Novell
Integration Manager™

6.0

September 29, 2006

CONNECT FOR LAWSON* SOFTWARE
USER'S GUIDE

www.novell.com

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2006 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.novell.com/company/legal/patents/ and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list.

Third-Party Trademarks

All third-party trademarks are the property of their respective owners.

Contents

About This Book

Purpose

This guide describes how to use the Novell Integration Manager™ Connect for Lawson Software. This product has design-time as well as runtime executables and uses J2EE Connector Architecture technology to provide integration capability.

Audience

This book is for developers and systems integrators who are planning to use Novell Integration Manager to develop services and components for Lawson Software.

Prerequisites

You should be familiar with the Integration Manager work environment and deployment options. You also should be familiar with Lawson Software. Familiarity with Java Connector Architecture is helpful but not required.

Software Versions

This guide assumes that you are using Novell Integration Manager (Enterprise or Professional) version 5.2 (or higher) and that you are deploying your applications to a J2EE* Version 1.3 application server or equivalent.

The Connect for Lawson Software is designed to work with Lawson Software version 7.

Additional Documentation

For the complete set of Novell exteNd documentation, see the Novell Documentation Web site (http://www.novell.com/documentation-index/index.jsp).

About the Product Name Change

In version 6.0. we've changed the name of exteNd Composer to Novell Integration Manager. In some places in the user interface, and in Integration Manager file and directory names, you will still see the name "exteNd Composer" or "Composer".

1 Welcome to the Novell Integration Manager Connect for Lawson Software

Welcome to the *Novell Integration Manager Connect for Lawson Software User's Guide*. This Guide is a companion to the *Novell Integration Manager User's Guide*, which details how to use all the features of Integration Manager except for the Connect Component Editors. You should be familiar with the *Integration Manager User's Guide* before using this Guide.

Novell Integration Manager™ provides separate Component Editors for each Connect, including the Connect for Lawson Software. The special features of each component editor are described in separate Guides, like this one.

If you have been using Integration Manager and are familiar with the core component editor (the XML Map Component Editor), then this Guide should be enough to get you started with the Lawson Software Component Editor.

NOTE: To be successful with this Component Editor, you must be familiar with Lawson Software and basic XML integration concepts.

About Novell Integration Manager

Novell Integration Manager is an XML integration-broker. It encompasses a set of design tools for building XML integration applications and Web services, plus a runtime engine that enables execution and administration of the services that you build. The applications and services that you build with Integration Manager can be deployed to any popular J2EE application server or servlet container. Supported servers include JBoss, IBM WebSphere and BEA WebLogic in addition to the Novell exteNd application server. Apache Jakarta Tomcat is also supported. Check the Novell web site for latest platform-support information.

At the core of Integration Manager is a robust XML transformation engine capable of performing a wide range of data transformations, including joining of multiple documents, decomposition of documents, and creation of entirely new documents. The underlying enabling technologies include XSLT, XPath, ECMAScript, and Java. The Integration Manager design environment offers a rich, intuitive graphical user interface, making it possible for you to specify XML transformations and mappings visually, using wizards, dialogs, and drag-and-drop gestures. You never have to write raw XSL or Java code.

Integration Manager supports numerous kinds of data-source connectivity, through individual adapters called Connects. Using the functionality exposed in the various Connects, you can design EAI applications and Web services that pull data in from or push data out to different kinds of back-end systems, using a variety of transport protocols and technologies, ranging from 3270 and 5250 terminal data streams to Telnet, HP3000, Unisys T27 (and UTS), Tandem, and Data General, in addition to HTML screen-scraping, JMS messaging, and CICS RPC transactions. You can also take advantage of JDBC, LDAP, and other mechanisms to reach back-end data repositories and systems that might or might not natively understand XML. Integration Manager Connects allow you to connect to these systems inobtrusively, so as to marshall non-XML data into XML form or vice versa without any need to modify host-system setups or code.
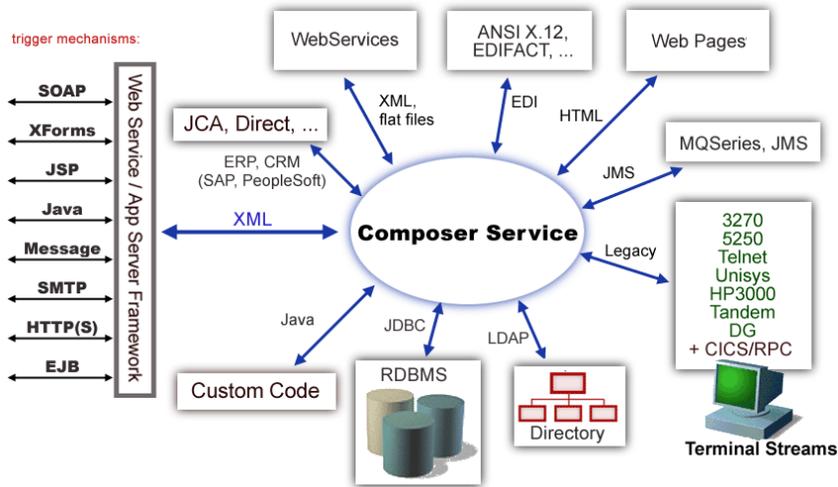
In addition to legacy data-stream and protocol-specific Connects, Integration Manager has Connects for ERP and CRM systems, including Baan, PeopleSoft, SAP, Siebel, JD Edwards, and Oracle Financials. As with other Connect solutions, the ERP and CRM Connects are fully integrated into Integration Manager's design-time environment so that you can use intuitive visual tools to create powerful custom integration solutions, eliminating the need to write Java code or edit raw XML or schemas by hand. You can also test the components that you build against live Lawson Software connections, using the animation facility (step-through debugger) of the design environment. As part of the design and debug process, your Lawson-aware components can call other Integration Manager Components (such as XML Map Components, JDBC Components, etc.) and make use of any of the core actions that Integration Manager defines (such as Map, Function, Log, and other actions).

About the Integration Manager Connect for Lawson Software

The Novell Integration Manager Connect for Lawson Software allows you to leverage the Lawson Application Gateway Service (AGS) interface to build powerful XML-based integration solutions.

The Integration Manager Connect for Lawson Software enables you to reuse your existing Lawson business functions with other applications—the key to building a successful e-business or integrated enterprise. The Integration Manager Connect enables you to incorporate Lawson business functions into new application initiatives.

You do not need to manually generate or install Lawson Software XML schemas in order to use the Integration Manager Connect for Lawson Software. The Connect will generate schemas for you, as needed, automatically.

You also don't have to take any steps to preinstall RARs (resource adapter archives, defined by the Java Connector Architecture) on the target application server ahead of time. Integration Manager handles RAR deployment for you automatically when you deploy any Integration Manager-built service that utilizes the Integration Manager Connect for Lawson Software.

NOTE: Some one-time setup and configuration steps *are* required in order to use the Integration Manager Connect for Lawson Software. These steps are described in Chapter 2, "Getting Started With the Integration Manager Connect for Lawson Software".

About J2EE Connector Architecture

The J2EE Connector Architecture (JCA) defines a standard architecture for connecting elements of the J2EE platform to a heterogeneous Enterprise Information System (EIS). Examples of EIS components include Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM), Supply Chain Management (SCM), mainframe transaction processing, database systems, and legacy applications that are not written in the Java programming language. By defining a common set of scalable, secure, transactional mechanisms reachable via a standard set of APIs, J2EE Connector Architecture enables the integration of an EIS with an application server and enterprise applications.

The J2EE Connector Architecture permits an EIS vendor to provide a standard resource adapter for its EIS. The resource adapter plugs into an application server, providing connectivity to an EIS, and integrating it with the rest of the enterprise. If an application server vendor has extended its system to support J2EE Connector Architecture, it is assured of seamless connectivity to multiple Enterprise Information Systems.

Before J2EE Connector Architecture, most EIS vendors offered vendor-specific architectures to provide connectivity between applications and their software. Each program interacting with an EIS needed to be hand-tooled by someone with a detailed knowledge of the target EIS. Custom software to provide connectivity across multiple systems was time consuming to develop, debug, and maintain.

By providing a standard set of APIs and contracts for managing connectivity, exposing EIS APIs, and using application-server services (like transaction control and connection pooling), J2EE Connector Architecture greatly reduces the need for custom programming. Developers can focus on business logic rather than connectivity and transaction-related logic and a variety of "plumbing issues."

