these nodes is selected in the Tree view and the Clients view is active, the Clients view is blank.

- **Properties view:** Displays the properties for the component that is selected in the Tree view. The information displayed in this window is the same as the information displayed in the **Properties** tab of the Panel view.

> **Note:** The Properties view is not available for Environment Manager groups. When one of these nodes is selected in the Tree view and the Properties view is active, the Properties view is blank.

To view Events, Details, Client, or Properties associated with an item selected in the Tree view, open a window by selecting the appropriate option from the **View** menu. The window remains open even after a different item is selected in the Tree view. This allows you to monitor multiple components of the environment using a single open window, though you may have multiple windows open to monitor multiple events as well.

For example, you can monitor events for multiple JClusters using a single **Event Log** window by following these steps:

1. Select a JCluster in the Tree view.
2. Select **View** > **New Event Log** to open an Event Log window.
3. Select the **Events** tab in the Panel view.
4. Select the events you wish to monitor and click **Apply**.
5. Select another JCluster in the Tree view. It can be another JCluster within the same Environment Manager or from a different Environment Manager. In the **Events** tab of the Panel view, select the events to monitor for this JCluster and click **Apply**.

> **Note:** Unlike the Details, Client, or Properties windows, the Event Log window displays an aggregate view of all selected events for the JClusters, JServices, etc. selected in the tree view.

At this point, a single **Events Log** window displays event logs for the selected JClusters. You might choose to open another **Events Log** window in case you needed to segregate the events of a single JCluster, JService or Environment Manager.

System Monitor Menus

The System Monitor menu bar contains the following menus:

- "File Menu" on page 520
- "View Menu" on page 520
- "Actions Menu" on page 521
- "Tools Menu" on page 522
- "Help Menu" on page 523

Shortcut menus are also available throughout the System Monitor and are accessed by selecting a component in the Tree or Details view and right-clicking on it.

The following sections detail the options available in each menu.

File Menu

Figure 239. The File Menu

To access the **File** menu using the mouse, click **File** in the menu bar. The **File** menu offers the following options:

Option	Command
Properties	Sets properties for the System Monitor.
Exit	Closes the System Monitor.

View Menu

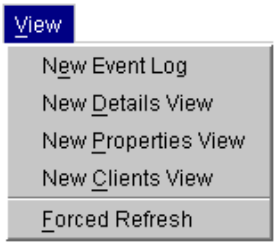


Figure 240. The View Menu

To access the **View** menu using the mouse, click View on the menu bar. The **View** menu offers the following options:

Option	Command
New Event Log	Opens a new **Event Log** window displaying information about events selected in the **Events** tab of the Panel view.
New Details View	Opens a new **Details View** window displaying information about the component selected in the Tree view.

Option	Command
New Properties View	Opens a new **Properties View** window displaying information about the component selected in the Tree view.
New Clients View	Opens a new **Clients View** window displaying information about the clients associated with the Environment Manager or JCluster selected in the Tree view.
Forced Refresh	Refreshes the information displayed in the System Monitor. For example, a forced refresh instructs the System Monitor to locate any new Environment Managers that started in the local sub-net since the System Monitor has been running.

Actions Menu

Figure 241. The Actions Menu

To access the **Actions** menu using the mouse, click **Actions** on the menu bar. The **Actions** menu offers the following options:

Option	Command
Environment	**Kill Environment:** Kills the selected Environment Manager. **Reload Configuration:** Reloads the configuration file for the selected Environment Manager. **Spawn JCluster:** Starts a new JCluster for the selected Environment Manager.

Option	Command
JClusters	**Kill JCluster:** Kills the selected JCluster **Kill JCluster Later:** The JCluster continues to process clients that are currently connected through the JCluster until either all clients are disconnected or the JCluster's rollover timeout value is exceeded. **Note:** The JCluster's rollover timeout value is set in the Configuration Manager (see "Environment Manager and JCluster Configuration" on page 479). The JCluster does not allow any more client connections after this point.
JServices	**Disconnect Client:** Disconnects the client from the JService. **Kill JService:** Kills the JService.

Note: If any of the **Kill** or **Disconnect** options are selected, a pop-up window will appear to verify the action.

Tools Menu

Figure 242. The Tools Menu

To access the **Tools** menu using the mouse, click **Tools** on the menu bar. The **Tools** menu offers the following options:

Option	Command
Print	**Selected View-Span:** This option prints the currently selected Properties, Details or Clients view. The JTables are printed in a multiple page format. **Selected View-Scale:** This option prints the currently selected Properties, Details or Clients view. The JTables are scaled to fit on a single page.

Note: Selecting either of the above print options opens a **Print** dialog box. Enter the correct parameters and select the desired print option.

Help Menu

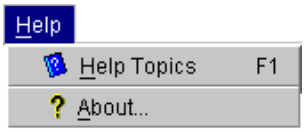


Figure 243. The Help Menu

To access the **Help** menu using the mouse, click **Help** on the menu bar. The **Help** menu offers the following options:

Option	Command
Help Topics (F1)	Opens a window containing help for the System Monitor.
About	Opens a window listing the version of the System Monitor and other information.

Shortcut Menus

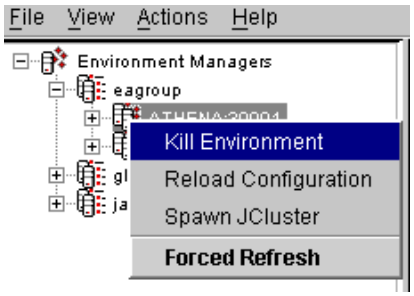


Figure 244. A Shortcut Menu

Shortcut menus are available in many places throughout the System Monitor user interface. To open a shortcut menu, select and right-click on a component in the Tree view or Details view.

Shortcut menus are context sensitive; they present only those options that apply in the current situation. As a result, shortcut menus are not always available, and the options that they present vary from situation to situation.

System Monitor Properties

To set the properties for the System Monitor, select **File** > **Properties**. The **Properties** dialog box opens.

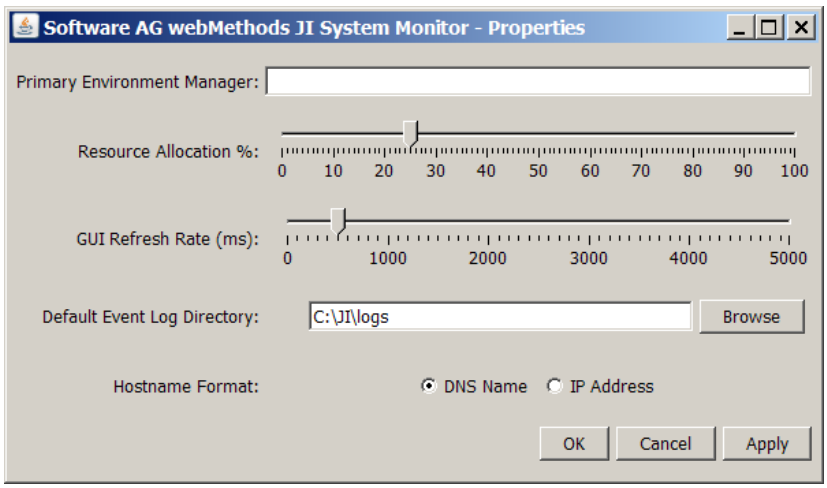


Figure 245. System Monitor Properties

The **Properties** dialog box offers the following options:

Option	Description
Primary Environment Manager	Allows the System Monitor to monitor webMethods JI Environment Managers on a different sub-net than the one it is running on. Enter the hostname and RMI port number of any Environment Manager in the sub-net that is running webMethods JI, and all Environment Managers in that subnet are monitored. By default (when the Primary Environment Manager Property is blank) the System Monitor monitors environments running on the same sub-net it is running on. As environments on the local sub-net are started/stopped, the System Monitor automatically updates the Tree view to show which environments are currently running. If the Primary Environment Manager property is set to an Environment Manager running on a different sub-net from the one the System Monitor is running on, then the System Monitor shows all environments running on that subnet. Due to the nature of multicasting used within webMethods JI, the System Monitor does not dynamically update the Tree view when an environment is started/stopped on the subnet it is not running on.
Resource Allocation	The percentage of resource load that the System Monitor uses for communication between the Environment Managers and the System Monitor. This setting effects both the Environment Manager and System Monitor machines. For more information, see the Latency ms parameter for Environment Variables, on page 529.
GUI Refresh Rate	This adjustment determines how often the GUI updates the JTables. As more events and property changes occur at the same time, it may be necessary to increase this setting to accommodate them. **Note:** This value is in milliseconds (ms).

Option	Description
Default Event Log Directory	The browse button can be used to select a directory other than the default one. The default location is determined by what is set in the Configuration file. **Note:** If this is not set in the Configuration file, the default is determined by the current working directory/logs. If the current directory/logs does not exist, the default location is determined solely by the current directory.
Hostname Format	Offers one of the following two formats for displaying Hostnames: • IP Address • DNS Name.

Monitoring Your webMethods JI Environment

webMethods JI includes a monitoring function that allows the environment to be monitored on a number of levels.

The System Monitor allows the user to "drill down" from high level monitoring to successively lower levels. This allows the entire webMethods JI environment to be monitored.

Examples of the different levels of monitoring include:

- **High level:** Monitoring JClusters and their concurrent service count.
- **Mid level:** A particular JCluster is selected and all services running in that JCluster are monitored.
- **Lower level:** Client information for the JCluster's list of services is monitored.
- **Low level:** A particular service is selected and tracing and/or messages for the service is monitored.

Environment Manager Events and Activities

The System Monitor provides an interface for the following items running in the webMethods JI environment:

- Number of running JClusters
- Environment Manager start/stop events

- Client connection/disconnection events
- Client Request For Service (RFS) events
- Result of RFS requests
- Error conditions

The System Monitor is also used as an interface to Environment Managers for:

- Starting and stopping JCluster processes
- Stopping Environment Managers.

The Environment Manager Tree View

Environment Managers are displayed in the Tree view as nodes associated with the Environment Manager groups to which they belong:

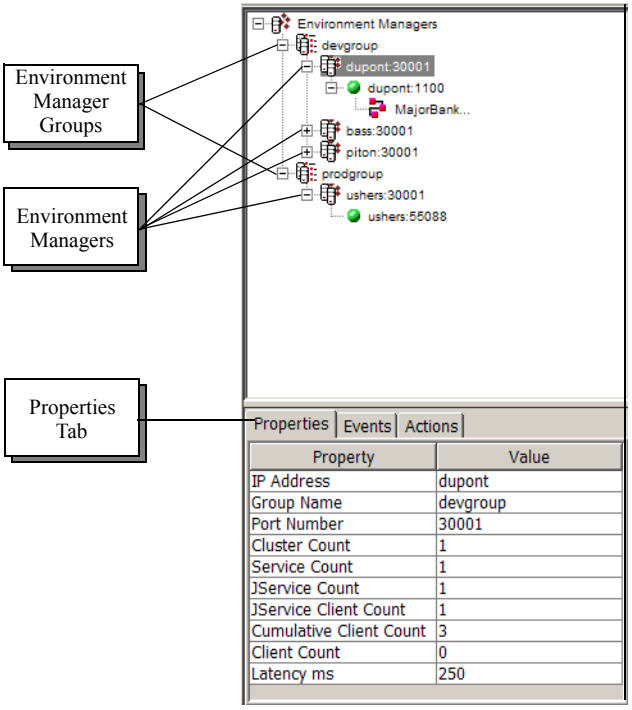


Figure 246. The Environment Manager Tree

Environment Managers are listed by their hostname and port number.

The Environment Manager Panel View

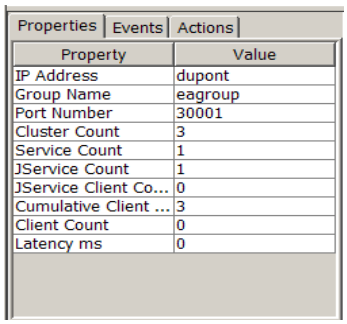
The Panel view includes three tabs:

- "The Environment Manager Properties Tab" on page 528

- "The Environment Manager Events Tab" on page 529
- "The Environment Manager Actions Tab" on page 531

The Environment Manager Properties Tab

To view the properties for an Environment Manager, select the Environment Manager in the Tree view and click on the **Properties** tab in the Panel view.



Properties | Events | Actions

Property	Value
IP Address	dupont
Group Name	eagroup
Port Number	30001
Cluster Count	3
Service Count	1
JService Count	1
JService Client Co...	0
Cumulative Client ...	3
Client Count	0
Latency ms	0

Figure 247. The Environment Manager Properties Tab

The Environment Manager **Properties** tab provides the following information:

Property	Description
IP Address	The host name or IP Address of the selected Environment Manager.
Group Name	The name of the Environment Manager group the Environment Manager belongs to.
Port Number	The port number of the Environment Manager.
Cluster Count	The total number of JClusters currently running for the Environment Manager.
Service Count	The total number of client listener threads for the Environment Manager.
JService Count	The total number of JServices for all JClusters currently running for the Environment Manager.
JService Client Count	The total number of clients currently connected to JServices for all JClusters running for the Environment Manager.

Property	Description
Cumulative Client Count	The cumulative number of clients that have been connected to JServices since the Environment Manager was started.
Client Count	The total number of clients currently connected to the Environment Manager.
Latency ms	This parameter works with the Resource Allocation setting in the System Monitor **Properties** dialog box (see "System Monitor Properties" on page 524). Based on the Resource Allocation setting, the System Monitor and Environment Managers dynamically adjust this setting to determine the amount of time in milliseconds that the Environment Manager waits before sending information to the System Monitor.

The Environment Manager Events Tab

To monitor the events for an Environment Manager, open a new **Events Log** window by selecting **View > New Event Log**. Select the appropriate Environment Manager in the Tree view and click on the **Events** tab in the Panel view. To select events to monitor, click on each appropriate event. A check mark appears next the each selected event.

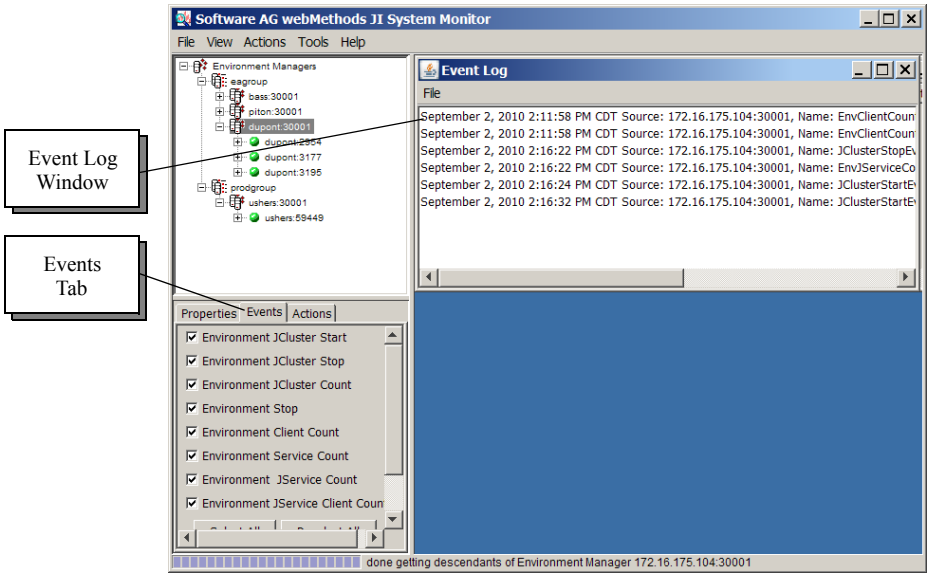


Figure 248. The Environment Manager Events Tab

The Environment Manager **Events** tab provides the following events:

Event	Description
Environment JCluster Start	Creates a log entry each time a JCluster is started by the Environment Manager.
Environment JCluster Stop	Creates a log entry each time a JCluster is stopped by the Environment Manager.
Environment JCluster Count	Creates a log entry each time the Environment Manager counts the number of currently running JClusters.
Environment Stop	Creates a log entry when the Environment Manager is stopped.
Environment Client Count	Creates a log entry each time the Environment Manager counts the number of current client connections.
Environment Service Count	Creates a log entry each time the Environment Manager counts the number of currently running services.
Environment JService Count	Creates a log entry each time the Environment Manager counts the number of currently running JServices.
Environment JService Client Count	Creates a log entry each time the Environment Manager counts the number of current client connections to JServices.

To save the file to your local hard disk, use the **File** menu with the **Event Log** window open. Select **File** > **Save Log to File** and identify the filename and location of the file. Events for multiple components in your webMethods JI environment can be monitored in the same **Event Log** window; select additional components in the Tree view when the **Event Log** window is open, and then select events for each component.

To cancel changes, click the **Cancel** button. To apply the changes, click the **Apply** button.

The Environment Manager Actions Tab

To view the actions available for an Environment Manager, select the Environment Manager in the Tree view and click on the **Actions** tab in the Panel view.

Figure 249. The Environment Manager Actions Tab

> **Note:** These actions are also available in a shortcut menu. Select an Environment Manager, right-click and select the action from the shortcut menu provided.

The Environment Manager Actions tab provides the following options:

Property	Description
Kill Environment	Kills the Environment Manager.
Reload Configuration	Reloads the configuration file for the Environment Manager selected in the Tree view.
Spawn JCluster	Starts a new JCluster for the Environment Manager.

The Environment Manager Detail View

The Environment Manager Detail view allows you to display four different windows to monitor events and to view event properties:

- "The Event Log Window" on page 532
- "The Details View" on page 532
- "The Properties View" on page 533
- "The Clients View" on page 533

The Event Log Window

The **Event Log** window is used to monitor events selected in the **Events** tab of the Panel view. For information about events associated with Environment Managers, see "The Environment Manager Events Tab" on page 529.