How J2EE Connector Architecture Works

The "major participants" in J2EE Connector Architecture include these components:

- Application server
- Resource adapter (RAR)
- Application

The application server is not strictly required. Certain services like connection pooling and transaction control will not be available in a "server" that is just a servlet container. But J2EE Connector Architecture resource adapters can still operate.

The RAR represents the interests of the underlying EIS.

The *application* interacts with the *resource adapter* using what J2EE Connector Architecture calls standard contracts. Standard contracts define what interactions are to take place and how they are exposed. The contract between the application and the resource adapter is called the Common Client Interface (CCI). The resource adapter, in turn, interacts with the application server under the Service Provider Interface (SPI), which defines how the management of resource adapter interactions occurs. The aspects of this include:

- Connectivity management
- Transaction demarcation
- Event listening (listeners can receive notification of significant events; for example, a connection failure)
- Pooling of connections and other resources

In the normal course of events, the application uses a *naming service* to locate the appropriate resource adapter. The application server supplies the naming service, and so it recognizes that a request is being made to locate a resource adapter. In such a case, the application server interposes a resource-adapter-supplied intermediate object that interacts between the resource adapter and the application server. Through this intermediating object, the application server manages the items within the SPI contract below the awareness of the application.

For more information about J2EE Connector Architecture, visit http://java.sun.com/j2ee/connector/.

About iWay Technology

Novell Integration Manager uses licensed J2EE Connector Architecture adapter technology from iWay Software* (a division of Information Builders, Inc.) to mediate EIS interactions in the Integration Manager Connect for Lawson Software. A leader in the J2EE Connector Architecture technology space, iWay Software provides resource adapters and connectivity solutions across a wide array of EIS and other systems.

For more information about iWay, see http://www.iwaysoftware.com.

What Kinds of Applications Can You Build Using the Integration Manager Connect for Lawson?

With Integration Manager Connect for Lawson Software, you can build any kind of Web service or integration application that needs to push data into or pull data from a Lawson Software-based data store using XML as the interchange format. Your integration application can be deployed to a J2EE application server and run as a public web service, or it can be used in "behind the firewall" scenarios. It can be triggered by a servlet, JSP, EJB, e-mail, timer, file arrival, JMS message arrival, or any of the supported Integration Manager trigger types. It can also run standalone or as part of a workflow built using Integration Manager Enterprise Edition's Process Manager. (For more information about deployment options, see "Deploying Your Project," in the *Novell Integration Manager User's Guide*.)
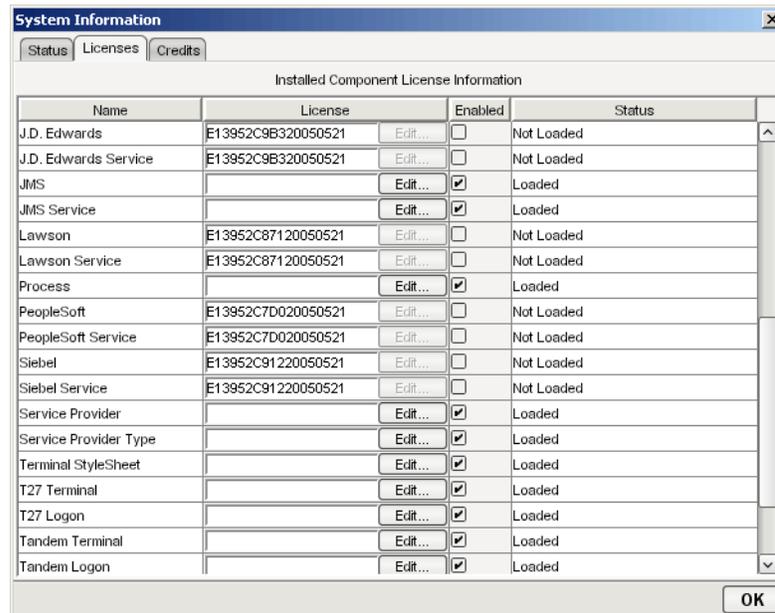
2 Getting Started With the Integration Manager Connect for Lawson Software

This section describes how to set up and configure the Integration Manager™ Connect for Lawson Software. Before you can use the Integration Manager Connect for Lawson Software you must update the licenses for your Integration Manager installation. You need to activate your design-time *and* runtime copies of the Novell Integration Manager Connect for Lawson Software.

Updating the Design-time Software License

1   Obtain a valid license string from your Novell representative.

   NOTE: You use the same license string to activate the design-time and runtime versions of Integration Manager Connect for Lawson Software.

2   Launch Integration Manager.

3   Select **About Integration Manager** from the **Help** menu. The About Integration Manager dialog box is displayed.

4   Select **System**. The System Information dialog box is displayed.

5   Click on the **Licenses** tab, located in the upper left corner of the System Information dialog box. The license information for your Integration Manager installation is displayed.

6   Scroll down until you see the row for Lawson.

   NOTE: The Connect ships with an Evaluation license string which may be used for 90 days.

7   To use the evaluation license string, select the check box in the **Enabled** column and skip to Step 10. To enter a different license string, select the check box in the **Enabled** column. The **Edit** button in the Lawson row is enabled.

8   Select **Edit** in the Lawson row. The **Edit License** dialog box is displayed.
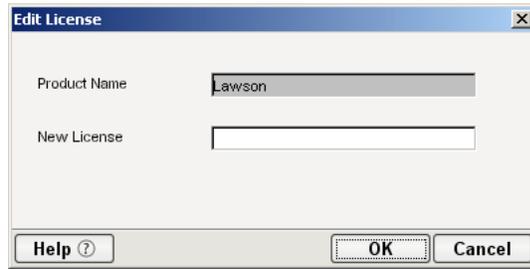
9   Type the license string for the Integration Manager Connect for Lawson in the **New License** field.

10   Scroll down until you see the row for Lawson Service.

11   Repeat Step 7 through Step 9 for the Lawson Service row.

12   Exit out of all dialogs by selecting **OK**.

Updating the Runtime License

NOTE: If you are using the Tomcat application server, copy the **jca10apis.jar** from the **\NIM6\Server Resources\Tomcat** directory (Windows) or **nim/Server Resources/Tomcat** directory (Linux) to the **Tomcat Composer\lib** and **shared\lib** directories before attempting to update the runtime license.

1   Make sure that the Novell Integration Manager Enterprise Server is installed.

2   Start the application server. The Novell Integration Manager Enterprise Server starts automatically when you start the application server.

3   Open a browser window and navigate to the Novell Integration Manager main administrative console. Typically, this is at:

    `http://localhost/exteNdComposer`

4   In the upper left corner of the console window, click the **exteNd Composer** logo immediately above the words "Server Console," as shown in the following illustration.

5   In the content area of the main frame, near the bottom, click **Licenses**. The Manage Licenses window is displayed. This window shows the license status of every Integration Manager Connect. See the following illustration.
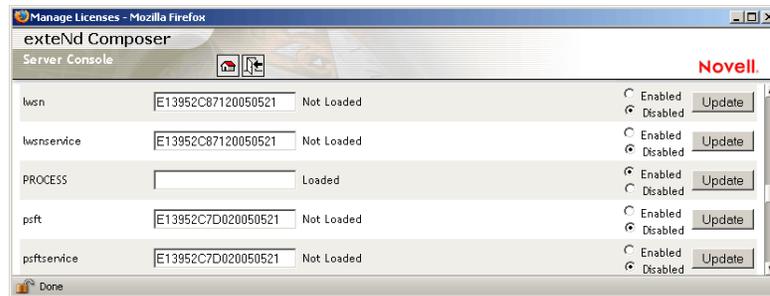
6 Scroll down to the entry labeled "lwsn".

NOTE: The Connect ships with Evaluation license strings which may be used for 90 days.

7 Type a license string in the text field in the "lwsn" row; then select **Enabled** in the "lwsn" row. To use the evaluation license string, just select **Enabled.**

8 Select **Update** in the "lwsn" row.

9 Scroll down to the entry labeled "lwsnservice".

10 Type a license string in the text field in the "lwsnservice" row; then select **Enabled** in the "lwsnservice" row. To use the evaluation license string, just select **Enabled.**

11 Select **Update** in the "lwsnservice" row.

3 Creating a Lawson Software Component

To create a Component that utilizes the Integration Manager™ Connect for Lawson Software, you need to do three things:

- Create a Connection Resource (to allow your component to connect to a Lawson Software system)
- Create any necessary XML Templates
- Create the Component itself (containing your business logic)

Each of these processes is discussed in detail in this section.

Creating a Connection Resource

Before creating a component that interacts with a Lawson Software system, you need to create a Connection Resource, which is a lightweight Integration Manager object (xObject) that encapsulates basic connection information (parameter values) associated with a connection to a back-end system.

In addition to a connection resource, a Lawson Software Component requires that you have already created XML templates so that you have sample input and output documents for use in designing your component. For more information, see "Creating an XML Template" in the *Novell Integration Manager User's Guide.*

If your component design calls for any other resources, such as custom scripts, XSL, XSD, etc., you should create these before creating the Lawson Software Component. For more information, see *"Creating Custom Scripts"* in the *Novell Integration Manager User's Guide*.

Types of Connection Resources

You create different types of connection resource depending on the type of interaction with the Lawson system that is desired. If your application needs to initiate Lawson business events, you need to create a Lawson Connection resource. If your application needs to process data when a business event occurs within the Lawson system, you need to create a Lawson Service File Connection resource.

About Constant-Driven and Expression-Driven Connection Parameters

You can specify Connection parameter values in one of two ways: as Constants or as Expressions. A constant-driven parameter uses the value you type in the Connection dialog every time the Connection is used. An expression-driven parameter allows you to set the value using a programmatic expression, which can result in a different value each time the connection is used at runtime. This allows the Connection's behavior to be flexible and vary based on runtime conditions each time it is used.

For instance, one very simple use of an expression-driven parameter in a Connection would be to define the User ID and Password as PROJECT Variables (e.g. PROJECT.XPATH("USERCONFIG/MyDeployUser"). This way when you deploy the project, you can update the PROJECT Variables in the Deployment Wizard to values appropriate for the final deployment environment. At the other extreme, you could have a custom script that queries a Java business object in the Application Server to determine what User ID and Password to use.

➢ **To switch a parameter from Constant-driven to Expression driven:**

1 Click the right mouse button in the parameter field you are interested in changing.

**2** Select **Expression** from the context menu and the editor button will appear or become enabled.

3 Click on the button and then create an expression that evaluates to a valid parameter value at runtime. (Strings should be wrapped in double-quotes.)

Creating a Lawson Connection Resource

➢ **To Create a Lawson Software Connection Resource:**

1 Select **File > New > xObject**. The New xObject dialog box is displayed.

2 Select the **Resource** tab.

3 Double-click on **Connection**. The "Create a New Connection Resource" Wizard is displayed.
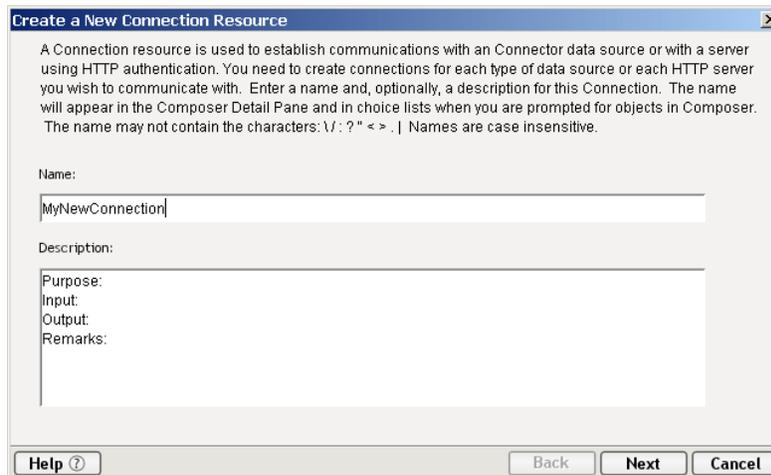
4 Type a **Name** for the connection object.

5 Optionally, type **Description** text.

6 Select **Next**. A connection parameters panel is displayed.

Create a New Connection Resource

Specify Lawson specific parameters here to make connection to Lawson EIS.

Connection Type: Lawson Connection
Host or IP Adress: 164.99.24.112
Lawson Port: 8080
CGI Root: /ibilawson/cgi-lawson
User ID: Admin
Password: *****
HTTPS: ☐
Maximum Pool Size: 10

Test
☐ Default

Help ?   < Back   Finish   Cancel
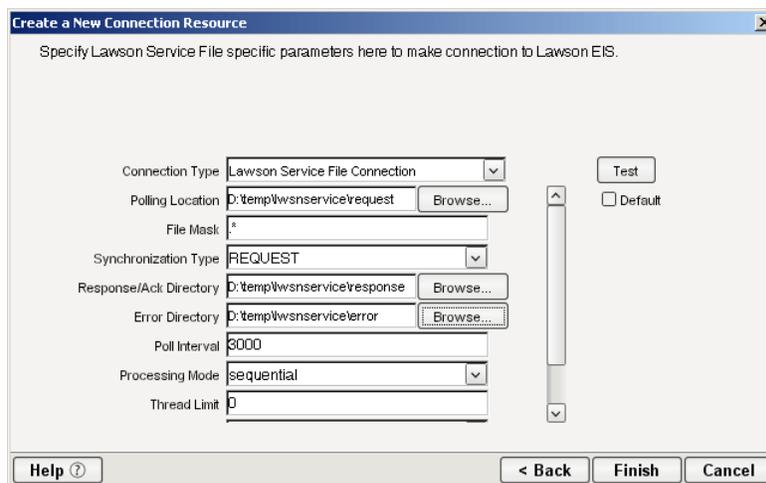
7   Select **Lawson Connection** from the **Connection Type** list.

8   Enter the host name or IP address of the Lawson Software server in the **Host or IP Address** field.

9   Enter the port number used in your Lawson Software configuration (e.g., 8080) in the **Lawson Port** field.

10   Enter the root directory name for the Lawson CGI (e.g., /ibilawson/cgi-lawson) in the **CGI Root** field.

11   Enter a valid User ID to sign on to the selected Lawson Software server in the **User ID** field.

12   Enter the password for the user specified in the **Password** field.

13   If the Lawson environment is set up to use HTTPS communications, select **HTTPS**.

14   Enter the desired maximum pool size in the **Maximum Pool Size** field. This is a JCA setting that specifies the maximum number of connections to be allowed in the connection pool (not applicable for non-managed environments).

15   If desired, select **Test** to see if your connection parameters and network environment allow you to create a live connection. Integration Manager displays a message indicating the success or failure of the test. The test connection is discarded immediately after the test. *You can continue working with the connection resource, even if the connection fails.*

     NOTE: This test does not test the connection pool.

16   Select **OK**. The newly-created resource connection object appears in the Integration Manager Connection Resource detail pane.
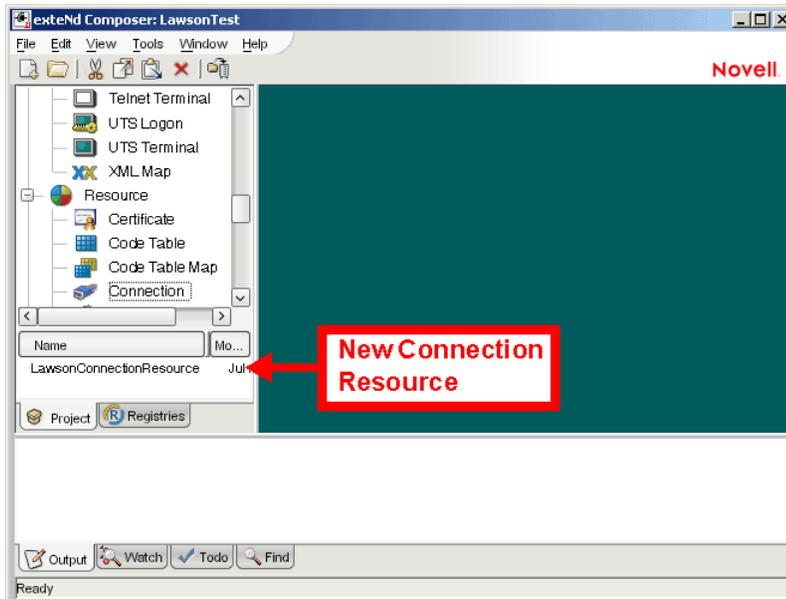
Creating a File Service Connection Resource

➢ **To Create a Lawson File Service Connection Resource:**

1 Select **File** > **New** > **xObject**. The New xObject dialog box is displayed.

2 Select the **Resource** tab.

3 Double-click on **Connection**. The "Create a New Connection Resource" wizard is displayed.

4 Type a name for the connection object in the **Name** field.

5 Optionally, type a description of the connection object in the **Description** field.

6 Select **Next**. A connection parameters panel is displayed.

7 Select **Lawson Service File Connection** from the **Connection Type** list.

8 Use the **Browse** button next to the **Polling Location** field to select the target file system location for the Lawson XML file.