The Details View

The Details window displays information about the currently running JClusters for the selected Environment Manager:

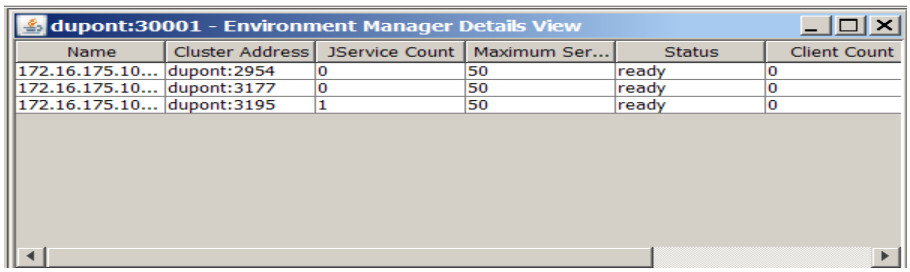


Figure 250. Details View

The following properties are listed in the **Details** window:

Property	Description
Name	The IP Address and port of the JCluster.This information forms the logical name by which the JCluster is internally identified.
Cluster Address	The hostname, or IP Address, and port of the JCluster.
JService Count	The number of services running within the JCluster.
Maximum Services	The maximum number of services that can be run in the JCluster.

Property	Description
Status	The status of the JCluster: • **Starting:** JCluster is launching • **Ready:** Ready to accept client connections. • **Waiting to Exit:** Applies only when the Kill JCluster Later action is selected (). • **Stopping:** JCluster is stopping. • **Dead:** Displayed before JCluster exits. • **Not Accepting Connections:** JCluster is not accepting new connections.
Client Count	The total number of clients currently connected to the JCluster.
Cumulative Client Count	The cumulative number of clients that have been connected to the JCluster.
Latency ms	Displays the number of milliseconds that the JCluster's Environment Manager waits to send information to the System Monitor. For more information, see the Latency ms description under "The Environment Manager Properties Tab" on page 528.

Note: These properties are the same as those displayed in the **Properties** tab of the Panel view for JClusters.

The Properties View

The Properties View displays the same information as displayed in the **Properties** tab of the Panel view. For more information about the properties, see "The Environment Manager Properties Tab" on page 528.

The Clients View

The Client's View displays the clients for the Environment Manager currently selected.

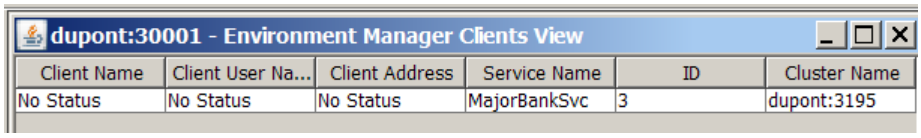


Figure 251. Client View

The following properties are listed in this window:

Property	Description
Client Name	The name of the clients associated with the component selected in the Tree view.
Client User Name	The name of the user that executed the client.
Client Address	The hostname of the client.
Service Name	The name of the client's service.
ID	The ID number of the client's JService.
Cluster Name	The name of the JCluster the client is connected to.

JCluster Events and Activities

The System Monitor provides a monitoring interface to the following items for each JCluster running within the webMethods JI environment:

- Number of running services.
- Types of services and service version information.
- Service start/stop events.
- Client Request For Service (RFS) events.
- JCluster status.
- JCluster start/stop events.
- Error conditions.
- Current load (the number of clients communicating to services through this JCluster).

The System Monitor is also used as an interface to JClusters for killing those processes.

The JCluster Tree View

JClusters are listed in the Tree view, subservient to the Environment Manager to which they belong:

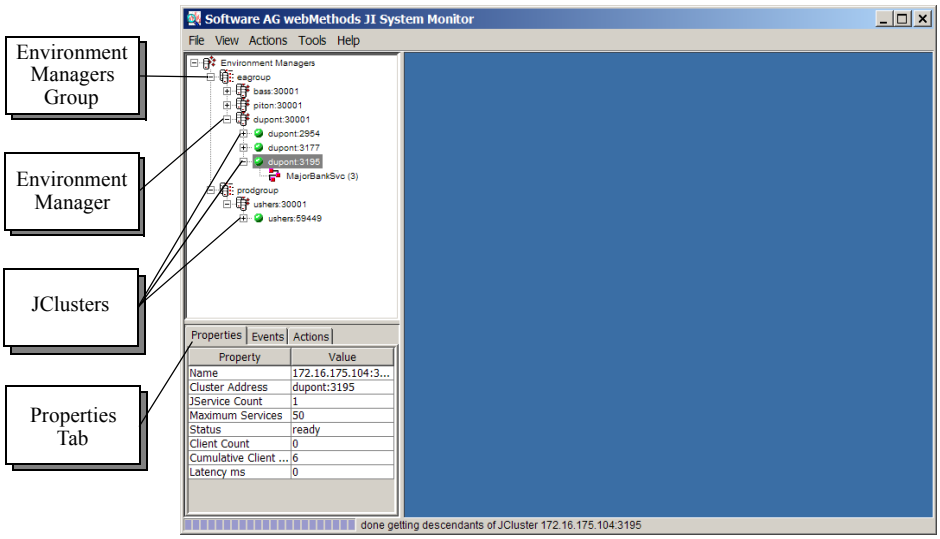


Figure 252. The JCluster Tree View

JClusters are listed by their hostname and port number.

The JCluster Panel View

The Panel view includes three tabs:

- Properties tab
- Events tab
- Actions tab

The JCluster Properties Tab

To view the properties for a JCluster, select it in the Tree view and click on the **Properties** tab in the Panel view.

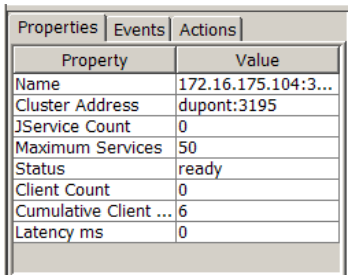


Figure 253. The JCluster Properties Tab

The JCluster **Properties** tab provides the following information:

Property	Description
Name	The IP Address and port of the JCluster.This information forms the logical name the JCluster is internally identified by
Cluster Address	The hostname, or IP Address, and port of the JCluster.
JService Count	The number of JServices running within the JCluster.
Maximum Services	The maximum number of services that can be run in the JCluster.
Status	The status of the JCluster: • **Starting:** JCluster is launching • **Ready:** Ready to accept client connections. • **Waiting to Exit:** Applies only when the Kill JCluster Later action is selected (). • **Stopping:** JCluster is stopping. • **Dead:** Displayed before JCluster exits. • **Not Accepting Connections:** JCluster is not accepting new connections.
Client Count	The total number of clients currently connected to the JCluster.

Property	Description
Cumulative Client Count	The cumulative number of clients that have been connected to the JCluster.
Latency ms	Displays the number of milliseconds that the JCluster's Environment Manager waits to send information to the System Monitor. For more information, see the Latency ms description under "The Environment Manager Properties Tab".

The JCluster Events Tab

To monitor the events for a JCluster, open a new **Events Log** window by selecting **View** > **New Event Log**. Highlight the appropriate JCluster in the Tree view and click on the **Events** tab in the Panel view. To select events to monitor, click on each appropriate event.

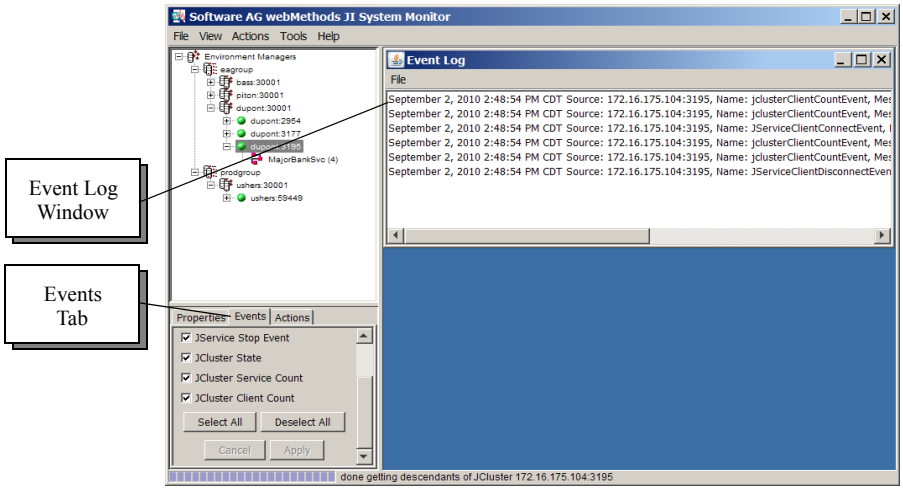


Figure 254. The JCluster Events Tab

The JCluster **Events** tab provides the following events:

Event	Description
JService Client Connect Event	Creates a log entry each time a client connects to a JService that is running in the selected JCluster.

Event	Description
JService Client Disconnect Event	Creates a log entry each time a client disconnects from a JService that is running in the selected JCluster.
JService Start Event	Creates a log entry each time a JService is started in the selected JCluster.
JService Stop Event	Creates a log entry each time a JService running in the selected JCluster is stopped.
JCluster State	Creates a log entry each time the JCluster determines its state.
JCluster Service Count	Creates a log entry each time the JCluster counts the number of JServices it is currently running.
JCluster Client Count	Creates a log entry each time the JCluster counts the number of clients currently connected.

To save the file to your local hard disk, use the **File** menu on the **Event Log** window. Select **File > Save Log to File**. Give the identity of the filename and location of the file.

Events for multiple components in the webMethods JI environment can be monitored in the same **Event Log** window. Select additional components in the Tree view while the **Event Log** window is active. Next, select the events for each component.

To cancel changes to the **Events** tab, click the **Cancel** button. To apply the changes, click the **Apply** button.

The JCluster Actions Tab

To view the actions available for a JCluster, select the JCluster in the Tree view and click on the **Actions** tab in the Panel view.

> **Note:** These actions are also available in a shortcut menu: select the JCluster, right-click, and select the action from the shortcut menu provided.

Figure 255. The JCluster Actions Tab

The JCluster **Actions** tab provides the following actions:

Property	Description
Kill JCluster	Kills the selected JCluster immediately.
Kill JCluster Later	The JCluster continues to process clients that are currently connected through the JCluster until either all clients are disconnected or the JCluster's rollover timeout value is exceeded. **Note:** The JCluster's rollover timeout value is set in the Configuration Manager (see "Environment Manager and JCluster Configuration" on page 479). The JCluster will not connect any more client connections at this point.

The JCluster Detail View

The JCluster **Detail** view allows you to display four different windows to monitor events and to view the properties:

- Event Log
- Details View
- Properties View
- Clients View

The Event Log Window

The **Event Log** window is used to monitor events selected in the **Events** tab of the Panel view. For information about JCluster-associated events, see "The JCluster Events Tab" on page 537.

The Details View

The **Details** window displays information about currently running JServices for the selected JClusters:

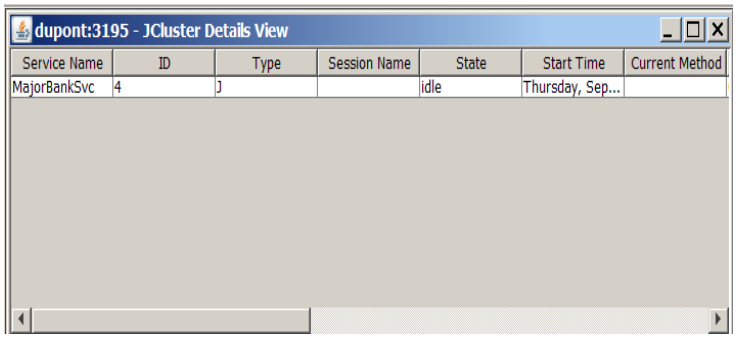


Figure 256. The Details View

The following properties are listed in the **Details** window:

Property	Description
Service Name	The name of the JService as entered in the Service Master entry in the Configuration Manager.
ID	The sequential numeric identifier that the JCluster gives to the JService.
Type	Indicates if the JService is a Java service or a "C" service. `J` = Java service, `C` = "C" service.
Session Name	The session name that the client has given to this JService to enable a re-connection to the service if it still exists.
State	The state of the JService. Values are: • **Idle:** No clients. • **Waiting:** Has client but is not doing anything. • **Busy:** JService is actively serving a client. • **Starting:** The JService is starting. • **Dead:** Displayed before the JService exits.

Property	Description
Start Time	Indicates the time that the JService was started, using the format Month/Day/Year Hour/Minute/Second.
Current Method	The service method that is currently being invoked.
Host Connection Count	The number of current connections between the JService and the host.
Cumulative Client Count	The total cumulative number of clients that have connected to this JService.
Cumulative Method Invoke	The total cumulative number of service method invokes that have been performed by this JService.
Client Name	The name of the most recent client that has connected to the JService.
Client User Name	The name of the user that executed the current client.
Client Address	The IP address of the client currently connected to the JService.
Trace Level	The trace level for the JService. This parameter controls the amount of information that is logged to the JService log file.
Cluster Name	The name of the JCluster.
Host Address	The IP address and port number of the host to which the JService connects.
Method Invoke State	The state of the service method invoke. Values are: • **Starting:** The service method is being invoked. • **Complete:** The service method invoke has completed. • **Exception:** If an exception was thrown, it is displayed here.

The Properties View

The Properties view displays the same information as displayed in the **Properties** tab of the Panel view. For more information about the properties, see "The JCluster Properties Tab" on page 535.

The Clients View

The Clients view displays the clients for the JCluster currently selected.

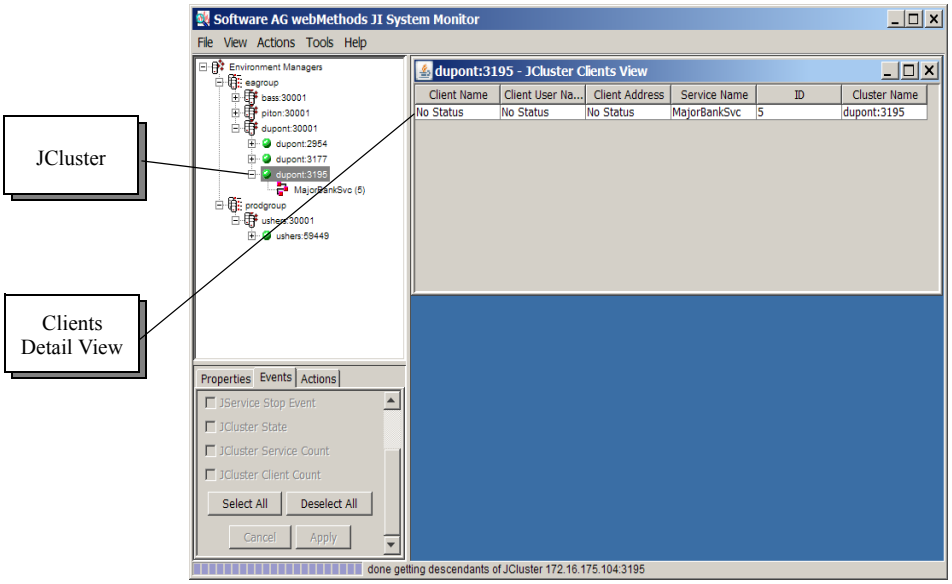


Figure 257. The Clients View

The following properties are listed in this detail window:

Property	Description
Client Name	The name of the clients associated with the component selected in the Tree view.
Client User Name	The name of user that executed the client.
Client Address	The IP address of the client.
Service Name	The name of the client's service.
ID	The ID number of the client's JService.
Cluster Name	The name of the JCluster the client is connected to.

JService Events and Activities

The System Monitor provides an interface to the following items for each JService running in the webMethods JI environment:

- Current state of the JService (idle, connected, etc.)
- JService starting, stopping, and initializing.
- Protocol Agent events.
- Client connection/disconnection events.
- Method invoke events.
- Error conditions.

Dynamically Changing the JService Trace Level

The trace level for the JService can be changed dynamically from within the System Monitor interface.

To change the trace level, double click in the **Trace Level** field on the JService **Properties** tab. This parameter is a editable field that can be changed dynamically so that the amount of JService logging can be changed without having to shut down the JService or the JCluster. Allowed values are `0` to `9`, with `0` indicating there is no logging.

After a new value has been entered, the new trace level takes effect after pressing **Enter**.

The JService Tree View

JServices are listed in the Tree view, subservient to the JCluster to which they belong:

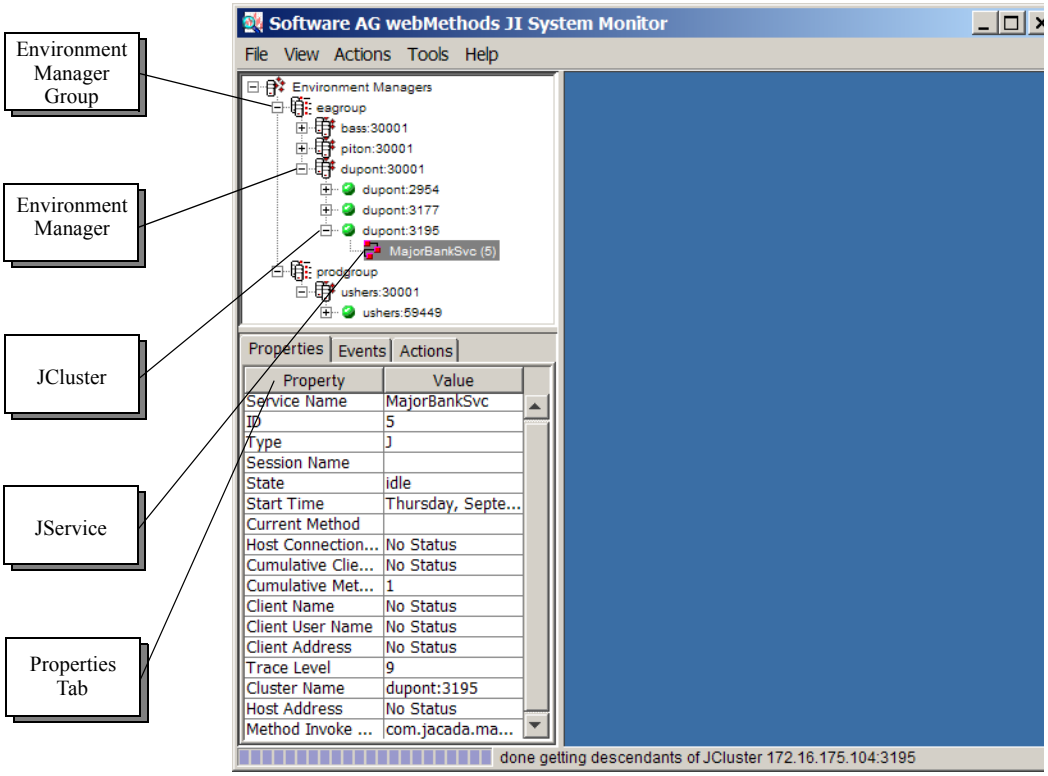


Figure 258. The JService Tree

JServices are listed by their service name.