9 In the **File Mask** field type the file name to be used for the output file generated as a result of this operation. The default is ".*".

10 Select REQUEST, REQUEST_RESPONSE, or REQUEST_ACK from the **Synchronization Type** list. Select the option that is correct for the system with which you are integrating (e.g., if your system is set up to process a response, select REQUEST_RESPONSE). The Synchronization Type selected also determines the tabs that are displayed in the Service Request Native Environment pane (e.g., if REQUEST_RESPONSE is selected, both Request and Response tabs are displayed).

11 Use the **Browse** button next to the **Response/Ack Directory** field to select the directory to which responses or acknowledgements from your application will be written.

12 Use the **Browse** button next to the **Error Directory** field to select the directory to which error information will be written.

13 In the **Poll Interval** field, type the interval (in milliseconds) in which the Polling Location is checked for input. The default is 3000 (3 seconds).

14 Select Threaded or Sequential from the **Processing Mode** list. Threaded indicates processing of multiple requests simultaneously. Sequential indicates serial processing of requests.

15 If Threaded processing mode is selected in the Processing Mode list, use the **Thread Limit** field to specify the maximum number of requests that can be processed simultaneously.

16 From the **Metadata Connection** list, select a connection resource from the list of defined connection resources. This connection is used for acquiring metadata from the system.

17 Type the desired maximum pool size in the **Maximum Pool Size** field. This is a JCA setting that specifies the maximum number of connections to be allowed in the connection pool (not applicable for non-managed environments).

XML Templates for Lawson Software Components

Before creating a Lawson Software Component, you should create any XML templates that might be needed. XML templates are used to tell Integration Manager which XML sample documents (e.g., input, output, scratch pad, fault, XSD, or DTD) to use. Once you've specified the XML templates, you can create a component that uses the sample documents to represent the inputs and outputs processed by your component. For more information about XML templates, see "Creating a New XML Template" in the *Novell Integration Manager User's Guide*.

Creating Lawson Software Components

As part of the process of creating a Lawson Software Component, you can select an existing Lawson Software connection resource, or you can create a new one. If you create the connection beforehand, it is available for use by any Lawson Software Components in the current project. *If you have not already created at least one Lawson Software connection resource in the current project, you will be prompted to do so when you try to create a Lawson Software Componen*t.

NOTE: You can still create a Component without first creating a Connection Resource, but you won't be able to use any debug features that depend on a live connection.

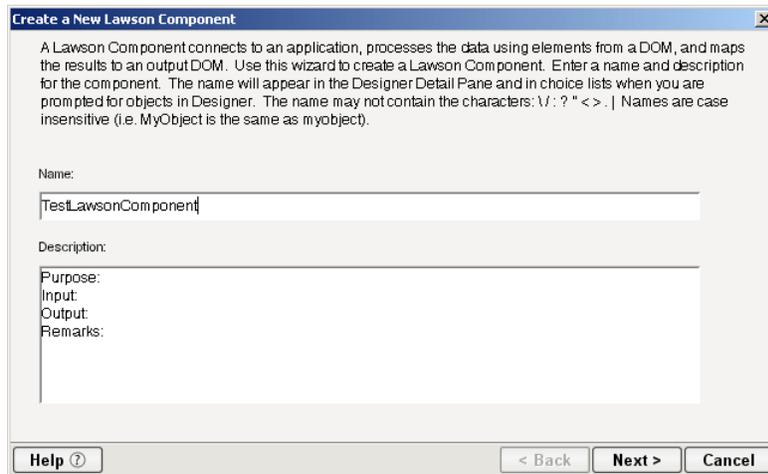➢ **To Create a New Lawson Software Component:**

1 Select **File>New>xObject**. The New xObject dialog box is displayed.

NOTE: Alternatively, under **Component** in the Integration Manager Navigator pane (Explorer view), you can highlight **Lawson**, click the right mouse button, then select **New**.
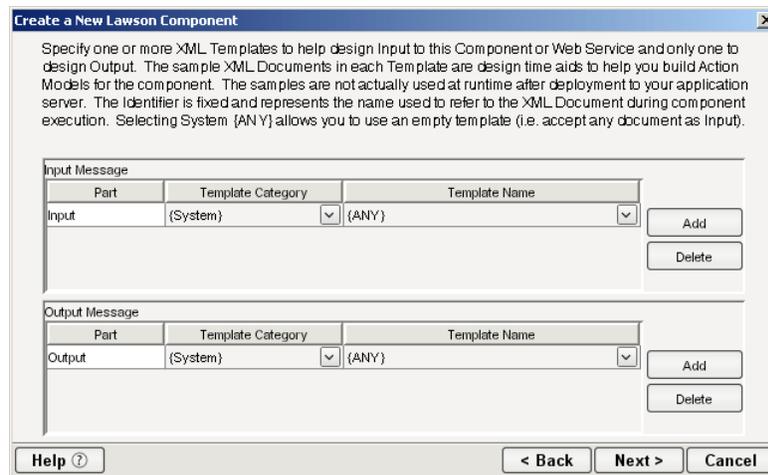
2 Select the **Component** tab.

3 Double-click on "Lawson."

If you have not previously defined a connection resource, you are prompted to do so now (see).

If you have already defined a connection resource, the "Create a New Lawson Component" Wizard is displayed.
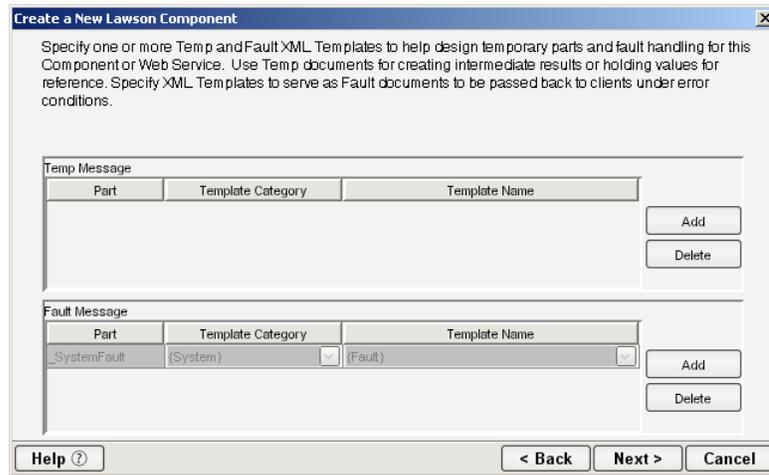

4 Type a **Name** for the new Lawson Software Component.

5 Optionally, type **Description** text.

6 Select **Next**. The XML Input/Output Message Property Info panel of the New Lawson Software Component Wizard is displayed.
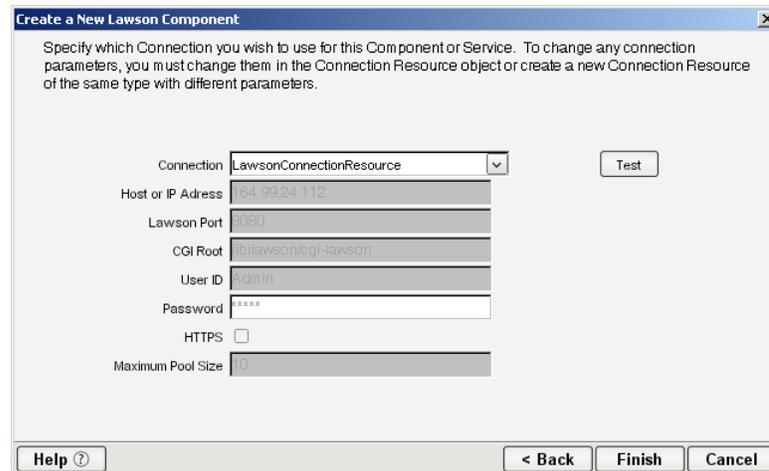

7 Specify the Input and Output templates as follows:

- Type a name for the template under **Part** if you wish the name to appear in the DOM as something other than "Input"

- Select a **Template Category** if it is different than the default category.

- Select a **Template Name** from the list of XML templates in the selected **Template Category**.

- To add additional input XML templates, click **Add**, then choose a **Template Category** and **Template Name** for each.

- To remove an input XML template, select an entry and click **Delete**.

8 Select an XML template for use as an Output DOM using the procedure described in the previous step.

NOTE: You can specify an input or output XML template that contains no structure by selecting {System}{ANY} as the Input or Output template. For more information, see "Creating an Output Document without Using a Template" in the *Novell Integration Manager User's Guide*.

9 Select **Next**. The Temp and Fault XML Template panel is displayed.

10 If desired, specify a template to be used as a scratch pad under the "Temp Message" pane of the dialog box. This can be useful if you need a place to hold values that will be used temporarily during the execution of your component. Select a **Template Category** if it is different than the default category. Then select a **Template Name** from the list of XML templates in the selected Template Category.

11 Under the "Fault Message" pane, select an XML template to be used to pass back to clients when an error condition occurs.

12 To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each. Repeat as many times as desired. To remove an input XML template, select an entry and click **Delete**.

13 Select **Next**. The Connection Info panel of the Create a New Lawson Component wizard is displayed.


14 Select a Lawson Software connection from the **Connection** list. The Connection list displays the names of the Lawson Software connection resources that have been defined in this project.

15 Select **Finish**. The component is created and the Lawson Software Component Editor is displayed.

About the Lawson Software Component Editor Window

The Lawson Software Component Editor includes all of the functionality of the XML Map Component Editor. It contains mapping panes for Input and Output XML documents, as well as an Action Model.

The Lawson Software Component Editor also includes a Native Environment pane, which appears as a grey pane until you create a Lawson Software Request action, at which time it will show a tabs for Request and Response panes with corresponding XML trees.
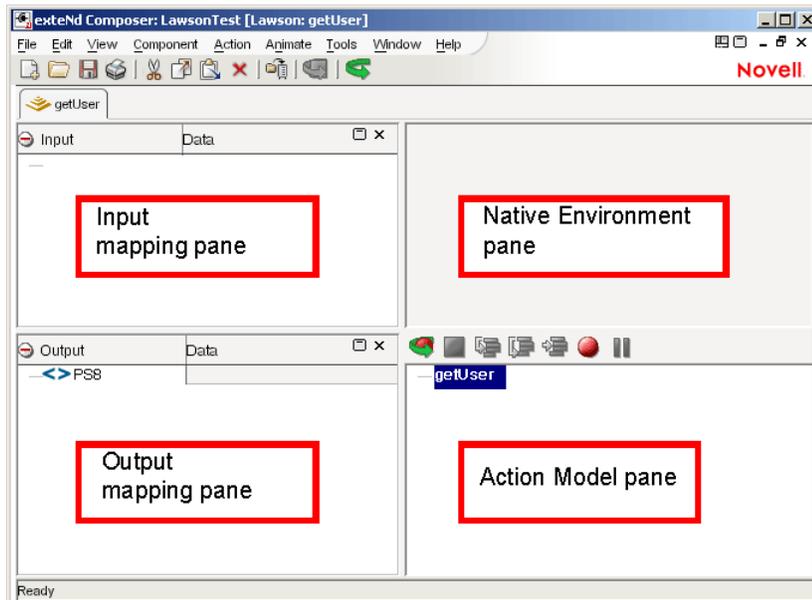
The Native Environment pane shown in the illustration is blank, because no Lawson Software Request action has been created.

Creating Actions in the Component Editor

You can create all the normal Integration Manager actions in the Action Model of your Lawson Software component (e.g., XML Map, Function, Log, Send Mail). In addition, you can create a Lawson Software Request action.

The Lawson Software Request action communicates requests (XML request documents) to your Lawson Software system and fetches responses back from the same system.

The Integration Manager GUI for creating a Lawson Software Request action allows you to

◆ Specify the type of Lawson Software business function that you want to invoke

◆ Automatically generate XML schemas (request and response) for the particular business function that you want to use

◆ Use the generated schemas to fine-tune the exact structure of the actual request document you want to use (a request document that conforms to the schema but reflects your own "setup" choices and initialization parameters, etc.)

◆ Automatically generate XML request and response documents that conform to the schemas and choices that you've made

You will see how this works in the following procedure.

➢ To Create a Lawson Software Request Action

1   Right-click in the Action Model at the location in which you want the new action to appear, then select **New Action > Request**.
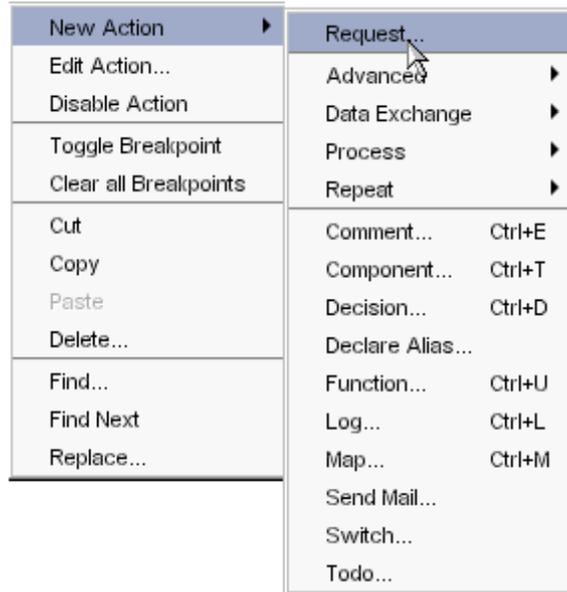
The Function pane of the Request dialog box is displayed. This page displays the Lawson components for the system to which you are connected.
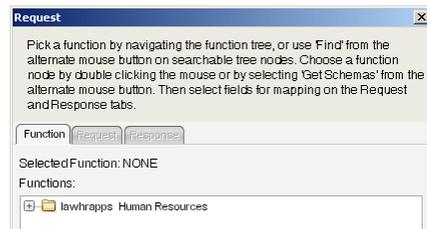

2   Click on the plus (+) sign to the left of the topmost node in the tree view to expand the tree view.

NOTE: If no nodes are displayed on the Function page, there may be an incorrect parameter in your connection resource. Check the parameters needed for connecting to your Lawson Software server, then update your connection resource if necessary (see).
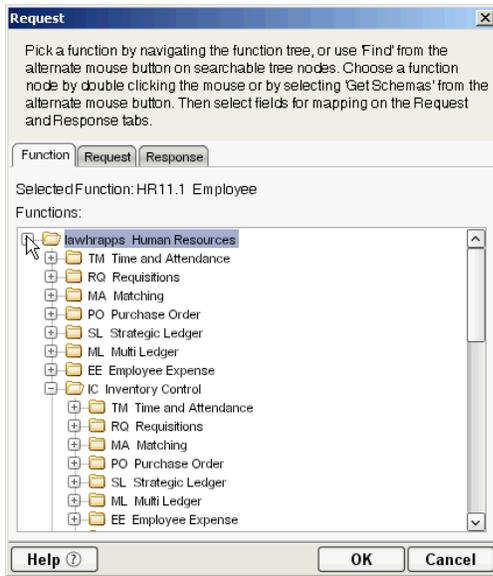
You will see a list of product names. Position the mouse pointer over a node to receive ToolTip information about the node. ToolTips display the following information, depending on the node type:

- Folder and Searchable Folder nodes: "type - Service"
- Function nodes: "type - Message"

3 Click on the plus sign to the left of a product name to see the business functions for that product name.

4 Right-click on the desired business function, then select **Get Schemas** from the context menu.
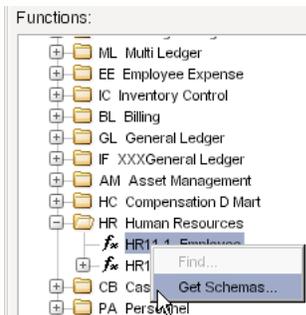
After a few seconds, the Request and Response tabs of the Request dialog box are enabled.