The JService Panel View

The Panel view includes three tabs:

- "The JService Properties Tab" on page 544
- "The JService Events Tab" on page 547
- "The JService Actions Tab" on page 548

The JService Properties Tab

To view the properties for a JService, select the JService in the Tree view and click on the **Properties** tab in the Panel view.

Properties | Events | Actions

Property	Value
Service Name	MajorBankSvc
ID	5
Type	J
Session Name	
State	idle
Start Time	Thursday, Septe...
Current Method	
Host Connection C...	No Status
Cumulative Client ...	No Status
Cumulative Metho...	1
Client Name	No Status
Client User Name	No Status
Client Address	No Status
Trace Level	9
Cluster Name	dupont:3195
Host Address	No Status
Method Invoke St...	com.jacada.maps...

Figure 259. The JService Properties Tab

The JService Properties tab provides the following information:

Property	Description
Service Name	The name of the JService as entered in the Service Master entry in the Configuration Manager.
ID	The sequential numeric identifier that the JCluster gives to the JService.
Type	Indicates if the JService is a Java service or a "C" service. `J` = Java service, `C` = "C" service.
Session Name	The session name that the client has given to this JService so that the client can reconnect to the service if it still exists.
State	The state of the JService. Values are: • **Idle:** No clients. • **Waiting:** Has client but is not doing anything. • **Busy:** JService is actively serving a client. • **Starting:** The JService is starting. • **Dead:** Displayed before the JService exits.

Property	Description
Start Time	Indicates the time that the JService was started, using the format Month/Day/Year Hour/Minute/Second.
Current Method	The service method that was most recently invoked.
Host Connection Count	The number of current connections between the JService and the host.
Cumulative Client Count	The total cumulative number of clients that have connected to this JService.
Cumulative Method Invoke	The total cumulative number of service method invokes that have been performed by this JService.
Client Name	The name of the client currently connected to the JService.
Client User Name	The name of the user that executed the current client.
Client Address	The IP address of the client currently connected to the JService.
Trace Level	The trace level for the JService. This parameter controls the amount of information that is logged to the JService log file. **Note:** This amount can be edited.
Cluster Name	The name of the JCluster
Host Address	The host name or IP address and port number of the host to which the JService connects. **Note:** For Java services, this value is blank if the map's default host connection is being used.

Property	Description
Method Invoke State	The state of the service method invoke. Values are: • **Starting:** The service method is being invoked. • **Complete:** The service method invoke has completed. • **Exception:** If an exception was thrown, it is displayed here.

The JService Events Tab

To monitor the events for a JService, open a new **Events Log** window by selecting **View** > **New Events Log**. Select the appropriate JService in the Tree view and click on the **Properties** tab in the Panel view. To select events to monitor, click on each appropriate event. A check mark appears next the each selected event.

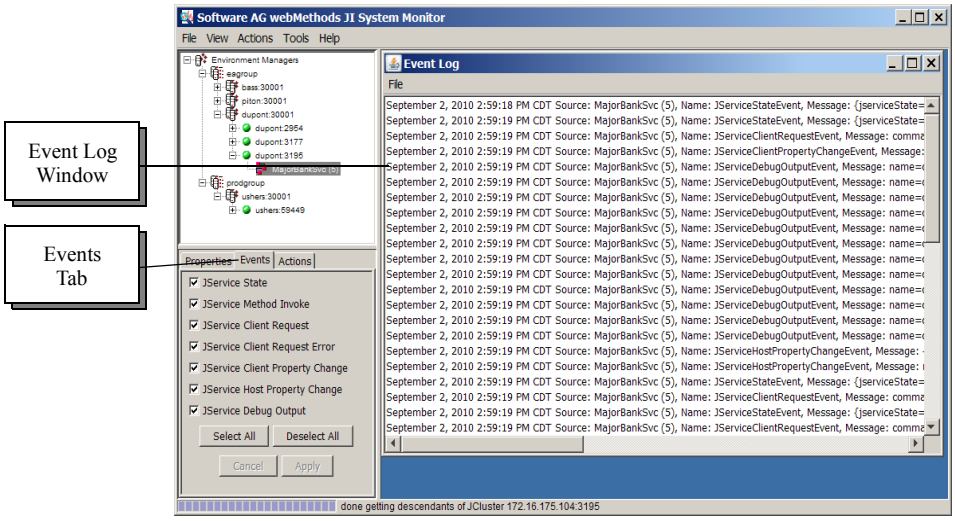


Figure 260. The JService Events Tab

The JService **Events** tab provides the following events:

Event	Description
JService State	Creates a log entry each time the state of the JService is determined.
JService Method Invoke	Creates a log entry each time the JService invokes a service method.

Event	Description
JService Client Request	Creates a log entry each time a client makes a request of the JService.
JService Client Request Error	Creates a log entry each time there is an error generated as a result of a failed client request.
JService Client Property Change	Creates a log entry each time the client properties change.
JService Host Property Change	Creates a log entry each time the host's properties change.
JService Debug Output	Creates log entries for JService debug output.

To save the file to your local hard disk, use the **File** menu in the **Event Log** window. Select **File** > **Save Log to File** and identify the filename and location of the file. Events for multiple components in your webMethods JI environment can be monitored in the same **Event Log** window by selecting additional components in the Tree view when the **Event Log** window is active, and then selecting events for each component.

To cancel changes to the **Events** tab, click the **Cancel** button. To apply the changes, click the **Apply** button.

The JService Actions Tab

To view the actions available for a JService, select the JService in the Tree view and click on the **Actions** tab in the Panel view.

> **Note:** These actions are also available in a shortcut menu. Select a JService, right-click, and select the action from the shortcut menu provided.

Figure 261. The JService Actions Tab

The JService **Actions** tab provides the following actions:

Property	Description
Disconnect Client	Disconnects the client from the JService, but leaves the JService running.
Kill JService	Kills the JService.

The JService Detail View

The JService Detail view allows you to display two different windows to monitor events and to view the properties:

- Event Log
- Properties View

The Event Log

The **Event Log** window is used to monitor the events selected in the **Events** tab of the Panel view. For information about events associated with JServices, see "The JService Events Tab" on page 547.

The Properties View

The Properties view displays the same information as displayed in the **Properties** tab of the Panel view. For more information about the properties, see "The JService Properties Tab" on page 544.

> **Note:** The Details view and Clients view are not available for JServices. If a JService is selected in the Tree view and a Details or Clients window is active, that window is blank.

Chapter 14. Access Control List for webMethods JI

The Access Control List (ACL) functionality in webMethods JI allows control over clients accessing the webMethods JI server and certain webMethods JI functions. The ACL features and functions include the following:

- User ID based authorization. Password authentication to permit only users who supply a valid login and password access to the webMethods JI server, MapMaker, Configuration Manager, and System Monitor.
- There is no practical limit to the number of User IDs and Passwords.
- User IDs and Passwords are created and configured using the Configuration Manager, and are stored in the webMethods JI Resource Database.

Program Resources Controlled by ACL

The following list shows the program resources that are ACL-controlled:

Application	Controlled Resources
MapMaker	Deploy a map to the database
Configuration Manager	Database access
	Modify the Resource Server configuration file
	Modify the Environment Manager configuration file
	Modify the Resource Map file
	Access the license file
System Monitor	Environment Manager actions: - Kill Environment - Reload Configuration

Application	Controlled Resources
	- Spawn JCluster
	JCluster actions: - Kill JCluster - Kill JCluster later
	JService actions: - Disconnect Client - Kill JService
	JService events - The following events are controlled as they may display sensitive information: - JService Client Request - JService Debug Output
	JService properties (i.e. set log level)

Command Line Actions Controlled by ACL

The following list shows the command line actions that are ACL-controlled:

`ea_admin`	Kill JCluster
`ea_shutdown`	Environment shutdown (Environment Manager and Resource Server only; the Tomcat Web Application server and Resource Database server is not controlled by ACL.)

Permissions

When ACL is enabled in webMethods JI, the permissions for the controlled resources and command line functions, are set implicitly to “not permitted”. Permissions are granted through the creation of an ACL Account Record for a specific user or users, and by the creation of an ACL Permission Record for that account.

Permission Records are generally set for specific applications (e.g. MapMaker, System Monitor) or the ACL-controlled command line programs, however a selection for “All” creates the equivalent of a super-user or administrator account.

Creating an account with the selected Application “All” is used to allow access by an ACL account to all resources.That is, the only selectable Resource is “All”. Finally, the only allowable Permissions for the Application “All” are also “All” which enables access to *everything* by the application.

Since this is the highest level of security, creation of such an account should be limited to system administrators, and the dissemination of account information (user name, password) be strictly controlled.

Creating the Initial Administrator Account

It is important to create an Administrator Account with permissions to allow the Configuration Manager to access any resources that require editing, specifically *ea_shutdown*.

If permissions for *ea_shutdown* are not enabled, and ACL is enabled, the webMethods JI environment must be shut down manually by killing the webMethods JI processes directly, the Access Control must be disabled, and the webMethods JI environment re-started to enable editing the resources.

The following steps assume the webMethods JI software is already installed, and otherwise configured. For new webMethods JI installations, ACL functionality is disabled by default.

To create an initial Administrator Account perform the following steps:

1 Start the webMethods JI Environment using the `ea_start` command. Execute the following at your command prompt:

> **Note:** These directions assume that the EA_ENV environment variable is set to the location of your webMethods JI configuration files, and that the correct parameters are set in the environment.ccf file. For more information about ea_start, see the webMethods JI Installation Guide.

UNIX:

```
cd <JI_install_dir>/bin
./ea_start
```

Windows:

```
cd <JI_install_dir>\bin
.\ea_start
```

2 Start the Configuration Manager as described in *"Configuration Manager" on page 455*

3 Expand the Resource Servers node in the Tree view to show the available servers.

4 Select the resource server for which ACL is to be implemented, right click, and select **Connect** to connect to the server.

5 Select one of the available data sources, right click, and select **Open datasource**.

6 Observe the **Resources** node is displayed and a node entitled **Access Control** appears at the bottom of the list for the selected datasource.
Select **Access Control**, right click, and select **New ACL Account record**.

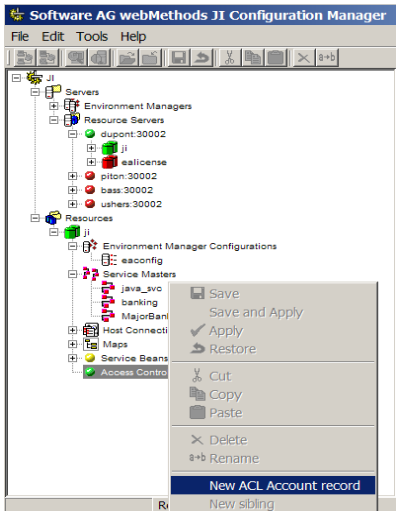


Figure 262. Creating a New ACL Account Record

Figure 263. The New Account is Displayed with the Default Name, Ready for Renaming

7 Observe that a `NewAccount` node is displayed with the default name highlighted (see Figure 263) and ready for re-naming. Rename the account and observe a password reset dialog box prompting for the entry of a new password.

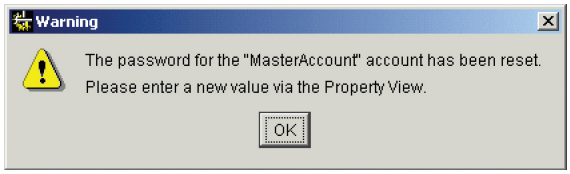


Figure 264. You are Prompted for a New Password

Note: Account names and passwords are case sensitive. When creating or changing an account name or password, make specific note of case usage.

8 Select the password in the Property panel and enter a new password. Right click in the Property panel and click **Save and Apply**. A password verification dialog box opens. Re-enter the password and click **OK**.

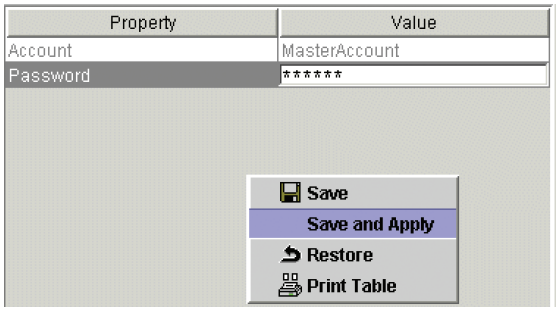


Figure 265. Saving and Applying the Password

After entering the password, the Configuration Manager validates that the password conforms to the following criteria:

- Contains at least six characters.
- Contains at least one numeric character from 0 to 9.
- Contains at least one alphabetic character.

If any of these criteria are not met, the appropriate Password Validation Warning is shown.

Note: These validation checks are only warnings; their password criteria are not enforced. The password is saved as entered, once **OK** is clicked. There is no limit to the number of characters that can be entered for the password, however only the first eight characters are validated.

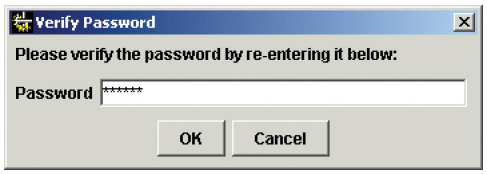


Figure 266. Verify Password message

9 Select the ACL account record that has just been created. From the **Edit** menu, or by right clicking in the Tree view, select **New ACL Permission record**.

   A new permissions record is created, with the selection `All` highlighted in the Tree.

10 Click anywhere in the panel and observe the ACL Permission record `All` has been created, with permissions for Application =“All”, Resource =“All”, and Permissions=“All”.

11 Exit the Configuration Manager.

12 At the command line type in `ea_shutdown` to stop all webMethods JI processes.

Enabling Access Control

Enabling and disabling Access Control is done by editing the Resource Server configuration file, *ressvr.cfg*, located in the *<JI_install_dir>\config* directory. The property `acl.resource` is added; its value is that of the database resource that is configured in the *resource.map* file located in the *<JI_install_dir>\config* directory.

Several database resources may be listed in the `resource.map` file, however, only one database resource may be enabled for ACL use.

1 Use a text editor to open and view the *resource.map* file located in the *<JI_install_dir>\config* directory. Locate the line that specifies the webMethods JI relational database, and make note of it or copy it to the clipboard. The entries are in this form:

```
eardb://<name>=eardb://<location>
```

   The text before the “=” sign is what is needed for the next step.

2 Use a text editor to open the *ressvr.cfg* file located in the *<JI_install_dir>\config* directory. Add the following line to the end of the file:

```
acl.resource=eardb://<name>
```

   Where `eardb://<name>` is the name of the database resource from the *resource.map* file. This is the database source that will be ACL-controlled. All Environment Managers that use this Resource Server and datasource will have ACL enabled.

Note: If the database resource name is changed, that change must be also be made to the ressvr.cfg and resource.map files, located in the <JI_install_dir>\config directory.

3 Save the *ressvr.cfg* file and exit. Close and exit the *resource.map* file as well.

4 Start the webMethods JI Environment using the `ea_start` command. Execute the following at your command prompt:

UNIX:

```
cd <JI_install_dir>/bin
./ea_start
```

Windows:

```
cd <JI_install_dir>\bin
.\ea_start
```

5 Start the Configuration Manager as described in "Configuration Manager" on page 455.

6 Expand the **Resource Servers** node in the Tree view to show the available servers.

7 Select the resource server for which ACL is to be implemented, right click, and select **Connect** to connect to the server. The **Access Controlled Resource Login** dialog box opens.

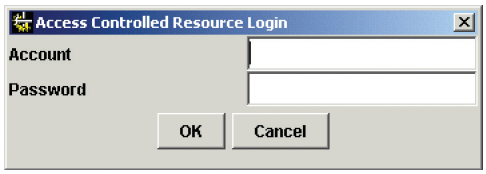


Figure 267. Type the Account Name and Password

> **Note:** Account names and passwords are case sensitive. When creating or changing an account name or password, make specific note of case usage.

8 Type in the **Account** name and **Password** exactly as created, for the Administrator account created earlier. Observe that a connection is made to the Resource Server.

9 From the available data sources, right click, and select **Open datasource**. Observe that the connection is made to the datasource and that a Resources node is displayed with Access Control and the administrator account appearing at the bottom of the list.

Adding New ACL Accounts and Permission Records

To add new ACL account records, permission records, or modify existing accounts, requires Read/Write permissions for the Configuration Manager for Database Access. For the Administrator account, these permissions are conveyed if the account was created for "All" resources, with "All" permissions.

> **Note:** Permissions at this level should be considered for Administrator level personnel only. Permissions granted at this level allow creation of any and all account types, including Administration level accounts.

Adding a New ACL Account Record

To add new ACL Account records perform the following:

1. Login to the Configuration Manager using the Administrator account and password.
2. Select **Access Control** in the **Resources** node. From the **Edit** menu, or by right clicking in the Tree view, select **New ACL Account record**. A new account record titled `NewAccount` is created and the field is highlighted for editing.

Figure 268. A New Account Record Highlighted for Editing

3. Rename the newly created account and observe the password reset warning dialog box. Click **OK**.

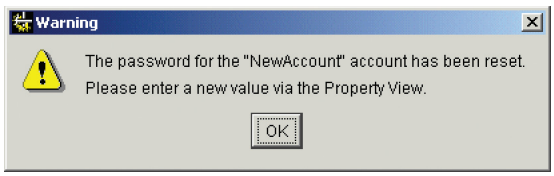


Figure 269. Rename the New Account

4. Click in the Property panel. Select the password and enter a new password. Right click in the Property panel and click **Save and Apply**. The password verification dialog box opens. Re-enter the password and click **OK**.

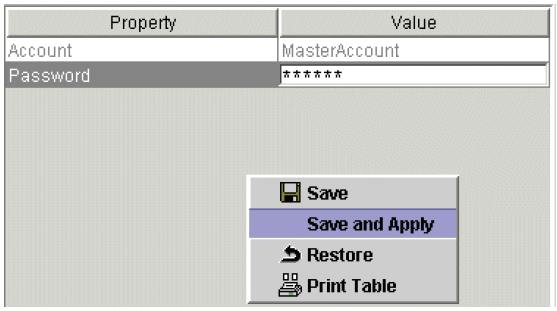


Figure 270. Choose Save and Apply and, when prompted, verify the password.