**5** Select the **Request** tab. The Request pane is displayed.
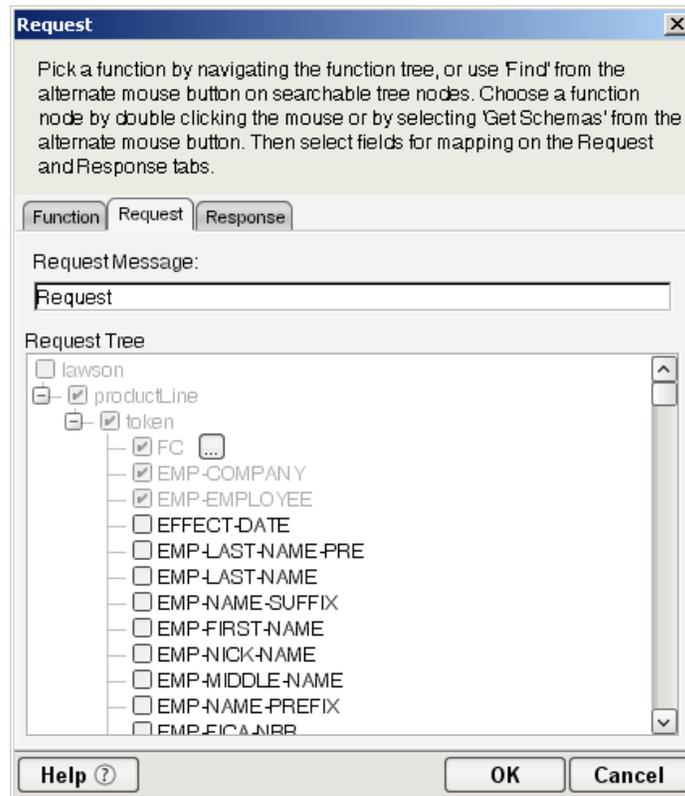
6 Type a name for the request message in the **Request Message** field, or accept the default as shown in the preceding illustration.

7 Expand the nodes of the Request Tree by clicking on the plus signs to the left of the nodes.

8 Select the elements that you want to include in the Request document by selecting or deselecting the check boxes in the node tree. Some nodes are greyed out (disabled) while others are not. This is because the document schema determines the nodes that you can edit.

Integration Manager strictly enforces schema rules. In other words, Integration Manager will not let you specify an invalid request document. Items that you *should not edit* are disabled, and items that you can edit have context-menus that non-customizable nodes do not have. For example:

- ◆ *Mandatory* elements and attributes are marked *selected* and *disabled*.
- ◆ All child nodes are left unselected and disabled if the parent node is not selected.
- ◆ A parent element, once selected, will autoselect mandatory elements and attributes of its children.
- ◆ When a parent node is selected, you will need to select *optional* elements *manually*.

ToolTips are an important aid in using the Request and Response panes, because they display document-structure rules from the request/response document schema. Familiarize yourself with the ToolTip feature by letting the mouse hover over various nodes, as shown in the following illustration.
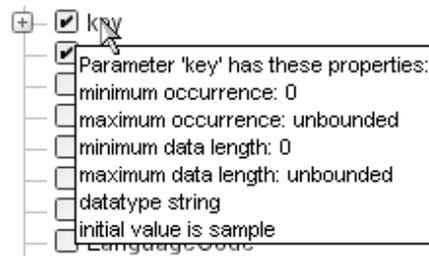
NOTE: The ToolTips that appear in this tree are also available in the Native Environment pane.

9 If a node has **Maximum Occurrence** property of greater than one, a command called "Create new..." is available by right-clicking on the node. Selecting that command brings up the following dialog box:
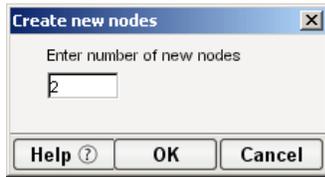

Enter the number of additional instances of this node-type that you want to insert in the document, then click **OK**. New nodes are added to the document structure.

CAUTION: There is no Undo for this operation.

10 If an enumeration button (…) is displayed to the right of a given node, you can click the button to choose from the list of appropriate (schema-allowed) values for that enumeration.
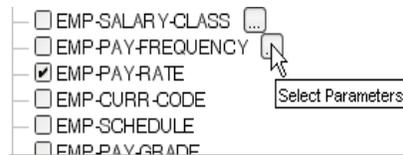
A common situation is that an enumeration offers choices of Update, Insert, or Delete. An example of this is shown in the following illustration, which shows the dialog that appears when you click an Enumeration Button.
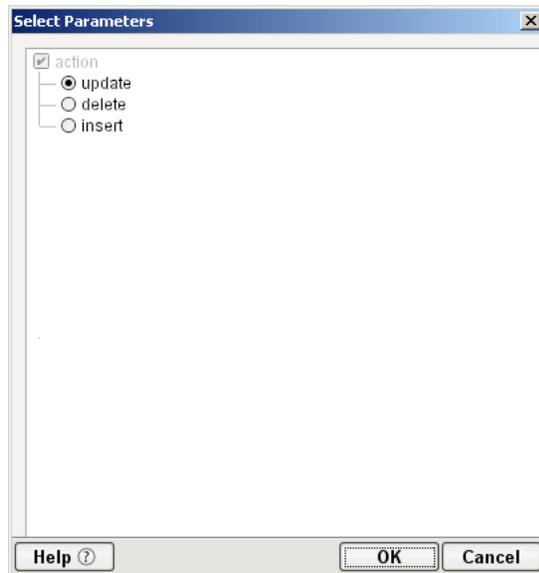
11 Visit all of the nodes in the request tree that are of interest to you, and use the Integration Manager user interface features described in the preceding steps to customize the request document structure.

12 Click the **Response** tab at the top of the dialog box to bring the Response document pane forward. This pane shows a tree view of the Response document, similar to the one shown for the Request document. The same user interface features that applied to the Request document pane apply to the Response document pane.
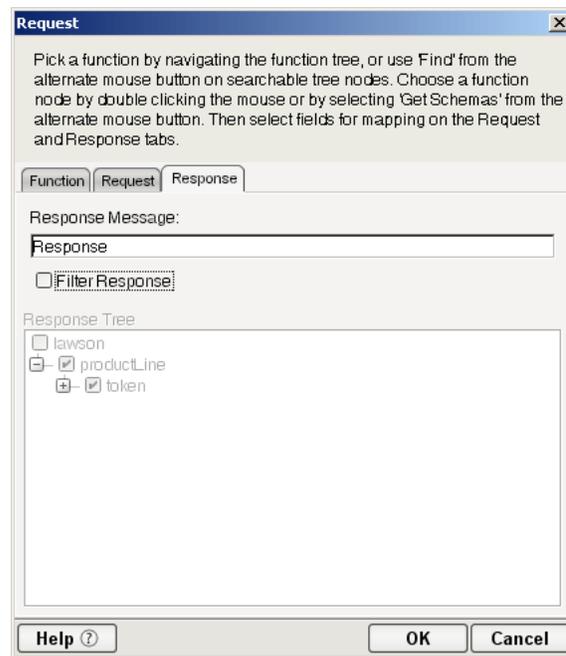
13 Type a name for the response message in the **Response Message** field, or accept the default response message name.

14 Select or deselect **Filter Response**, as desired. Filter Response is selected by default. If deselected, the tree will be disabled and greyed (no longer editable), indicating that a default schema configuration will be used. If selected, you can customize the structure of the request document by selecting the nodes to be included in the response document.

15 Visit all of the nodes in the Response document tree and customize the settings as desired.

16 Select **OK**. The dialog closes and a new action appears in the Action Model of your component. The Native Environment Pane changes to show the Request and Response tabs along with tree views of the request and response documents.

17 To map a node of the Input document to a node in the Request document, drag a node from the Input pane and drop it on a node in the Request tab of the Native Environment pane. This automatically creates a new XML Map Action in the Action Model.

18 To map a node of the Response document to a node in the Output document, drag a node from the Response tab of the Native Environment pane and drop it on a node in the Output pane. This automatically creates a new XML Map Action in the Action Model. You can cut, copy, or delete the actions once they appear in the Action Model pane.

19 Add or remove actions from the Action Model to create the desired business logic.

Returning to Schema-Edit Mode

Once you have created a Request Action in your Action Model (see), you can go back and edit the Request and Response schema trees by double-clicking the Request Action in the Action Model. Double-clicking causes the Request dialog box to be displayed. You can use the Request and Response tabs to navigate the request- and response-document schemas and change your customizations as desired. When you exit from of the dialog box, your changes are reflected in the Native Environment Pane.

NOTE: If you manually edit the Request or Response documents in the Native Environment Pane (see "Manually Editing the Request and Response Documents" on page 32), you will not be able to reopen the Request Action dialog. Avoid manually editing request/response documents. Instead, make modifications through careful use of the XML Map Action or by using Request.createXPath( ) and DOM methods in a Function Action.

Request and Response Documents

After you have created a Request Action (see "Creating Actions in the Component Editor" on page 26), the Native Environment Pane portion of the Integration Manager editor view displays Request and Response documents in tree-view form, as shown in the following illustration.
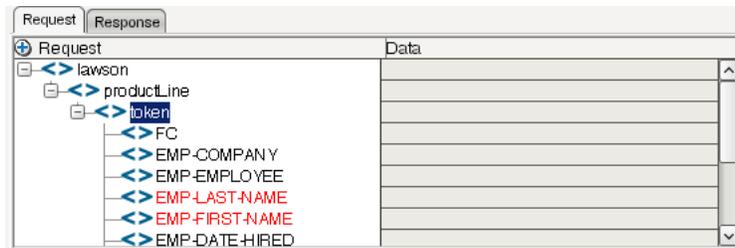
You can drag and drop data from the Input document to the Request document to create XML Map actions. You also can use drag-and-drop mapping to map from Response to Output. In the preceding illustration, some nodes are displayed in red. The red color indicates that data has been mapped to the nodes.

Manually Editing the Request and Response Documents

If you right-click anywhere inside the Request or Response tree views in the Native Environment Pane, a menu is displayed.

The **Edit Document** command in this menu opens the document in a text-editor, as shown in the following illustration.

```
Edit Document                                          X
<?xml version="1.0" encoding="UTF-8"?>
<lawson>
  <productLine>
    <token>
      <FC> </FC>
      <EMP-COMPANY/>
      <EMP-EMPLOYEE/>
      <EMP-LAST-NAME/>
      <EMP-FIRST-NAME/>
      <EMP-DATE-HIRED/>
      <EMP-POSITION/>
      <EMP-SUPERVISOR/>
      <EMP-SALARY-CLASS>H</EMP-SALARY-CLASS>
      <EMP-PAY-FREQUENCY>2</EMP-PAY-FREQUENCY>
      <EMP-PAY-GRADE/>
      <DT0>
        <LINE-FC>D</LINE-FC>
        <VALUE/>
        <HEU-CURRENCY-CODE/>
        <HEU-BASE-AMOUNT/>
      </DT0>
    </token>
  </productLine>
</lawson>

  Help ?                              OK      Cancel
```
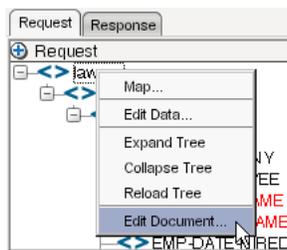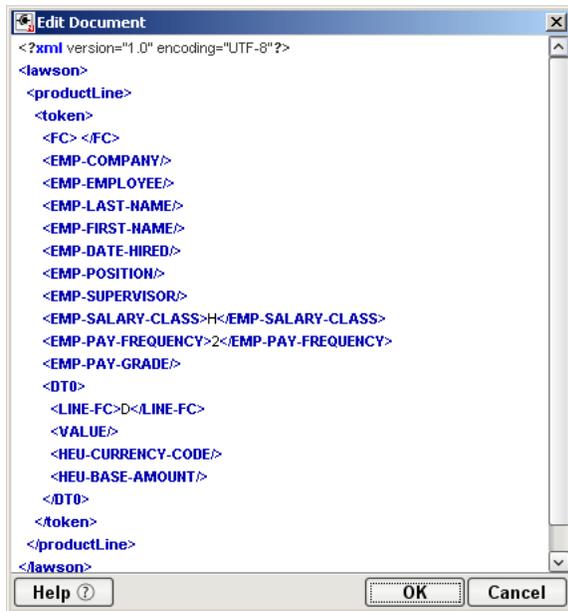
In this window, you can manually edit the Request or Response document.

CAUTION: *If you manually edit the document, it may no longer conform to the original schema. As a result, you will no longer be able to open the document in the Request Action dialog box.*

If you have manually edited a Request or Response document, double-clicking on the corresponding Request Action in the Action Model will cause the following message to be displayed:

```
Message                                                      X

  This Action can no longer be edited because the Request mapping message has been manually modified.

                                                      OK
```

If this message appears, and you want to work with the document using the Request Action dialog box, you must recreate the action in order to see the original Request or Response document. Manual edits will be lost.

To prevent this situation from occurring, *avoid making manual edits to Request or Response documents*. Instead use XML Map Actions and Function actions to modify documents or node contents. Be aware that doing so may violate schema constraints. This will not necessarily cause runtime problems for your component when executing in the Integration Manager Enterprise Server environment, because Integration Manager does not validate the request document against the schema at runtime. However, your Lawson system may validate incoming requests, in which case a manually-edited Request document can cause errors that are difficult to troubleshoot.

"Before Execute" and "After Execute" Actions

Whenever a Lawson Software Request action is created in the Action Model, Integration Manager adds two additional lines to the action list: Before Execute Actions and After Execute Actions. These are header labels (grouping labels) for blocks of actions that occur *before* the request is sent to the Lawson Software system, or *after* a response document has come back, respectively. Typically you will want to map data fields from Input to Request and have those actions be grouped under the "Before Execute Actions" header. Likewise, any map actions or other action logic that you need to perform on the Response document after the Lawson Software system executes your request should be grouped under the "After Execute Actions" heading.
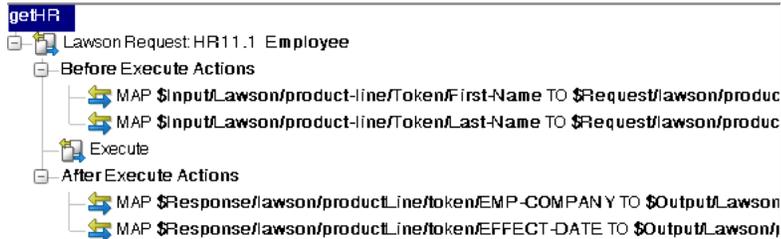
Creating Lawson Services

This section describes how to use the Integration Manager™ Connect for Lawson Software to create a Lawson Service component to connect to Lawson Software and listen for events. As part of the process of creating a Lawson Service, you can select an existing Lawson File connection resource, or you can create a new one. If you create the connection beforehand, it is available for use by any Lawson Services in the current project. *If you have not already created at least one Lawson File connection resource in the current project, you will be prompted to do so when you try to create a Lawson Service.*

NOTE: You can still create a Lawson Service without first creating a connection resource, but you won't be able to use any debug features that depend on a live connection.

➢ **To Create a New Lawson Service:**

1   Select **File** > **New** > **xObject**. The New xObject dialog box is displayed.

   NOTE: Alternatively, under **Service** in the Integration Manager Navigator pane (Explorer view), you can highlight **Lawson Service**, click the right mouse button, then select **New**.
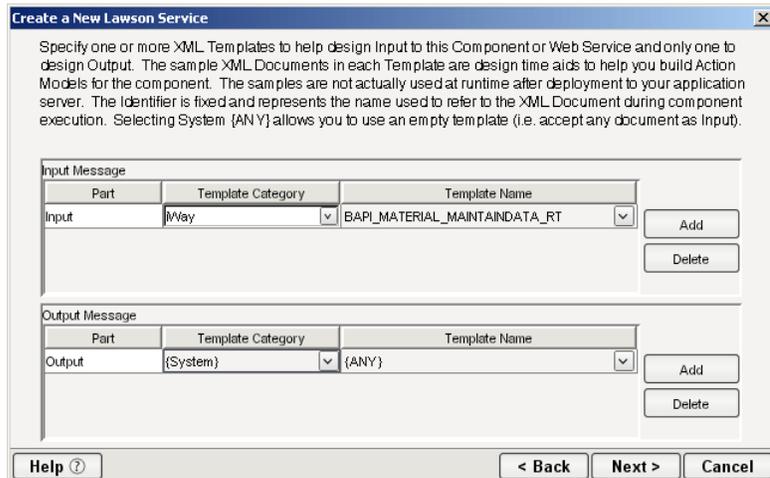
2   Select the **Process/Service** tab.