5. The Configuration Manager validates the password. The password should conform to the following criteria:
   - Contains at least 6 characters

- Contains at least one numeric character from 0 to 9
- Contains at least one alphabetic character

Note: The validation checks are only warnings. There is no enforcement of the password criteria. The password is saved as entered after **OK** is clicked, whether the criteria were observed or not.

6 After the validation checks, the following verification dialog is shown. Re-type the password and click **OK**.

Figure 271. Retype the password and click OK

7 Add new account records as required.

Modifying the ACL Account Record

The ACL account name and password can be modified after their initial creation.

Note: Account names and passwords are case sensitive. When creating or changing an account name or password, make specific note of case usage.

To change the password:

1 Click in the **Password** field in the **Value** column, and highlight the password.
2 Type in the new password, adhering to the password criteria, and click **Save and Apply**.
3 The **Verify Password** dialog box is displayed. Re-enter the new password and click **OK**.

To change the account name:

1 In the Tree view select the account name to be changed. Right click and select **Rename**.
2 Type in the new account name, and click outside the newly typed account name. The password reset warning dialog box opens.

Figure 272. The Password Reset Warning Dialog Box

3 Click in the **Password** field in the **Value** column, and highlight the password.

4 Type in the new password, adhering to the password criteria, and click **Save and Apply**.

5 The **Verify Password** dialog box is displayed. Re-enter the new password and click **OK**.

Note: Once the ACL account name has been changed, the Configuration Manager must be closed and re-started to enable the account.

Adding a New ACL Permission Record

To add new ACL Permission Records perform the following steps.

1 Login to the Configuration Manager using the Administrator account and password.

2 In the Resources node, select an ACL account record that has been created under **Access Control**. From the **Edit** menu, or by right clicking in the Tree view, select **New ACL Permission record**. A new permissions record is created with `All` highlighted.

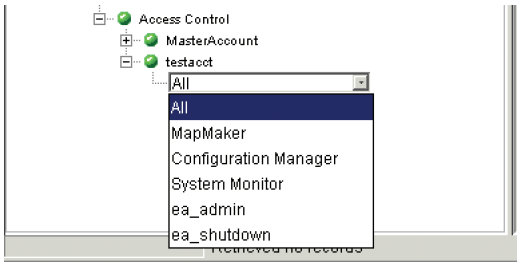


Figure 273. Adding a New ACL Permission Record

3 From the pull down menu in the tree view panel, select the resource for which you wish to create permissions.

4 In the Properties panel, in the **Value** column, click on the **Application** field and select the value from the drop down list.

5 Again in the **Value** column, click on the **Resource** field, and select the value from the drop down list. The resource list for any selected application includes a list of the available resources for that application, and the selection `All`, to designate that all the resources for the application will be ACL-controlled.

Modifying the ACL Permission Records

To modify the ACL Permission records:

1. Login to the Configuration Manager using the Administrator account and password.
2. In the Tree view under the Access Control node, and under the chosen ACL Account record, select the ACL permission record that is to be modified.
3. In the Property panel, in the **Value** column, click on the field that is to be modified (**Application**, **Resource**, or **Permissions**) and use the drop down list to change the value as desired.
4. Click **Save and Apply** and the record is modified.

Permission Records Usage

Duplicate Permissions

Creation of duplicate permission records for any ACL Account record is not permitted. An attempt to create a permissions record for a resource already having those permissions, results in the following warning dialog box.

Figure 274. Duplicate Permission Records for ACL Account Warning

For example, when a permissions record for Application ="All", Resource ="All", and Permissions="All" exists, and the user attempts to create another permissions record for the same ACL Account, the warning dialog is displayed, and the action is not allowed. Because the account already has "All" permissions, any additional permissions record would be superfluous.

Overlapping Permissions

Overlapping permissions occur when permissions records are created that duplicate selections for a specific Application, Resource, or Permissions. When overlapping permissions are created, the result can be undefined behavior by applications that access a controlled resource. There are three instance where overlapping permissions can occur:

- Creation of a permissions record with "Application = All" permissions when a permissions record granting explicit permissions (e.g. "Application=Configuration Manager" "Resource = Database Access" "Permissions = Read/Write") already exists.

- Creation of a permissions record for a specific Application with "Resource=All" permissions when a permissions record granting explicit permissions (e.g. "Resource = Database Access") already exists.
- Creation of a permissions record for a specific Application and Resource that grants opposing permissions. For example, a permissions record exists for "Application = Configuration Manager" "Resource = Database Access" "Permissions = Read/Write". A second permission record is created "Application = Configuration Manager" "Resource = Database Access" "Permissions = Read Only". An overlap exists for the Permissions, and will result in undefined behavior by any applications that access the database.

Application Behavior With ACL-Controlled Environments

This section describes the behavior of ACL-enhanced applications when accessing an ACL-enabled environment.

Configuration Manager

When connecting to an access controlled Environment Manager or Resource Server, the login dialog box is displayed.

If a non-existent account/password is entered, or if **Cancel** is selected, the message `Unable to login` is shown on the status line. When attempting to open the database without database access permissions, the **Permissions Error** dialog box appears.

Figure 275. Permissions Error Dialog Box

System Monitor

The System Monitor does not prompt for a login until a controlled resource is accessed. It then presents the login dialog box. If the user does not have the proper permissions or clicks **Cancel**, the **Permissions Error** dialog box appears.

MapMaker

When deploying a map, a list of Resource Server URLs is shown. When one is selected that points to an access controlled environment, the login dialog box is displayed.

ea_shutdown

When attempting to shut down an access controlled environment, a prompt for an account name and password is displayed.

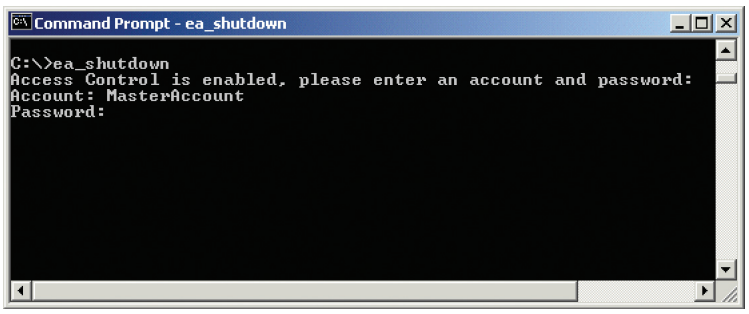


Figure 276. A prompt for an account name and password in ea_shutdown

When the wrong account name or password is entered, the following prompt for a valid account name and password is displayed.

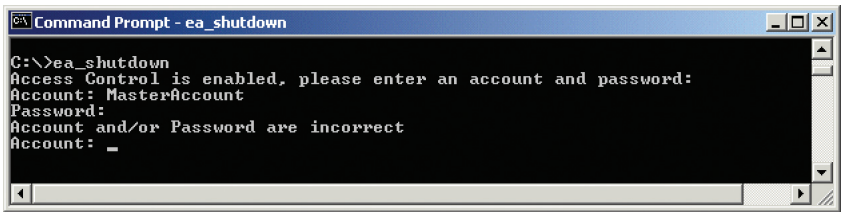


Figure 277. A prompt for a valid account name and password

ea_admin

When attempting to use the “stop” command to stop a JCluster on an access controlled environment, the prompt for an account name and password occurs as it does for `ea_shutdown`.

Compatibility with Previous Versions of webMethods JI

When a new version of webMethods JI which contains Access Control support is installed, webMethods JI works the same as before unless Access Control is explicitly enabled. By default, Access Control is disabled, and must specifically be enabled, as previously described, to implement the Access Control functionality. The following section details the restrictions and capabilities involved when a previous version of webMethods JI attempts to interface with the newer ACL-controlled environment.

Pre-ACL Environment Manager Access to ACL-Enabled Resource Servers

By default, pre-ACL Environment Managers cannot access Resource Servers that have ACL enabled. This includes:

- Access to the resource database, which is configured by the `resourcename` parameter in the *envmgr.cfg* file.
- Access to the license server, which is configured by the `license` parameter in the *envmgr.cfg* file.

When attempting to access an ACL-enabled Resource Server, the following occurs:

Trying to access an ACL-enabled database	The Environment Manager does not start and the *envmgr_stdout.txt* log file keeps growing until the Environment Manager process is killed.
Trying to access an ACL-enabled license server	The Environment Manager does not start and a Null Pointer Exception error is logged in the *envmgr_stderr.txt* log file.

Enabling Access to an ACL-Enabled Database

To enable pre-ACL Environment Manager access to an ACL-enabled database, perform the following steps:

1. Use the Configuration Manager to add an ACL account to the database.
2. Add a permission record, with the following properties, for the newly created account.

Property	Value
Application	Configuration Manager
Resource	Database Access
Permissions	Read Only

Figure 278. Adding a Permission Record for a New Account

3. Edit the *envmgr.cfg* file and add the following lines:

```
account=<account name>
password=<password>
```

Where `<account name>` is the name of the ACL account, and `<password>` is the password.

Enabling Access to an ACL-Enabled License Server

To enable a pre-ACL Environment Manager access to an ACL-enabled license server, perform the following steps:

1 Use Configuration Manager to add a new ACL account record to the database.

> **Note:** It is not necessary to create a Permissions record to access the license server.

2 Edit the *envmgr.cfg* file and add the following lines:

```
account=<account name>
password=<password>
```

Where `<account name>` is the name of the ACL account, and `<password>` is the password.

Prior Application Access to ACL-Enabled Environments

The Access Control List feature was introduced with JI version 3.5 patch 8. While it is not recommended, this section describes what occurs when attempting to access an ACL-enabled environment with previously released applications which do not include ACL functionality.

Configuration Manager

When trying to connect to an ACL-enabled environment, the message on the status line reads "`Could not connect...`".

System Monitor

When attempting to access a controlled resource, a message on the status line may indicate that the operation was aborted, but in any case, the operation does not take place.

MapMaker

When attempting to deploy a map to an ACL-enabled environment, a Login window appears. If you click on the **Cancel** button, a message indicating that the map could not be deployed is displayed.

ea_admin

The `ea_admin stop` command shows no errors when attempting to stop a JCluster in a controlled environment, but the JCluster is not stopped.

ea_shutdown

When attempting to shut down an ACL controlled webMethods JI 4.5.5 or older environment, from the following is displayed:

```
Shutting down the Environment Manager...
Environment Manager was stopped.
Shutting down the Resource Server...
Resource Server could not be stopped.
Shutting down the MySQL daemon...
MySQL daemon was stopped.
```

The Environment Manager will still be running, contrary to the display. The Resource Server will still be running, but the MySQL daemon will be stopped.

Attempting to shut down a JI 4.5.6 or newer Resource Database server with the ea_shutdown command from a prior release, will result in the shutdown command hanging due to incompatibility in Resource Database protocol.

Chapter 15. Protocol Agents

Overview of Protocols Supported with webMethods JI

webMethods JI uses Protocol Agents to manage communication protocols used for communication between webMethods JI services and the host application. Protocol Agents are not separate executables, but are included in the service executable code. Because this code is generated automatically from MapMaker, the webMethods JI graphical development environment, developers can create services that interact with legacy applications without having to code for the underlying communication protocol.

webMethods JI Protocol Agents support a number of standard protocols:

- TN3270/TN3270E Models 2-5.
- TN5250 Standard (80 column) and Large (132 column) screens.
- Telnet: VT100, VT220, and VT320 (text only).

The webMethods JI Compatibility Kit

The webMethods JI Compatibility Kit is used to execute "C" services developed in webMethods JI version 2.x, and uses different Protocol Agents than those described in this chapter. This chapter refers specifically to Protocol Agents used by webMethods JI Java services developed in MapMaker.

For a description of Protocol Agents used with the Compatibility Kit, see the *webMethods JI Compatibility Kit User's Guide*.

Defining the Communications Protocol

Communications Protocols are defined in MapMaker in the **Hosts** dialog box by selecting the Communications Protocol and the model type from drop-down lists included in the dialog box. For more information about the Hosts dialog box, see "Character Mode Data Streams" on page 128.

Trace Information

Protocol Agents traces are logged to the service log file, if the service log level is set to 9. For more information about logging, see the *webMethods JI Supplemental Reference Guide Chapter 5: Logging Information*.

Protocol Agent Terminal Window

A **P**rotocol **A**gent **Term**inal window is referred to generically as a PATerm. It specifically refers to the PATerm written in the "C" programming language. A JTerm, or Java Terminal window, refers to a PATerm written in the Java programming language.

Protocol Agent terminal windows can be used to monitor the Protocol Agent's presentation space, allowing the user to debug the interaction between a service and a legacy application. A write-enabled PA terminal window also provides the user with a way to input data into a legacy application.

Creating a Protocol Agent Terminal (PATerm) Window

Creating a Server-side Terminal Window

Protocol Agent terminal windows can be displayed on the machine on which the Environment Manager is running. The terminal window can be created as soon as a service connection is made to the legacy host, which is helpful for debugging initialization methods. The terminal window can also be created when a client connects to the service.

Server-side terminal window behavior is established for each service in the Configuration Manager, using the Server-side Terminal Window record. This record can be set in the Service Master node for a service, and can be overridden by the same record in the Service Detail node.

For more information about Service Masters and Service Details, see Chapter 12 - "Configuration Manager" on page 455.

The Server-side Terminal Window Record

The Server-side Terminal Window Record, set in the Configuration Manager for either Service Masters or Service Details, determines whether a server-side terminal window will be displayed. If this record is set to `Enabled`, the terminal window is displayed on the host machine on which the Environment Manager is running, unless the $DISPLAY environment variable is sending the display to another machine.

The behavior of this option varies depending on the setting for the Service Master's Host Connection Default State and Service Detail's Minimum Number of Concurrent Instances or the Service Master's Default Minimum Number of Concurrent Instances records.

If either Minimum Number of Concurrent Instances record is set to any non-zero number, the following occurs based on the setting of the Service Master's Host Connection Default State record:

- If Host Connection Default State is set to Dynamic, the Server-side terminal window opens when the service makes a connection to the host.
- If the Host Connection Default State is set to Pooled, the terminal window opens when the service starts.

Compatibility Kit Server-side Terminal Windows

For information about creating server-side terminal windows using "C" services (used with the Compatibility Kit), see the Protocol Agent information in the *webMethods JI Compatibility Kit User Guide*.

Creating a Client-side Terminal Windows

Protocol Agent terminal windows can be created using the "C" or JClient3 Client Libraries. They are created in thread-safe and Win32 clients using the EAClntPAgentMakeTerminal() function, for non-threaded clients using the EA_ClntPAgentMakeTerminal() function, and for JClient3 clients using the makeTerminalWindow() method in the Service Connection class.

For more information about these functions, see the *webMethods JI API Function Reference*.

Attaching Terminal Windows to Protocol Agents

For clients, a terminal window may be attached to a Protocol Agent that is already running. For thread-safe and Win32 clients, use the EAClntPAgentAttachAllTerminals() function, for non-threaded clients, use the EA_ClntPAgentAttachAllTerminals() function, and for JClient3 clients use the attachAllTerminalWindows() method.

To disconnect the terminal window from the Protocol Agent, use the EAClntPAgentCloseAllTerminals() or EA_ClntPAgentCloseAllTerminals() functions, or closeAllTerminalWindows() method for JClient3 clients.

In order for a client to attach a Protocol Agent terminal window to an already running Protocol Agent, it must attach terminal windows to all Protocol Agents connected to the corresponding service. A client can only close all PATerms associated with a specific service via EAClntPAgentCloseAllTerminals(). There is no option for a client to close a specific PATerm.

For more information about these functions, see the *webMethods JI API Function Reference*.

Components of a Protocol Agent Terminal Window

Server-side Terminal Window

This window is displayed when a server-side terminal window is created (except for Compatibility Kit server-side terminal windows). The terminal window contains a representation of the Presentation Space, and contains the other components of a normal TN3270 client, such as the Operator Information Area.

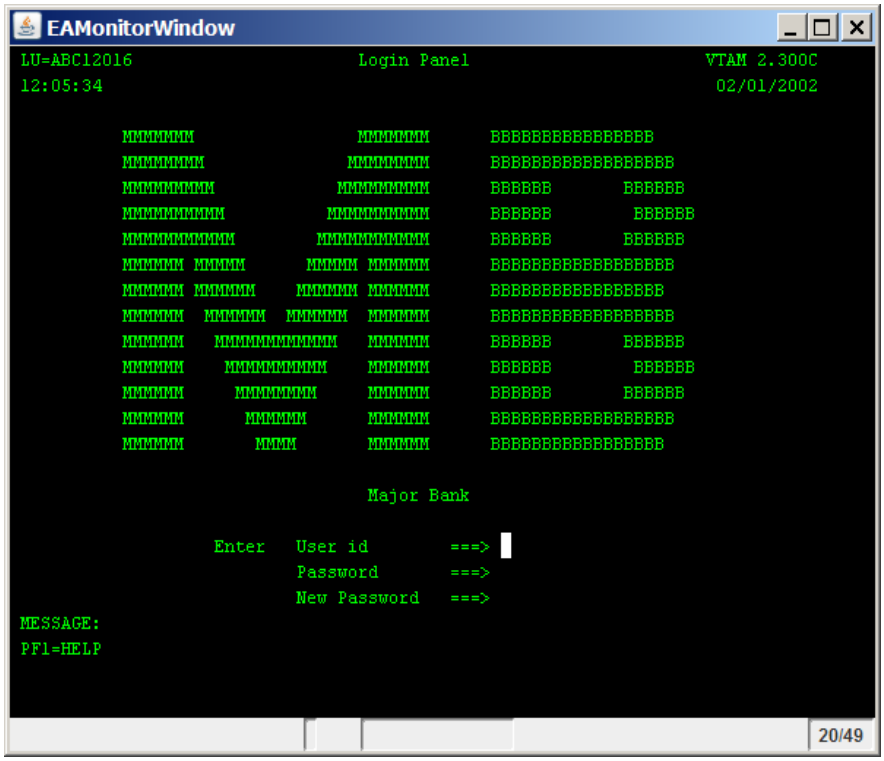


Figure 279. Server-side Terminal Window

PATerm

The PATerm terminal window is created when a “C” or JClient3 client creates a terminal window. This window is displayed in a command window. The title bar contains status information about the Protocol Agent connection, such as the IP address, port number, and session ID of the Protocol Agent. The Read/Write status of the Protocol Agent is also displayed.

The terminal window itself contains a representation of the Presentation Space, and contains the other components of a normal TN3270 client, such as the Operator Information Area.

A “C” PATerm window relies on the *paterm.kbm* file for keyboard mapping. The client must use the proper APIs to set the location of the webMethods JI installation directory, which allows the PATerm to find the keyboard map file. If the “C” PATerm does not locate the keyboard mapping file, an error dialog is shown and the PATerm exits.

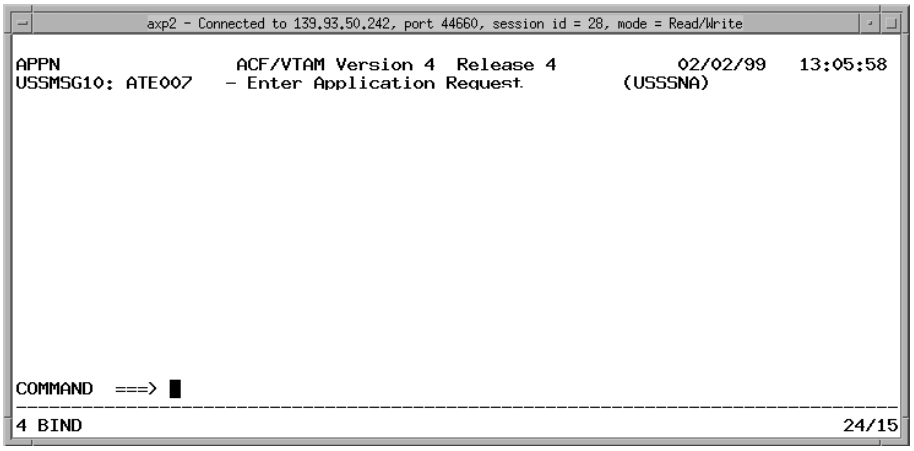


Figure 280. The PATerm Terminal Window

JTerm

The JTerm (Java terminal) window is created when a JClient3 Java client creates a terminal window. It is also created when a “C” client creates a terminal window when the `WIDESTRING` or `JTERM` options are set to `TRUE`. The title bar contains status information about the Protocol Agent connection, such as the IP address, port number, and session ID of the Protocol Agent. The Read/Write status of the Protocol Agent is also displayed.

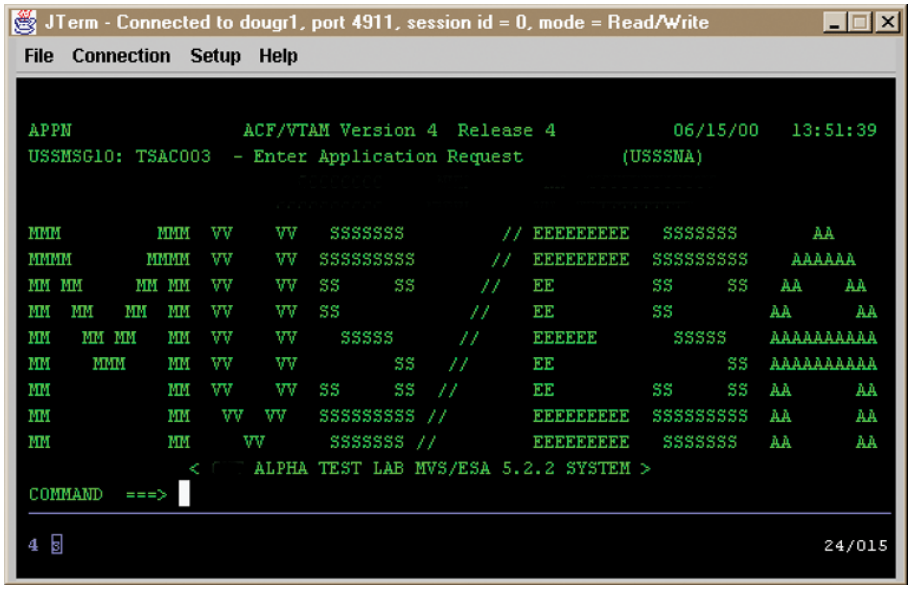


Figure 281. The JTerm (Java Terminal) Window

A JTerm window also uses the *paterm.kbm* file to determine the proper keyboard mapping. The client must use the proper APIs to set the location of the webMethods JI installation directory, which allows the JTerm to find the keyboard map file. If the JTerm does not locate the keyboard mapping file, default values are used. See “Default Keyboard Mapping for JTerm Windows” on page 573 for specific key mappings.

JTerm Properties

The JTerm provides a way to view various connection-related parameters. To view this screen select **File** > **Properties**, and observe the Properties display:

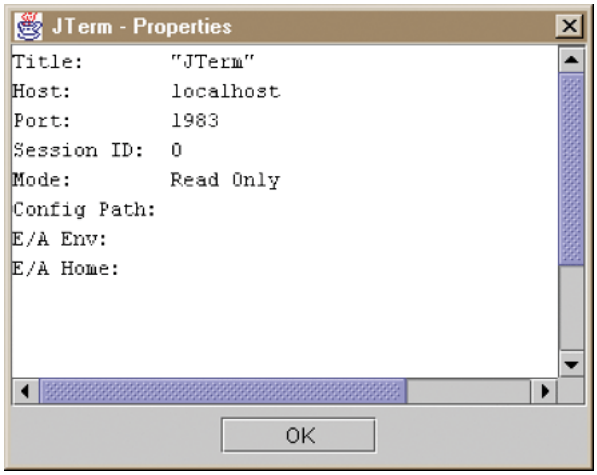


Figure 282. The JTerm Properties Window

JTerm Logging

The JTerm also provides a way to view logging information. To view this screen select **Help** > **JTerm Log**, and observe the JTerm Log display:

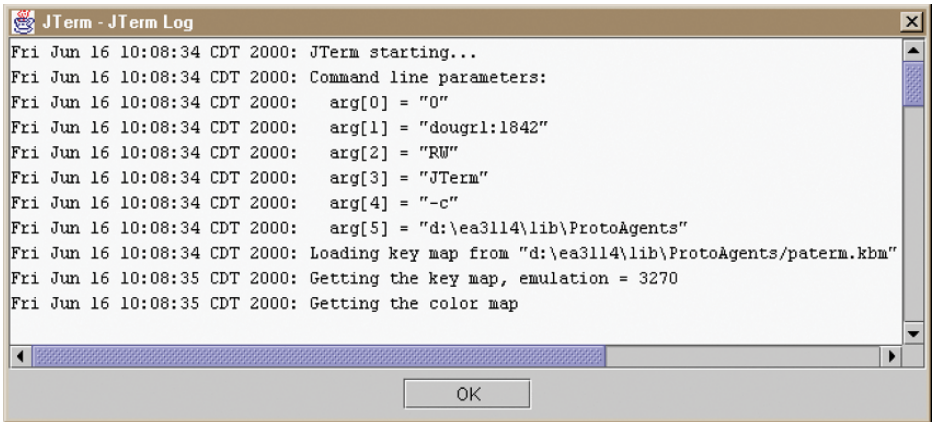


Figure 283. The JTerm Log Screen

Macro and Cut/Paste

Default Keyboard Mapping for JTerm Windows

In the Default Keyboard Mapping table, the left side shows the emulation function, the right side is the keyboard key sequence that produces the emulation function.

Table 3. Default Keyboard Mapping

Emulation Function	Keyboard Sequence
3270 Emulation	
PF1	F1
PF2	F2
PF3	F3
PF4	F4

Emulation Function	Keyboard Sequence
PF5	F5
PF6	F6
PF7	F7
PF8	F8
PF9	F9
PF10	F10
PF11	F11
PF12	F12
PF13	Shift-F1
PF14	Shift-F2
PF15	Shift-F3
PF16	Shift-F4
PF17	Shift-F5
PF18	Shift-F6
PF19	Shift-F7
PF20	Shift-F8
PF21	Shift-F9
PF22	Shift-F10
PF23	Shift-F11

Emulation Function	Keyboard Sequence
PF24	Shift-F12
PF9	Control-F1
PF10	Control-F2
PF11	Control-F3
PF12	Control-F4
PF21	Control-F5
PF22	Control-F6
PF23	Control-F7
PF24	Control-F8
Enter	Carriage Return
Cursor Left	Backspace
Cursor Left	left arrow
Cursor Right	right arrow
Cursor Up	up arrow
Cursor Down	down arrow
Home	Home
Insert	Insert
Delete	Delete
Tab	Tab

Emulation Function	Keyboard Sequence
Backtab	Shift-Tab
Attention	Control-a
Backtab	Control-b
Clear	Control-z
Dup	Control-d
Delete	Control-e
Tab	Control-f
Cursor Select	Control-g
Cent Sign	Control-c
Insert	Control-o
FIeld Mark	Control-j
New Line	Control-n
Erase EOF	Control-p
Home	Control-q
Reset	Control-r
SysReq	Control-s
PA1	Control-t
PA2	Control-u
PA3	Control-u

Emulation Function	Keyboard Sequence
Erase Input	Control-x
5250 Emulation	
F1	F1
F2	F2
F3	F3
F4	F4
F5	F5
F6	F6
F7	F7
F8	F8
F9	F9
F10	F10
F11	F11
F12	F12
F13	Shift-F1
F14	Shift-F2
F15	Shift-F3
F16	Shift-F4
F17	Shift-F5

Emulation Function	Keyboard Sequence
F18	Shift-F6
F19	Shift-F7
F20	Shift-F8
F21	Shift-F9
F22	Shift-F10
F23	Shift-F11
F24	Shift-F12
F9	Control-F1
F10	Control-F2
F11	Control-F3
F12	Control-F4
F21	Control-F5
F22	Control-F6
F23	Control-F7
F24	Control-F8
Enter	Enter
Cursor Left	Backspace
Cursor Left	left arrow
Cursor Right	right arrow

Emulation Function	Keyboard Sequence
Cursor Up	up arrow
Cursor Down	down arrow
Home	Home
Insert	Insert
Delete	Delete
Tab	Tab
Backtab	Shift-Tab
Attention	Control-a
Backtab	Control-b
Clear	Control-z
Dup	Control-d
Delete	Control-e
Tab	Control-f
Cursor Select	Control-g
Cent Sign	Control-c
Insert	Control-o
New Line	Control-n
Erase EOF	Control-p
Help	Control-j

Emulation Function	Keyboard Sequence
Home	Control-v
Field Minus	Control-k
Field Plus	Control-l
Field Exit	Control-q
Reset	Control-r
SysReq	Control-s
PA1	Control-t
PA2	Control-u
PA3	Control-u
Erase Input	Control-x
Roll Down	Page Up
Roll Up	Page Down
Telnet Emulation	
F1	F1
F2	F2
F3	F3
F4	F4
F5	F5
F6	F6

Emulation Function	Keyboard Sequence
F7	F7
F8	F8
F9	F9
F10	F10
F11	F11
F12	F12
F13	Shift-F1
F14	Shift-F2
F15	Shift-F3
F16	Shift-F4
F17	Shift-F5
F18	Shift-F6
F19	Shift-F7
F20	Shift-F8

Write-enabled Terminal Windows

A write-enabled Protocol Agent terminal window provides a means of entering input to a legacy application.

For clients, write-enabled terminal windows are created using the `TERMOPTS` option for the EAClntPAgentMakeTerminal() or EA_ClntAgentMakeTerminal() functions.

For clients, write-enabled PATerms can only be created via EAClntPAgentMakeTerminal(). Hence, a client will not be allowed to attach a write enabled PATerm to an already running protocol agent.

Remote JTerm (PATerm) Windows

This section describes the webMethods JI Remote JTerm feature. The Remote JTerm is a Java Protocol Agent terminal window that is designed to run on a machine other than the machine on which the webMethods JI client is running. Currently, this feature is available to JClient3 clients only.

A typical use for a Remote JTerm would be for a user on one machine using a web browser to access, for example, a Siebel application running on another machine; a JTerm window is required to allow user interaction. The Remote JTerm feature allows the user to have a JTerm window on the machine on which the browser is running interfacing with the Siebel application running on another machine. The Remote JTerm is controlled by the client via a control connection between the client and the JTerm. The Siebel application that the user is accessing connects to webMethods JI through a MQ Gateway or the HTTP Gateway.

A JTerm is a Protocol Agent Terminal (PATerm) written in the Java programming language, therefore the generic parameter "`remotePATermID`" is used to identify the Remote JTerm (PATerm) to the Environment Manager.

In the Figure 284, the solid lines show data transfer connections, and the dashed lines show Remote JTerm control connections.

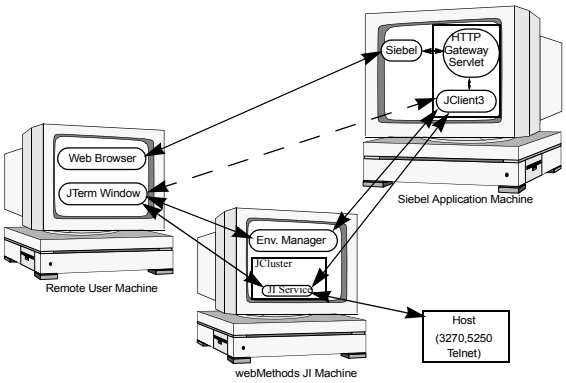


Figure 284. Remote JTerm Interface

Installing and Configuring the Remote JTerm

The Java Virtual Machine (JVM) JRE/JDK, is required to run the Remote JTerm feature.

Note: Due to the nature of Java technology where objects are exchanged across a network, it is recommended that all *webMethods JI* Java-based programs, regardless of platform, be run using the same version of the Java Runtime Environment to avoid any JVM version incompatibilities.

Installing the Remote JTerm Feature - Stand Alone

The Remote JTerm feature is delivered with the webMethods JI and is located in the *<JI_Install_Dir>/lib* directory. The file *remote_jterm.jar*, contains the JAR files, batch file (Windows), shell script (UNIX), and JTerm HTML (to run JTerm as an applet).

Installing the Remote JTerm when webMethods JI is not Installed

If webMethods JI has not been installed on the machine on which the Remote JTerm feature is to be installed, obtain the *remote_jterm.jar* file from the *<JI_Install_Dir>/lib* directory on a machine where webMethods JI *has* been installed.

To remap keys, you may copy the *paterm.kbm* keyboard mapping file for use by the Remote JTerm.The *paterm.kbm* file is normally found in the *<JI_Install_Dir>/lib/ProtoAgents* directory.

For UNIX or Windows

To install the Remote JTerm application on a UNIX or Windows platform:

1. Identify (or create) the directory in which to install this feature, and copy the file r*emote_jterm.jar,* located in *<JI_Install_Dir>/lib,* into the directory.
2. From the command line extract the files using the following command:

   ```
   jar -xvf remote_jterm.jar
   ```
3. Observe that the following files are installed in the directory:

Files/Directory	Description
jterm.jar	The JAR file containing the JTerm classes.

Files/Directory	Description
jterm.bat	A sample Windows batch file for starting JTerm from the command line.
jterm.sh	A sample UNIX shell script for starting JTerm from the command line.
jterm.html	Sample HTML for running JTerm as an applet. To be placed in the Apache Web Server directory.
jterm.cer	JTerm certificate file. (Only required when running as an applet.)

Note: For Windows, users may choose to use an unzip utility to perform the extraction.
After extraction, perform the following command on the jterm.sh file to make it executable in the UNIX environment:

```
chmod +x  jterm.sh
```

Installing the Remote JTerm Feature - Java Applet

The Remote JTerm feature may also be run as an applet. To run the Remote JTerm feature as an applet requires the use of Internet Explorer (5.5 or newer) and the Oracle Java Plug-in must be installed and enabled. JTerm may run as an external frame (the default), or it may be embedded in a web page. To run the JTerm as an applet, an applet tag is included the HTML that the browser accesses. The following sections describe how to set up the applet tag and also describes applet security issues.

The Remote JTerm HTML file is delivered with the webMethods JI, and is located in the *<JI_Install_Dir>/lib* directory in the file *remote_jterm.jar*. This JAR file contains the *jterm.html* file (with the Java Applet tags enabling this feature) and the *jterm.jar* file that contains the JTerm classes.

Web Server Installation

Examples provided in this documentation are specific for installation on the Apache Web Server. For installation on other web servers, visit the vendor's web site for installation instructions and additional information.

For Installation in the Apache Web Server

To install the Remote JTerm applet in the Apache Web Server:

1. In the *<Apache_Install_dir>/htdocs* directory create a directory called *jterm*, and copy the file r*emote_jterm.jar,* located in *<JI_Install_Dir>/lib,* into the new directory.
2. From the command line, extract the files using the following command:

   ```
   jar -xvf remote_jterm.jar
   ```
3. Observe that the following files are installed in the directory:

Files/Directory	Description
jterm.jar	The JAR file containing the JTerm classes.
jterm.bat*	A sample Windows batch file for starting JTerm from the command line.
jterm.sh*	A sample UNIX shell script for starting JTerm from the command line.
jterm.html	Sample HTML for running JTerm as an applet. To be placed in the Apache Web Server directory.
jterm.cer	JTerm certificate file.

* These files are not needed when the Remote JTerm feature is enabled via the web browser.

For Windows, users may choose to use an unzip utility to perform the extraction.

1. Delete the *jterm.bat* and *jterm.sh* files from the directory; they are not needed for this installation.
2. Copy the following files, *jsse.jar, cert.jar, jnet.jar* located in *<JI_Install_Dir>/classes,* into the *<Apache_Install_dir>/htdocs/jterm* directory. These files are needed to enable SSL encryption when SSL is instantiated by the client.

Configuring the Remote JTerm Applet

In the *jterm.html* file the following attributes must be set in the APPLET tag:

Attribute	Value
CODE	com.jacada.jterm.JTApplet
ARCHIVE	No encryption - jterm.jar SSL Encryption - jterm.jar, jsse.jar, jnet.jar, jcert.jar **Note:** If the JClient3 client enables SSL encryption, the ARCHIVE parameter must include the JSSE JAR files listed here. These files are found in the <JI_Install_Dir>/classes directory. See "Enabling and Using SSL" on page 589 for information on implementing SSL encryption.
WIDTH	573
HEIGHT	520

The JTerm applet also includes a number of param name attributes that can be set to configure the Remote JTerm window. The JTerm applet has the following param name attributes:

PARAM NAME	Value
environmentManager	The name of the machine on which the webMethods JI Environment Manager is running, in <host>:<port> format. This may also be expressed as the numeric IP Address: port number.
remotePATermID	The environmentManager Remote PATerm ID value.
launcher	Set to Y to enable the "launcher" feature. **Note:** If either of the environmentManager or remotePATermID parameters are not set, the launcher is enabled automatically, regardless of the launcher setting.

PARAM NAME	Value
embedded	Set to Y to embed the JTerm in the browser page. Default is N, resulting in an external frame.
title	The JTerm title. This parameter is optional.
readWriteOption	RW = Read/Write, RO = Read Only
kbmURL	The URL of the *paterm.kbm* keyboard re-map file. **Note:** When the paterm.kbm file is not specified, default keyboard mappings are used. See Figure 3 for the mappings.
fixedSize	Set to Y to enable the fixed size feature, indicating that the JTerm window should stay a fixed size and the font should be adjusted to fit in the window. 'N' indicates that the JTerm window should be re-sized.
logFile	The full path of the log file.
logLevel	Specifies the logging level (1-9).

Editing the *jterm.html* File

To edit the *jterm.html* file:

1. Use a text editor to open the *jterm.html* file and observe the following applet code is present:

```
<applet code="com.jacada.jterm.JTApplet"
        archive="jterm.jar"
width=573 height=520>
```

   Modify the width or height as desired.
2. The param name attributes are commented out in the file. Remove the leading (<!--) and trailing (-->) comment delimiters for the parameters that will be set for the Remote JTerm.

3 Set each param value as desired to properly configure the Remote JTerm feature. Save the file.
4 Using a text editor, open the *index.html* file in the *<Apache_Install_dir>/htdocs* directory and add a link to the *jterm.html* page just modified. Save the file.

Importing the Certificate in Internet Explorer

To enable running the Remote JTerm as a Java applet, a certificate must be imported to the Internet Explorer web browser.

1 Launch the Internet Explorer. Select **Tools** > **Internet Options**. The **Internet Options** dialog box is displayed.
2 Select the **Content** tab and click **Certificates**. The **Certificate Manager** dialog box is displayed.
3 Click **Import**. The **Certificate Manager Import Wizard** dialog box is displayed. Click the **Next** button to continue and the **Select file for import** window is displayed. Click **Browse** and in the **Open** dialog box, select **X509 Certificate (*.cer, *.crt)** from the **Files of type** drop-down list.
4 Navigate to the directory in which the file *jterm.cer* is stored, select the file, and click **Open**.
5 With the file selected, click **Next** and observe the **Select a Certificate Store** window is displayed. Select the radio button titled **Automatically select the certificate store based on the type of certificate**. Click **Next** to continue. The Completing the Certificate Manager Import Wizard dialog box is displayed, showing the certificate file chosen.
6 Click **Finish**. The Root Certificate Store dialog box is displayed prompting you to add the certificate to the Root Store. Click **Yes** to add the certificate, and an **Import Successful** message box is displayed. Click **OK**.
7 Click **Close** to close the **Certificate Manager** dialog box, and click **OK** to close the **Internet Options** dialog box.

Obtaining a Certificate From the Web Server

When running the Remote JTerm as a Java applet, a certificate may be provided via the web server. To make the certificate available to users, establish a link via the index page to the stored certificate file. Users may then import the certificate file using the procedure “Importing the Certificate in Internet Explorer” on page 588.

Importing an Updated Certificate in Internet Explorer

When an update to the Remote JTerm is received, the existing JTerm certificate must be removed, and a new certificate must be imported to the Internet Explorer web browser.

1. Launch the Internet Explorer. Select **Tools** > **Internet Options**. The **Internet Options** dialog box is displayed.
2. Select the **Content** tab and click **Certificates**. The **Certificate Manager** dialog box is displayed.
3. Select the **Trusted Root Certification Authorities** tab. The list of certificates is displayed.
4. Scroll down to the JTerm certificate and select it. Click the **Remove** button to remove it from the list.
5. Click **Import**. The **Certificate Manager Import Wizard** dialog box is displayed. Click the **Next** button to continue and the **Select file for import** window is displayed. Click **Browse** and in the **Open** dialog box, select **X509 Certificate (*.cer, *.crt)** from the **Files of type** drop-down list.
6. Navigate to the directory in which the new *jterm.cer* file is stored, select the file, and click **Open**.
7. With the file selected, click **Next**. The **Select a Certificate Store** window is displayed. Select the radio button titled **Automatically select the certificate store based on the type of certificate**. Click **Next** to continue. The **Completing the Certificate Manager Import Wizard** dialog box is displayed, showing the certificate file chosen.
8. Click **Finish**. The **Root Certificate Store** dialog box is displayed prompting you to add the certificate to the Root Store. Click **Yes** to add the certificate. An **Import Successful** message box is displayed. Click **OK**.
9. Click **Close** to close the **Certificate Manager** dialog box, and click **OK** to close the **Internet Options** dialog box.

Enabling and Using SSL

If the client enables SSL encryption, the JTerm requires the use of the Oracle JSSE package which includes the *jsse.jar*, *jnet.jar* and *jcert.jar* files. To be used by JTerm these JAR files must have permissions granted to them, or they need to be signed by the customer.

Granting Permissions to the JSSE JAR Files

Permissions are granted by editing the Java policy file and adding necessary permissions. There are policy files located in the Java Installation directory (system policy) and in the user's home directory (user policy). Either the system policy file or the user policy file may be edited to grant permissions.

The system policy file is located at:

```
java.home/lib/security/java.policy (UNIX)
java.home\lib\security\java.policy (Windows)
```

A system policy file on a Windows machine might appear in the directory as shown here:

```
C:\Program Files\JavaSoft\JRE\1.4.2\lib\security
```

The user policy file is normally located at:

```
user.home/.java.policy (UNIX)
user.home\.java.policy (Windows)
```

An example of a user policy file directory for a user name “brian” on a Windows machine:

```
C:\Documents and Settings\brian
```

Modify the policy file by adding the following permissions for each of the JSSE JAR files:

```
grant codeBase "<URL for the /jterm/jsse.jar>" {
    permission java.security.AllPermission;
};

grant codeBase "<URL for the /jterm/jnet.jar>" {
    permission java.security.AllPermission;
};

grant codeBase "<URL for the /jterm/jcert.jar>" {
    permission java.security.AllPermission;
};
```

An example of a URL for the `/jterm/ jsse.jar` would be:

```
http://<Webserver containing the Remote JTerm Applet:port>/jterm/jsse.jar
```

When permissions are modified, the policy file must be modified on every user’s machine.

Signing the JSSE JAR Files

An alternative to modifying the policy file on each user’s machine is to sign the JSSE JAR files on the web server machine where the Applet resides. Oracle policy prevents Software AG from distributing modified versions of these files, but customers may modify the files.

Signing these JAR files requires JDK 1.4.2_05 or newer from Oracle. The *keytool* and *jarsigner* programs, which are located in the *<JDK Install Dir>/bin* directory, are used to sign the JAR files.

To sign the JAR files, perform the following steps:

1. Use the keytool program to create a keystore file.
2. Use the keytool program to create an X509 certificate file. This certificate must be imported into Internet Explorer. See the section titled "Importing the Certificate in Internet Explorer" on page 588 for instructions.
3. Use the jarsigner program to sign the JAR files.

Users may wish to create a batch (.bat) or shell script (.sh) file to retain the command syntax and make future edits easier. The following example shows how a company called "My Company" would sign the JAR files:

> **Note:** Though displayed here on two lines, each of the keytool commands is typed on a single line.

```
keytool -genkey -alias MyCompany -dname "cn=My Company" -validity 3650
-storepass pw -keypass mycomp1 -keystore mycomp.ks

keytool -export -alias MyCompany -rfc -file mycomp.x509 -keystore mycomp.ks
-storepass mycomp1

jarsigner -keystore mycomp.ks -storepass mycomp1 jsse.jar MyCompany
jarsigner -keystore mycomp.ks -storepass mycomp1 jnet.jar MyCompany
jarsigner -keystore mycomp.ks -storepass mycomp1 jcert.jar MyCompany
```

When the JTerm applet is run the first time, it needs to use the JSSE class files and the user will see a dialog similar to the one shown in Figure 285.

Launching a Remote JTerm From the Command Line

To launch a stand-alone Remote JTerm window:

1. In Windows, launch a Command Prompt window. In UNIX start a console or terminal. Navigate to the directory containing the *jterm.bat* or *jterm.sh* file.
2. From the command prompt, enter a command in the format shown, using the Remote JTerm parameters described below.

> **Note:** Though displayed here on several lines, the command is typed on a single line.

```
java -classpath <full_path_to>\lib\jterm.jar com.jacada.jterm.JTerm
<Environment_Manager_Host_Name>:<port>%<remotePATermID> 0 RW [<title>]
[-e <JI_Install_dir>\config]
```

Parameter	Description
<Environment_Manager_Host_Name>	The name of the machine on which the webMethods JI Environment Manager is running. This may also be expressed as the numeric IP Address.
<port>	The Environment Manager listen port.
<remotePATermID>	The name of the Remote PATermID. Do not use spaces in the Remote PATermID name.
<PA port>	Not used; MUST be set to zero (0).
<Read/Write Options>	RW = Read/Write mode. RO = Read Only mode.
<title>	This parameter is optional. The name to put on the title bar of the JTerm window.
<switches>	
-e <JI config dir>	The path to the */config* directory under the directory in which webMethods JI is installed. This is used by the JTerm to locate the keyboard remap file (*paterm.kbm)*.* See "How PATerm Finds the Keyboard Remap File" on page 593 for more details.
-i <JI install dir>	The path to the directory in which webMethods JI is installed. This is used by the JTerm to locate the keyboard remap file (paterm.kbm).* See "How PATerm Finds the Keyboard Remap File" on page 593 for more details.
-c <keyboard remap file dir>	The path to the directory containing the keyboard remap file (paterm.kbm).*

Parameter	Description
-f <fixed size setting>	Used when the JTerm changes height and/or width. 'Y' indicates that the JTerm window should stay a fixed size and the font should be adjusted to fit in the window. 'N' indicates that the JTerm window should be re-sized.
-l <log file>	The full path of the log file.
-d <log level>	Specifies the logging level (1-9).

Note: * When the paterm.kbm file is not specified, default keyboard mappings are used. See Table 3 for the mappings.

For example:

```
java -classpath d:\EA2000_3.5.1.4\lib\jterm.jar com.jacada.jterm.JTerm
mymachine.com:30001%paterm1 0 RW RJTerm -e d:\EA2000_3.5.1.4\config
```

In this example, webMethods JI is installed on a Windows machine in the *D:\EA2000_3.5.1.4* directory, the Environment Manager is running on port 30001 on the host “mymachine.com”, the PATerm is set to Read/Write mode, the title is “RJTerm” and the Remote PATerm ID is “paterm1”.

3 A Remote JTerm window is displayed with the title, “remotePATermID”, and the phrase “**...waiting for session...**” in the title bar.

When a client gains control of the Remote JTerm window (that is after the Environment Manager has provided the host:port information to the client) and the client subsequently opens a service and invokes a method, the Remote JTerm then displays the information for that method as it occurs.

During the time the client has control of the Remote JTerm, the information in the title bar changes to reflect the connection and port information provided by the client.

The client can also change the Read/Write options, as necessary, to enable user interactions.

How PATerm Finds the Keyboard Remap File

The '-e', '-i' and '-c' switches are used by the PATerm to find the keyboard remap file (*paterm.kbm*) as follows:

- When the -c switch is used it points directly to the directory containing the *paterm.kbm* file.
- When the -i switch is used, the location of the file is determined by the following means. In the */config* directory under the directory specified by the -i switch, the parameter EA_HOME is read from the *environment.ccf* file. Then the path, as specified by EA_HOME points to the *<EA_HOME>/lib/ProtoAgents* directory where the *paterm.kbm* file resides.
- When the -e switch is used, the location of the file is determined by the following means. In the directory specified by the -e switch, the parameter EA_HOME is read from the *environment.ccf* file. Then the path, as specified by EA_HOME points to the *<EA_HOME>/lib/ProtoAgents* directory where the *paterm.kbm* file resides.

Note: IMPORTANT FOR WINDOWS! For the EA_HOME parameter to be properly interpreted by the JTerm.jar, it must be specified with two back slashes after the drive letter.
For example:
EA_HOME=D:\\EA2000_3.5.1.4
Or the parameter may also be specified with a single forward slash:
EA_HOME=D:/EA2000_3.5.1.4

Using the Batch/Shell Script File to Launch the Remote JTerm

The *jterm.bat* or *jterm.sh* file allows users to set the parameters for the Java command that launches the Remote JTerm window to specify a host, port number, PATerm Remote ID, read/write option, JTerm title, keyboard mapping and logging options. These parameters are the same as those used when launching the Remote JTerm from a command line, and they are described here in the command line syntax format as used in the Windows batch file.

Note: Though displayed here on several lines, the command must be placed on a single line in the bat/sh file to ensure proper execution of the command. In the Windows .bat file two '%' characters must appear before the PATermID. For example: `localhost:30001%%PATermID`

To launch the remote JTerm:

1. Use any text editor to open the *jterm.bat* (Windows) or *jterm.sh* (UNIX) file and modify the .bat/sh file to set these parameters:

Note: Items in bold reflect the default command as it appears in the unedited .bat/sh file.

```
java -classpath <full_pathname_to>\jterm.jar com.jacada.jterm.JTerm
<Environment_Manager_Host_Name>:<port>%%<PATermID> 0 <R/W Options>
[<title>] [<switches>]
```

2 Save the modified file.

3 In Windows, launch a Command Prompt window. In UNIX start a console or terminal. Navigate to the directory containing the *jterm.bat/sh* file.

4 From the command prompt, type
   - For Windows: `jterm.bat`
   - For UNIX: `./jterm.sh`

5 A Remote JTerm window is displayed with the title, "remotePATermID as defined", and the phrase "**...waiting for session...**" in the title bar.

Launching a Remote JTerm From a Browser

The Remote JTerm may also be run from a Web browser (IE 5.5 or newer with the Oracle Java plug-in installed). An Applet tag must be included in the HTML that the browser accesses.

To launch a Remote JTerm using the web browser:

1 Launch the Internet Explorer, and in the Address window type the URL of the web server on which the Remote JTerm applet is hosted.

2 When the index page is displayed, click on the link to the Remote JTerm:
   - When the applet is being run the first time, the following dialog box (or similar) is displayed:
     Check the **Always trust content from this publisher** checkbox to prevent this dialog box from being shown in the future. Clicking **Run** without checking the **Always trust content from this publisher** checkbox allows the JTerm applet to run, but the next time it is run, this dialog box is displayed again.

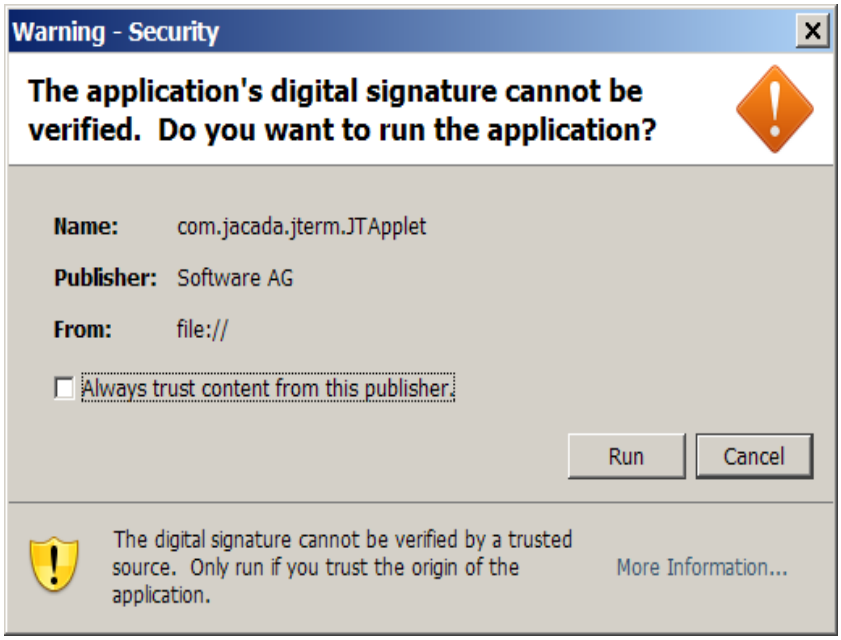


Figure 285. Java Plug-in Security Warning

- When the launcher is enabled, or when either the Environment Manager Host:Port or the Remote PATermID are not defined in the JTerm applet, the **Remote JTerm Launcher** dialog box is displayed. If the parameter names are defined and the launcher is enabled, the fields in the launcher dialog box display the defined parameters.

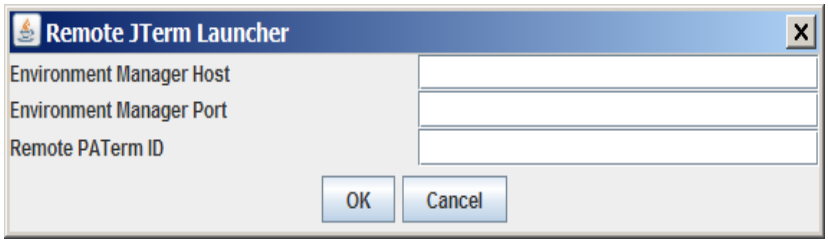


Figure 286. Remote JTerm Launcher Dialog

- The **Environment Manager Host** may be defined by the numeric IP address (e.g., 10.150.8.97) or by its DNS name (e.g, MyMachine.com). Occasionally DNS name reconciliation is not properly configured, and the connection will not be made. If this occurs, check with the System Administrator or use the numeric IP address instead. The **Remote PATermID** must be unique, as duplicate IDs are not allowed.

When the Remote JTerm is run and an external (not embedded) JTerm is specified, an external JTerm window is launched and appears as any other PATerm, as described earlier in this chapter. When an *embedded* PATerm is specified, the Remote JTerm resides within the browser window as shown in Figure 287.

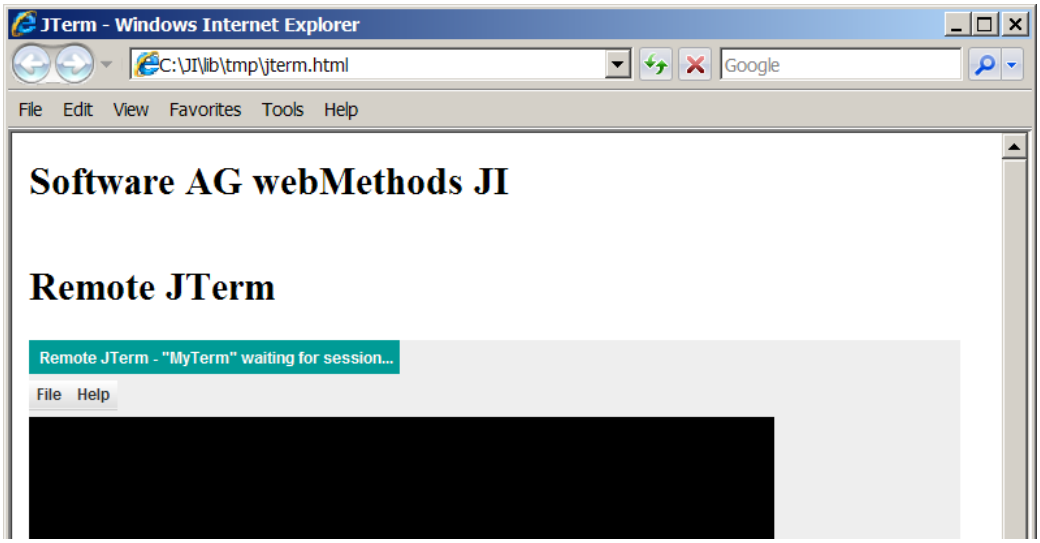


Figure 287. Embedded Remote JTerm

Remote JTerm Errors

This section describes the errors and error conditions that may be encountered when using the Remote JTerm feature.

Duplicate Remote PATerm ID Error Message

If a requested Remote PATerm ID is already in use, the following error dialog box is displayed.

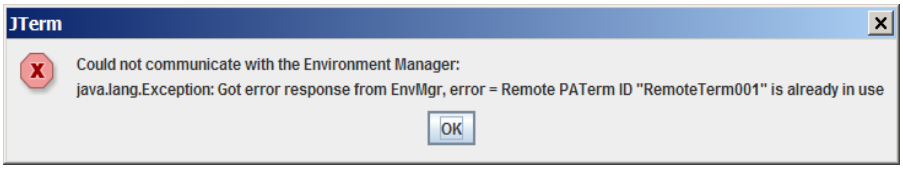


Figure 288. JTerm Error

Connecting to an Old Environment

Attempts to connect a Remote JTerm to an environment that does not have the Remote JTerm feature, results in an error dialog box and/or log an error.

Restarting an Environment

If a Remote JTerm is currently attached to an environment, and that environment is restarted, the JTerm will not be able to detect it, and therefore it must be restarted.

HTTP and MQ Gateway Changes

The HTTP Gateway and the MQ Gateway both accept the PATerm Remote ID in the request. The parameter `EA_REMOTE_PATERM_ID` is passed to the HTTP Gateway via a configuration file setting or via the accessing URL.

Siebel Specific Instructions

When a Siebel user uses the remote PATerm feature, the remote PATerm identifier is passed to the HTTP Gateway. This can be accomplished two ways:

- By editing the URL that is used to access the servlet to set the `EA_REMOTE_PATERM_ID` parameter in the data portion of the URL. The following is an example URL containing the `EA_REMOTE_PATERM_ID` parameter:

  ```
  http://localhost:8080/EAi/httpgateway/config/
  Siebel.cfg?EA_REMOTE_PATERM_ID=paterm1
  ```

- By setting the `EA_REMOTE_PATERM_ID` parameter in the property file that is specified in the URL that is used to access the servlet. The following URL defines a properties file in which the `EA_REMOTE_PATERM_ID` parameter is set.

  ```
  http://localhost:8080/EAi/httpgateway/config/Siebel.cfg
  ```

  To set the parameter in the properties file (*config/Siebel.cfg* in the above example), the following line is added to the file:

  ```
  EA_REMOTE_PATERM_ID=paterm1
  ```

Error Logging

At log level 9, the Environment Manager logs all requests from, and responses to, a client. The Environment Manager considers a Remote JTerm to be a client and is handled as such. Client requests appear in the log with the text “Client request:” followed by the message parameters. Responses to the client appear in the log with the text “Response to client:” followed by the message parameters.

When a Remote JTerm is started, the request parameters include the Remote PATerm ID, the host and the listen port. When a client (i.e., JClient3) takes control of a Remote JTerm, the request parameter is the Remote PATermID and the response parameters are the host and listen port of the Remote JTerm.

When the debug level is at a level >= 3, information is logged when a Remote JTerm is started.

Status

The `ea_admin` command line utility shows all the Remote PATerms with the "status" command. From the command line type `ea_admin status` and the output displayed is similar to that shown here:

```
EnvMgr Status:
  RemotePATerm_0=139.93.8.38:1237%paterm1
  aclEnabled=false
  clientCount=1
  freeMemory=766216
  groupName=eagroup_3514
  idleThreadsInPool=18
  ipAddress=dougr1
  jclusterCount=1
  jserviceClientCount=0
  jserviceCount=0
  jserviceCumClientCount=1
  port=10520
  serviceCount=2
  threadsInPool=20
```

The Remote PATerms are shown in <host>:<port>%<Remote PATerm ID> format.

Chapter 16. Secure Socket Layer (SSL) Implementation

About SSL

The Secure Socket Layer (SSL) protocol was developed to permit secure, encrypted communication between a client and a server. Originally developed for web-based applications, it is also extensible to other computing applications and is used by webMethods JI for encrypting data transferred between a client and a service.

SSL provides a means for secure communication using cryptographic encryption with a selectable set of cipher keys, and server authentication using certificate files. A cryptographic hash function, part of the user-selected cipher suite, ensures data integrity in encrypted communications.

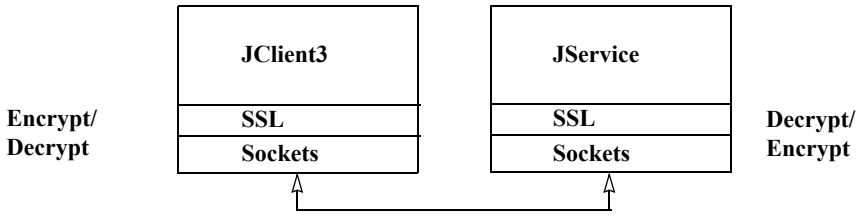


Figure 289. SSL implementation

In the illustration above, the JClient3 application encrypts a service request using the SSL element, routing it to the JService via the Socket connection. The JService receives the request through its socket connection, and decrypts the service request using its SSL logic. The JService then responds to the request and encrypts its reply, and sends it back to the JClient3 via the socket connection. The JClient3 application then decrypts the JService reply, again using the SSL layer.

SSL Requirements and Limitations

Limitations

The following limitations apply to this implementation of SSL in webMethods JI:

- Only JClient3 clients support encryption.
- Only the Java PATerm (JTerm) data transfers are encrypted; C PATerm data transfers are plain text.
- webMethods JI log files are not encrypted.
- The System Monitor can show plain text contents of client messages.

Requirements - Server Changes

Key and Certificate Files

Files containing the keys and certificate required by the SSL code can be found in *<JI_Install_dir>/config* directory as follows.

File	Contents
sslkeys	Contains the public/private key pair and certificate required by the server sockets in the Environment Manager and the JCluster.
ssl.cer	Certificate required by a client to authenticate the server. This is only required by the client if server authentication is enabled (via an API call).

For security reasons, the *sslkeys* file is not installed on a client-only installation. The files installed by the webMethods JI installer contain usable values, but for security reasons, it is recommended that these files be regenerated on the webMethods JI server and only the *ssl.cer* file be copied to the client machines if server authentication is required. The following script (UNIX) or batch (Windows) file can be found in the *<JI_Install_dir>/etc* directory to allow file generation:

Windows	`genkeys.bat`
UNIX	`genkeys.ksh`

genkeys Usage Notes

The *sslkeys* and *ssl.cer* files are generated by *genkeys* to your current directory. After generation, they must be copied to the *<JI_Install_dir>/config* directory if they are not already in that directory.

After generating new keys, webMethods JI must be re-started. If the client is doing server authentication, it must have access to the newly generated certificate file.

Environment Manager Configuration

A new property (`clientEncryption`) can be found in the Environment Manager configuration file (*envmgr.cfg*). This property has been added to prevent longer start-up times for webMethods JI installations where encryption is not required.

clientEncryption	Set to true to enable webMethods JI to handle client encryption.

JClient3 SSL-Related API Information

Methods

The following methods can be found in the JClient3 and EnvironmentManagerConnectionProperties and the classes:

- `setEncrypted()`
- `getEncrypted()`
- `setCipherSuite()`
- `getCipherSuite()`
- `getCipherSuites()`
- `setServerAuthentication()`
- `getCertificateFilePath()`

For information on JClient3, see Chapter 2 - "The Java Client Library Version 3 (JClient3)" on page 25 of the *webMethods JI Client Developer Guide*.

Exceptions

SSL-related exceptions are described here:

Exception	JClient3Exception
Message	Invalid cipher suite
Condition	If setCipherSuite is called with an invalid suite.

Exception	FileNotFoundException
Message	Unable to load keystore
Condition	If the SSL initialization code cannot find the keystore file.

Exception	IOException
Message	Various
Condition	If any problem occurs when trying to establish a secure connection to webMethods JI.

Exception	JClient3Exception
Message	Environment does not support encryption
Condition	If the client tries to connect to a webMethods JI environment that does not support encryption.

Enabling Encryption Using SSL

This section provides JClient3 code examples for:

- Enabling encryption
- Selecting a cipher suite
- Enabling server authentication.

Enabling Encryption

```
EnvironmentManagerConnection emc;
EnvironmentManagerConnectionProperties props;
try {
    props = new EnvironmentManagerConnectionProperties(host, port);
}
catch (Exception e) {
    System.out.println("Could not create EnvMgr props: " + e);
    return -1;
}
props.setEncrypted(true);
emc = new EnvironmentManagerConnection(props);
```

Then:

- ...get a service connection
- ...open the service
- ...invoke a method or methods
- ...close the service

Enabling Server Authentication

```
EnvironmentManagerConnection emc;
EnvironmentManagerConnectionProperties props;

try {
    props = new EnvironmentManagerConnectionProperties(host, port);
}
catch (Exception e) {
    System.out.println("Could not create EnvMgr props: " + e);
    return -1;
}
```

```
props.setEncrypted(true);
props.setServerAuthentication("<JI_Install_dir>/config/ssl.cer");

emc = new EnvironmentManagerConnection(props);
```

Then:

- ...get a service connection
- ...open the service
- ...invoke a method or methods
- ...close the service

Logging

Server Logging

The Environment Manager logs the fact that it is creating a secure listen socket when the log level is >= 1. The log messages are similar to the following:

```
Creating a secure listen socket...
Secure listen socket was created, port number = N
```

JClient3 Logging

If the debug level is at a level >= 3, information about whether or not encryption is enabled, the cipher suite, and the path to the certificate file are logged.

Java SSL Debugging

To see all SSL negotiations and data transfers, add "`-Djavax.net.debug=all`" when running the JClient3 client. The information is sent to `stdout`.

Chapter 17. Troubleshooting

Installation Issues

Temporary Directory

The webMethods JI installation program temporarily extracts data into a temporary directory. Approximately twice the disk space required for a JI installation must be available in the temporary directory. These files are removed from the temporary directory after installation is successfully completed.

See *webMethods JI Installation and Configuration Guide* for details about disk space requirements for a typical installation based on operating system.

UNIX

In UNIX, the installation program checks to see if enough space is available in the /*tmp* directory. If there is not enough space, an error message is printed to the screen instructing the user to clear space in the /*tmp* directory.

Windows

In Windows, the installation program extracts the file to the directory identified in the %TMP% environment variable. If there is not enough disk space on the temporary directory's disk partition to extract the data, the user is prompted to select another drive and directory for temporary extraction.

Runtime Issues

Using the *ea_start* and *ea_shutdown* executables

The EA_ENV Environment Variable

In order to use the *ea_start* and *ea_shutdown* executables, along with other administrative executables, the EA_ENV environment variable should be set to the *<JI_install_dir>/config* directory. If this environment variable is not set, *ea_start* and *ea_shutdown* must be started using the command line option `-c <configuration directory>`.

Also, the EA_ENV environment variable must be set when using the compatibility kit, and can be used by certain webMethods JI clients.

The environment.ccf File

The *environment.ccf* file must be located in the directory referenced by the `EA_ENV` environment variable. The *environment.ccf* file must contain settings for the `EA_HOME`, `DB_TCP_PORT`, and `EA_ENVMGR` parameters:

- **EA_HOME:** The root directory of your webMethods JI installation. (For example: *C:\JI*)
- **DB_TCP_PORT:** The TCP port number of your Resource Database server.
- **EA_ENVMGR:** The port number of the Environment Manager, used to determine if the Environment Manager is running.
- **START_TOMCAT:** Specifies whether the embedded Tomcat servlet engine is started and stopped, by default, with the `ea_start` and `ea_shutdown` commands. The default setting is `"true"`.

Additional parameters in the *environment.ccf* file are used by the Compatibility Kit portion of webMethods JI and clients developed with certain webMethods JI client libraries.

Also, the `EA_HOME`, `DB_TCP_PORT`, and `EA_ENVMGR` parameters can be defined as environment variables, if the *environment.ccf* file is not used.

DISPLAY Environment Variable

On UNIX, when using webMethods JI graphical interfaces or opening a terminal window, the $DISPLAY environment variable must be set to the appropriate IP address or host name of the machine on which your X-server software is running. This variable is set using the format `<host>:0.0` where `<host>` is the name or IP address of the host machine.

For example:

```
export DISPLAY=127.0.0.1:0.0
```

Localhost

The host *localhost* must be defined in your *hosts* file (or NIS/YP, if appropriate). This is the default host for webMethods JI. The loopback IP address generally is 127.0.0.1.

Client/Service Interaction Issues

Services Running after Client Disconnects

If a client is connected to a service and the service is executing its methods, the service continues to run even if the client gets killed. This also occurs if the client uses a non-blocking method invoke and the client gets killed or deliberately exits before the method completes. This allows the service to return to a known state rather than hanging at the point at which the client exited, but it could result in System Administrators noticing services that are running but do not have any clients connected to them.

Glossary

ACL	Access Control List
Actions	Used by webMethods JI Java services to navigate from one legacy screen to the next on the legacy application. An action includes all data that was input by the user during trail recording in MapMaker, along with the AID key or Action that caused the screen transition.
Action Key	A key sequence that performs an action in the legacy application. Action keys are valid for the Telnet protocol and are similar in concept to AID keys.
Applet	A program written in the Java programming language that is accessed from a Web browser.
Application	A Java program run as a stand-alone program.
API	API - Application Programming Interface. The library of C or Java functions callable from UNIX and Windows programs. Used to develop webMethods JI clients and services.
AID Key	The Attention Identifier Key (AID). A single key on the keyboard that, when pressed by the user, performs an action in the legacy application. Typical AID keys include the Enter and PF keys, although the legacy application may change their usage or use other AID keys. AID keys are valid for the TN3270 and TN5250 protocols.
Browser	A program that allows users to access information on a Web server. Also known as a Web browser.
C services	webMethods JI services written using the “C” programming language. Because these services were written for previous versions of webMethods JI (versions 2.x), these services are used with the “Compatibility Kit” and are sometimes called “legacy services” or “legacy C services”.

CGI
Common Gateway Interface. A standard method for external gateway programs to interface with Web servers.

Character Encoding
The format or encoding of a language-set character. Character encodings are usually 1, 2, 3, or 4 bytes. Unicode is an example of a 2-byte character encoding. Other examples are ASCII, EBCDIC, and UTF-8.

Character Mode
Character mode describes the functionality in webMethods JI that communicates with character-based applications over the Telnet protocol.

Client
In webMethods JI, client refers to one of two items:

- A Runtime version of an application developed in C, Java, or ActiveX that uses webMethods JI client APIs to communicate with webMethods JI services.
- Also refers to a third-party software application that interfaces with webMethods JI services and functions similarly to an application developed with a webMethods JI client API.

Client Functions
The C or Java functions used to allow clients to connect to services, execute service methods, and input and extract data.

Compatibility Kit
The portion of webMethods JI that is included to allow “C” services developed with webMethods JI version 2.x to interact with the modern Java-based webMethods JI environment.

Content Pane
A content pane, also called a panel, is a GUI component that acts as a container for various GUI components. A content pane is basically a window that other GUI objects, such as buttons and text fields, are placed on.

Cookie
A general mechanism used by Web servers to both store and retrieve information on the client side of the connection.

Custom Classes
Classes that are used to “extend” or customize service code that was generated in MapMaker.

Data Field A data field is an individual field on the legacy screen that is added to either a data template or a table template. Data fields are used in conjunction with output variables to extract data from the legacy screen.

Data Mapping Refers to the mapping of data in the flow of a method. Every point in a method where data is sent to or retrieved from an external source requires data mapping. Data mapping is defined in the Data Mapping Editor.

Data Stream The flow, or stream, of information between computer programs. Data on the data stream is represented using "character encoding" and is transferred using a mutually agreed upon protocol.

Data Template A data template is a logical representation of non-repeating data fields on the legacy screen. Data templates are defined in MapMaker and are used in conjunction with output variables to extract data from the legacy application. Similar to table template, used to define repeating data fields.

Data Typing Refers to the creation and definition of data types. Data types are defined and maintained in MapMaker's Business Entity Editor.

DBCS Double-Byte Character Set.

DLL Dynamically Linked Library. A library of function calls used in Windows environments.

DOM Document Object Model (aka "random access" protocol for XML) – an XML parser that converts the XML document into a collection of objects, which can then be manipulated in any way you choose.

DTD Document Type Definition for XML – an optional part of the XML document prolog that specifies the kinds of tags that can be included in an XML document and the valid arrangement of those tags.

EAServiceBean The interface between webMethods JI Java service code and the JService that manages the service in the webMethods JI server environment. The EAServiceBean can be extended or customized to change the interface if required.

ECS Extended Character Support.

EIS Enterprise Information System - an application providing information of critical importance to the day-to-day planning and/or operation of a business. EISs are generally run on larger platforms, such as mainframes or minicomputers. EISs provide the information infrastructure for an enterprise. Examples of EISs include enterprise resource planning systems, mainframe transaction processing systems, relational database management systems, and other legacy information systems.

Enterprise System A system involved in an organization's critical business processes. Typically, enterprise systems are large and complex, use database management systems (DBMSs), and run on mainframes or minicomputers.

Formatted Fields Fields on the legacy application that have special formatting characteristics. MapMaker identifies all such fields on the legacy screen and uses the field layout to match screens (unless the fields are disabled and Tags are used to identify screens).

GBBasic A Java package, *com.jacada.mapstudio.GBBasic*, that is included with webMethods JI and can be used to extend or customize Java service code that was generated in MapMaker.

GUI Graphical User Interface. An application that allows users to interface with computer programs in a graphical environment. In webMethods JI, MapMaker, the Configuration Manager, and the System Monitor are all Graphical User Interfaces.

HTML HyperText Markup Language, a format used to create Web documents.

Hos **t**A computer machine where applications reside. In webMethods JI, hosts can be the machine on which components of the webMethods JI environment are running, the machine on which the telnet, TN3270, or TN5250 server reside, or the machine on which the legacy applications reside.

IBE Internal Business Entity. Refers to data types that are used internally by the webMethods JI Service. A global variable may be defined of an IBE data type.

IDE Integrated Development Environment. Refers to a graphical development tool that uses standard GUI components to facilitate application development.

Jacada Integrator See webMethods JI**.**

Java An object-oriented, platform-independent programming language developed by Oracle.

Java services webMethods JI services developed using the Java programming language. Java services are generated from the MapMaker graphical development interface (GUI) and can be customized using the custom service classes included with webMethods JI.

JClient3 The Java Client Library version 3 is an improved version of the JClient, which allows webMethods JI clients to be developed using JDK 1.4.2_05 or newer.

JDBC Java DataBase Connectivity.

JDK Java Development Kit. The development environment for the Java programming language. Includes a Java Runtime Environment.

JRE Java Runtime Environment. The minimum environment required to run Java applications. This is a combination of a JVM along with the core classes and files required to run Java applications.

JVM Java Virtual Machine. A Java interpreter that converts Java code into executable code.

Legacy data Data residing on a mainframe platform.

Legacy host The machine or host on which legacy applications reside.

Map The logical representation of the screens, fields, data input and AID keys that make up the user interaction with a legacy application. Maps are created in MapMaker. Note that webMethods JI's use of the term Map is distinct from the java.util.Map that is included with Java.

MapMaker The graphical development interface provided with webMethods JI for the purpose of developing Java services. Used to record trails, maps, data and table templates, create methods and services, and then generate and optionally deploy the services into the webMethods JI server environment.

Methods Object-oriented entity of one or more functions. A collection of methods, initialization code, and events make up a service.

MLM Map-List-Map. Refers to an XBE data type used for communication with a client, such as a Java, C, or VB Client.

Multicasting A connectionless IP networking communication in which applications on the IP network broadcast information over a well-known socket.

Multithread-safe See thread-safe.

Multithreaded See threaded.

Offset Refers to the location of data on the legacy screen. The offset is determined using the following format: For an 80 column screen, the offset is (column # - 1) + 80 x (row # -1). For example, the first column of the second row is position 80:
(1 - 1) + 80 x (2-1).

Package A collection of java classes that are grouped together to form a logical combination of classes.

Presentation Space A representation of the communications between the legacy host to the webMethods JI environment, including the screen and field information from the legacy application.

Protocol Agent A software interface that governs the procedures used to exchange information between physically remote entities such as computer systems.The Protocol Agent governs the format of the messages, the generation of checking information, and the flow control, as well as the actions to take in the event of errors.

Proxy Server Special instances of Resource Servers that allow multicasting communication to take place over multiple sub-nets.

Resources The components of webMethods JI that are managed by the Resource Server. These components include Resource Databases and license files.

Resource Database A database that is used in the webMethods JI environment to store environment and service configuration, along with service code and maps. For the Compatibility Kit, the Resource Database is also used to store screen, field, and protocol agent definitions.

Resource Serve **r**Manages the communication between environment managers and the webMethods JI resources.

RMI Remote Method Invocation. A standard from Oracle that allows distributed Java objects to communicate with each other over TCP/IP networks.

Screen The screen, as contained in the data stream, that is coming from the legacy host.

Screen Mapping The process of marking the physical boundaries of, and defining the screen components found on, an external application screen. The components include tags, fields, areas, repeating areas, and repeating fields.

Service A collection of methods which answer requests from a client.

SOCKS SOCKS is a firewall proxy protocol.

System Monitor The tool used to monitor the webMethods JI System. It allows you to monitor, log, and view real-time activity for all or selected Environment Managers, JClusters and JServices, clients, and services.

Table Template A Table Template is a logical representation of the area on a legacy screen that contains repeating Data Fields. Table templates are defined in MapMaker and are used in conjunction with output variables to extract data from the legacy application. Similar to Data Template, used to define non-repeating Data Fields.

Tag A user-defined component of the host application screen that most often serves as a label for fields. You can define any static screen text as a tag. MapMaker can identify external application screens by the tags that are defined for them.

Telnet A TCP/IP-based terminal emulation protocol. Requires a Telnet server. Telnet is also used to describe the Character Mode functionality within webMethods JI.

Terminfo UNIX terminal information database. See the UNIX terminfo(4) man page.

Thread-safe Also multithread-safe (MT-safe). A description of a function or library that may be called in a threaded environment without any additional coding.

Thread A single flow of control within a process or address space. Programs using two or more threads are referred to as threaded or multi-threaded.

Threaded Also multithreaded. A form of multi-tasking that uses multiple independent execution threads.

TN3270 An implementation of the telnet protocol that is used to communicate between TCP/IP networks and IBM mainframe applications that use IBM 3270 terminals. A TN3270 server is required for connection from the TCP/IP network to the mainframe.

TN5250 An implementation of the telnet protocol that is used to communicate between TCP/IP networks and IBM AS400 applications that use IBM 5250 terminals. A TN5250 server is required for connection from the TCP/IP network to the AS400.

Trail The linear path of all screens encountered during navigation through a host application. MapMaker records trails during host application interaction.

Unicode A universal character code.

Web A network of computers based on the client-server model. The Web uses Web browsers to access information from a Web server. A Web can be a part of the World Wide Web or can be a part of a separate network, also know as an “Intranet”.

Web browser A program that allows users to access information on a Web server.

webMethods JI Consists of one or more Environment Managers, Resource Servers, resources including the Resource Database, and webMethods JI clients and services.

XBE eXternal Business Entity. Refers to data types that are used for the webMethods JI Service to send and receive data to and from an external source. Such external sources are Legacy Screens and Clients.

XML eXtensible Markup Language – a text-based markup language that is fast becoming the standard for data interchange on the web.

XSD XML Schema Definition. Specifies how to formally describe the elements in an XML document. One of the XBE data types supported by MapMaker is XML/XSD.

Index

A

Access Control List 551
ACL-controlled environments 562
adding a new ACL account record 558
adding a new ACL permission record 560
creating a new account 553
enabling access control 556
modifying an ACL account record 559
modifying an ACL permission record 561
ACL
...see Access Control List
Action
recovery 299
Action Key 611
Actions 164, 611
definition 147
AID Key 611
API 611
Applet 611
Application Programming Interface 611
ASP.NET Client 416
AutoContinue Option 303
AutoFetch Option 302

B

Block Mode 157
Breakpoints 367
enabling and disabling 377
Browser 611
Business Entity
...see also Data Type
confirm change 234
copy to other data types 228
create relations 231
definition 203
exporting 226
formatting 236
importing 223
managing 221
sample value 216
synchronizing 228
Business Entity Editor 210
confirm change dialog box 234
drag and drop 213
managing business entities 221
shortcut menus 215, 285

C

C Services 25, 611
CGI 612
Character Encoding 612
Character Mode 158, 612
Client 612
ASP.NET generation 416
functions 612
Interfaces 24
Libraries 24
Client to Host Communication
Overview 32
Compatibility Kit 612
Compatibility Mode 114
input 116
Method 279
method 115
output 116
step 117
Configuration Manager 28, 455
adding and editing a license key 507
Environment Managers
configuring 464, 478
locating 464
Host Connections 498
load balancing

configuring 480
Resource Database
Configuring 477
Resource Servers
Configuring 470
Resources
Configuring 474
Service Details 486
Service Masters 489
Starting 455
Conversion rules 252
Cookie 612
Copy To operation 228
create relations 231
synchronization 228
Create Relations operation 231
Custom Classes 427
Custom Code
Deleting and viewing deployed maps 502
Deploying EAServiceBean 419
Customizing MapMaker Classes 112

D

Data 181
Data Field 613
Data Mapping 265, 613
Data Mapping Editor 268
source to global variable 270
source to target via global variable 271
Data Mapping Editor 268
launching 269
Data Modeling 201
Business Entity Editor 210
data mapping 265
Data Mapping Editor 268
data typing 203
defining data structure relationships 249
Structure Relationship Editor 249
suggested workflow 272
verifying result of XPath expression 257
Data Stream 613
Data Templates 183, 613
Data Type
confirm change 234
conversion rules 252
create relations 231
defining data structure Relationships 249
definition 203
drag n drop 213, 250
external business entity (XBE) 207
legacy screen 209
Map-List-Map 208
XML/XSD 207
formatting 236
internal business entities 204
isArray 232
synchronization 228
Data Typing 203, 613
definition 204
Debug Cursor 367
defining properties 378
Debugger 363
accessing 365
breakpoints 367, 377
Debug cursor 367, 372
defining debug cursor properties 378
detailed workflow 364
general procedure 363
Host Screen Viewer 379
interface 365
menus 373
Method Flow toolbar 374
toolbars 373
TypeDataModel Viewer 383
Variables and Breakpoints toolbar 376
viewing and editing method data 383
viewing the connected host 379
views 365
watching values update 389
Deploy Service 419
as a web service 422
map and/or custom EAServiceBean 419

selecting the database 421
selecting the resource server 421
DLL 613
Document Object Model (DOM) 613
Document Type Definition (DTD) 613
Documentation 19
Viewing on-line 21
DOM 613
DoWhileCondition Steps 343
Drag and Drop 213, 250
DTD 613

E

EAServiceBean 614
Deleting and viewing deployed maps 502
Deploying 419
ECS 614
Environment
Managing 29
Monitoring 526
Overview 30
Environment Manager 27, 44
Actions in System Monitor 531
configuring 51, 68, 464, 478
Detail View in System Monitor 531
Events and Activities 526
locating 464
Properties in System Monitor 528
Starting 44
Stopping 50
environment.ccf 608
ErrorReportType IBE 205
Euro Symbol 138
Extensible Markup Language (XML) 619
External Business Entity (XBE) 207
legacy screen 209
Map-List-Map 208
XML/XSD 207

F

Fetch Steps 304
Fields
Formatted 614
ForCondition Steps 340
Formatted Fields 614
Formatting
input variable 239
output variable 239
Formatting Conventions 19
Formatting data types 236

G

GBBasic 614
Global Actions 153
defining 146
Global Variable 150, 206
default value 216
Graph Window 108
Groups 65
GUI 614

H

Host 139, 614
Host Connections 498
Host Screen Viewer 379
accessing 379
stepping through a screen-based step 380
HTML 614

I

IBE 615
...see Internal Business Entity
IDE 615
IfCondition Steps 338
IfElseCondition Steps 339

Input Variable
adding 280
enabling and disabling 282
formatting 239
setting properties 281
Input variable 280
Internal Business Entity (IBE) 204
ErrorReportType 205
exporting 242
importing 224
Invoke Steps 315, 316
Metadata properties 117
isArray 232

J

Java
runtime environment (JRE) 615
virtual machine (JVM) 615
Java Development Kit (JDK) 615
Java Runtime Environment (JRE) 615
Java Services 615
Java Virtual Machine (JVM) 615
JClient 615
JClient3 Steps 329
JClusters 27, 54
Events and Activities 534
Properties in System Monitor 535
JDBC 615
JDK 615
JRE 615
JService 27, 56
events and activities 543
properties in System Monitor 544
Trace Level 489
Default 497
Dynamically Changing 543
JTerm 571
configuring the remote JTerm applet 585
default keyboard mapping 573, 581
installing the remote JTerm feature . . 583, 584
launching a remote JTerm 591, 595
properties 572
remote JTerm errors 597
remote JTerm windows 582
JVM 615

K

Keyboard Mapping for JTerm windows . . 573, 581

L

Legacy Applications 27
Legacy Data 615
Legacy Host 615
Legacy Screen Data Type 209
creating 247
License
adding and editing a license key 507
checking statuses 510
compatibility 514
config.fromUser 513
config.licenseCheckTime 513
config.mailHost 513
config.mailUser 513
count 506
duplicates 513
ea_admin jclusters -e 511
ea_admin licenseStatus 510
ea_admin status 510
ea_start 510
envmgr.cfg 507
expiration email notification 511
expire flag 506
feature type 505
JI_CLIENT 74, 505
JI_DEMO 505
JI_HOST 75, 505
JI_MAPMAKER 75, 505
JI_SERVER 74, 505

JI_SERVICE 74, 505
key 504
license.txt 506
NO_LICENSE 505
priority 505
Licensing 74
Link Modes 355
Auto Arrange 356
branch 355
next 355
onFail 356
Swap Content 356
Load Balancing 26, 65
"fast" versus "accurate" 70
configurations 66, 68
groups 65
configuring 65, 480
Logging 411

M

Map 616
backward compatibility 115
convertor 115
Map Convertor 115
Map-List-Map (MLM) Data Type 208
MapMaker 28, 81, 616
Actions 164
Communications Connections 126
Configuring 96
Custom Classes 427
Custom EAServiceBean
Deploying 419
Data 181
Data Templates 183
Debugger 363
Generating Service Report 111
Global Actions 153
Global variable 150
Graph Window 108
Input variable 280
Maps
Deploying 419
Deleting deployed maps 500
Viewing deployed maps 499
Method Steps 288
Method variable 282
Output Variable 360
Services 391
Deploying 419
Deleting deployed maps 500
Viewing deployed maps 499
Generating 398
Generating Report 111
Starting 81
Table Templates 190
The Mapping Process 125
Trail Recording 137
MapPlayer
Starting 120
Maps
comparing screen images in 171
Deploying 419
Deleting deployed maps 500
Viewing deployed maps 499
Metadata
external 319
internal 319, 321
Method Steps 288
autofetch option 302
Break 291
conditional steps 333
Continue 290
debugging 363
DoWhileCondition 290, 343
EndIf 290
EndThread 291
External Invoke 289
Fetch 288, 304
ForCondition 290, 340
IfCondition 289, 338
IfElseCondition 290, 339
Internal Invoke 289, 315, 316
invoke steps 315

JClient3 . . . . . 289, 329
link modes 355
linking method steps 292
method step restrictions 359
operators 334
Perform 288, 308
recovery action 299
screen based steps 297
screen-based steps 297
Set 291, 345
set steps 345
Start 277, 293
Stop 277, 296
Thread 291, 345
ThreadJoin 291, 352
Traverse 288, 300
UserInteraction 288, 313
validating the current screen 298
WhileCondition 290, 342
Write 288, 311
Method Variable 282
adding 283, 286
deleting 287
duplicating 287
managing 286
Methods 616
Compatibility Mode 279
configuring method properties 278
debugging 363
definition 275
deleting 278
duplicating 278
Input variable 280
the Method tool bar 277
the Methods tab 275
Middleware 23
Monitoring 73
Multicast 53, 616
Multithreaded 616
Multithread-safe 616

O

Offset 616
onFail Processing 356
Output Variable 360, 361
adding 360
formatting 239
Overview 23

P

Package 616
Perform Steps 308
Presentation Space 616
Protocol Agent 617
Protocol Agent Terminal Window 568
components 570
JTerm 571
PATerm 570
server-side terminal window 570
creating
client-side 569
server-side 568
default keyboard mapping 573
JTerm windows 581
write-enabled terminal windows 581
protocols 26
Proxy Server 76, 617

R

Recovery action 299
Remote JTerm Windows 582
configuring the remote JTerm applet 585
errors 597
HTTP and MQ Gateway changes 598
installing
Java applet 584
stand-alone 583
launching
from browser 595

from command line 591
supported platforms 583
Request for Service 32
Resource Adapter Tab . . . . . 406 to 409
Resource Database 27, 63, 617
Configuring 477
Starting 63
Resource Manager 617
Resource Server 27, 57
Configuring 470
Proxy Server 76
Starting 58
Stopping 62
resource.cache file 53
Resources 474, 617
Configuring 474
RMI 617

S

Sample Value 216
Screen 617
comparison 171
mapping 617
Screen Images
comparing 171
Screen-based step
debugging 380
Server Environment 25
Service 617
Service Details 486
Service Masters 489
Services 25, 391
Deploying 419
Deleting deployed maps 500
Viewing deployed maps 499
Generating 398
Generating Service Report 111
Set Steps 345
Setting Condition Properties 334
Snapshot
comparison 171
SOCKS 617
SSL 601
Status bar 95
Structure Relationship Editor 249
drag and drop 250
XPath 254
Synchronization 228
System Monitor 28, 617
Environment Manager
Actions 531
Detail View 531
Events and Activities 526
Properties 528
JCluster
Events and Activities 534
Properties 535
JService
events and activities 543
JService Properties tab 544
Properties 524
Starting 515

T

Table Template 618
Table Templates 190
Tag 618
Tags 158
definition 147
Telnet 26, 128, 618
Terminfo 618
Thread 618
Thread Steps 345
Threaded 618
ThreadJoin Steps 352
Thread-safe 618
TN3270 . . . . . 26, 126, 618
Device Name Configuration 132
TN5250 . . . . . 26, 126, 618

Session Name Configuration 132
Trail 619
Trail Recording 137
Traverse step
debugging 382
Traverse Steps 300
TypeDataModel Viewer 383

U

Unicode 619
UserInteraction Steps 313

V

Validating the current screen 298
Variables
Global 150
Method 282
VT100/ 220/320 26

W

Web 619
Web Browser 619
webMethods JI environment 615, 619
WhileCondition Steps 342
Write Steps 311

X

XBE 619
...see External Business Entity
XML 619
XML/XSD Data Type 207
XPath 254
Evaluator 257
verifying result of XPath expression 257