3   Double-click on "Lawson Service."

   If you have not previously defined a connection resource, you are prompted to do so now (see "Creating a Connection Resource" on page 19).

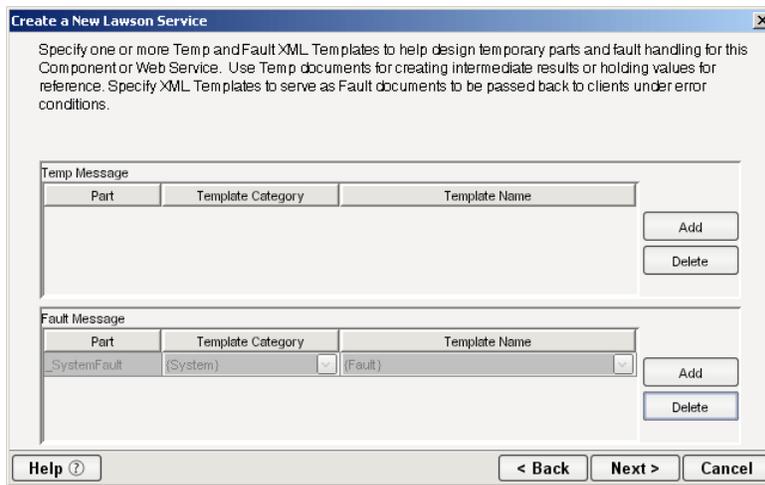   If you have already defined a connection resource, the "Create a New Lawson Service" Wizard is displayed.
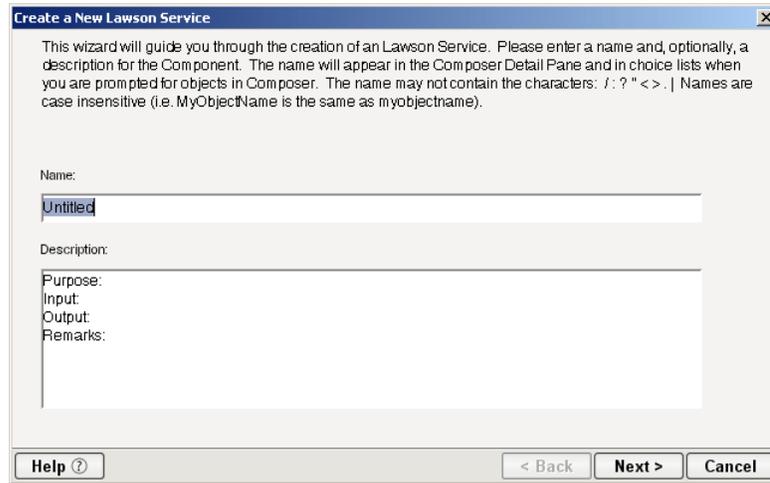
4 Type a **Name** for the new Lawson Service.

5 Optionally, type **Description** text.

6 Select **Next**. The XML Input/Output Message Property Info panel of the New Lawson Service Wizard is displayed.
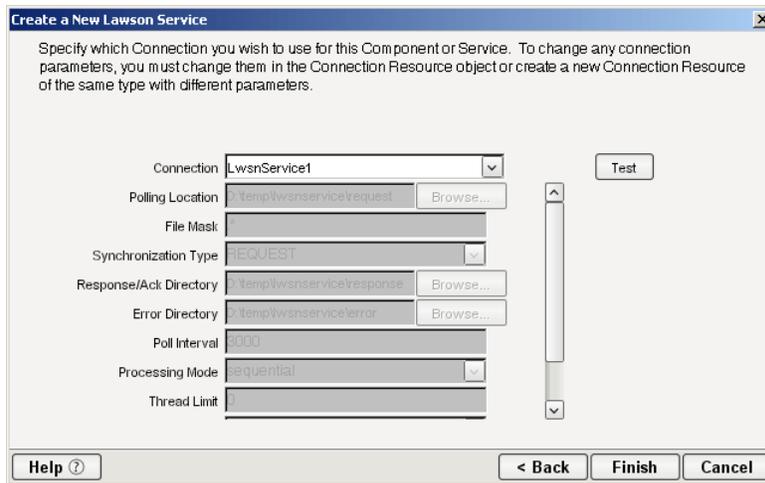

7 Specify the Input and Output templates as follows:

- Type a name for the template under **Part** if you wish the name to appear in the DOM as something other than "Input"

- Select a **Template Category** if it is different than the default category.

- Select a **Template Name** from the list of XML templates in the selected **Template Category**.

- To add additional input XML templates, click **Add**, then choose a **Template Category** and **Template Name** for each.

- To remove an input XML template, select an entry and click **Delete**.

8 Select an XML template for use as an Output DOM using the procedure described in the previous step.

NOTE: You can specify an input or output XML template that contains no structure by selecting {System}{ANY} as the Input or Output template. For more information, see "Creating an Output Document without Using a Template" in the *Novell Integration Manager User's Guide*.

9 Select **Next**. The Temp and Fault XML Template panel is displayed.

10 If desired, specify a template to be used as a scratch pad under the "Temp Message" pane of the dialog box. This can be useful if you need a place to hold values that will be used temporarily during the execution of your component. Select a **Template Category** if it is different than the default category. Then select a **Template Name** from the list of XML templates in the selected Template Category.

11 Under the "Fault Message" pane, select an XML template to be used to pass back to clients when an error condition occurs.

12 To add additional input XML templates, click **Add** and choose a **Template Category** and **Template Name** for each. Repeat as many times as desired. To remove an input XML template, select an entry and click **Delete**.

13 Select **Next**. The Connection Info panel of the Create a New Lawson Service wizard is displayed.

14 Select a Lawson Service File connection from the **Connection** list. The Connection list displays the names of the Lawson connection resources that have been defined in this project.

15 Select **Finish**. The component is created and the Lawson Service Editor is displayed. The features of this window are identical to the features described in "About the Lawson Software Component Editor Window" on page 25. The following sections describe the steps required to create a Lawson Service Action.

Creating Lawson Service Actions

You can create all the normal Integration Manager actions in the Action Model of your Lawson Service (e.g., XML Map, Function, Log, Send Mail). In addition, you can create a Lawson Service Request action.

The Lawson Service Request action is similar to a Switch action (see "The Switch Action" in the in the *Novell Integration Manager User's Guide)*. A Switch action allows program control to branch to a block of actions based on a match between an input value and a Case value. In a Lawson Service Request, Integration Manager receives a DOM, and compares a series of cases against the DOM. If an exact match occurs between the DOM and a case, execution branches to the actions listed underneath the case in the Action model. Cases are tested in the order listed; and once a match is found, execution of the match logic precludes execution of any other logic in the Lawson Service Request.

➢ **To Create a Lawson Service Request Action**

1 In the Action Model, right-click on Lawson Service Request:

```
☐─ LawsonFileService
   ☐─?T→ Lawson Service Request: ServiceRequest
       └─ Default    New Action          ▶
                     Edit Action...
                   ⥮ Disable Action

                     Toggle Breakpoint
                     Clear all Breakpoints

                     Cut
                     Copy
                     Paste
                     Delete...

                     Find...
                     Find Next
                     Replace...
```

2 Select **Edit Action**. The Service Request dialog box is displayed.

3 In the Function tree, click on the plus (+) sign to the left of the product node to expand the tree view for that node.

4 Right-click on the desired function name and select **Add Case,** or double-click on the function name. The Adapter Service Request dialog box is displayed. This dialog box is similar to the Lawson Request dialog box (see "Creating Actions in the Component Editor" on page 26).

5 Select or deselect **Filter Request**, as desired. Filter Request is selected by default. If deselected, the tree will be disabled and greyed (no longer editable), indicating that a default schema configuration will be used. If selected, you can customize the structure of the request document by selecting the nodes to be included in the request document.

6 Expand the nodes of the Service Request Tree by clicking on the plus signs to the left of the nodes.

7 Select the elements that you want to include in the Service Request document by selecting or deselecting the check boxes in the node tree. Some nodes are greyed out (disabled). This is because the document schema determines the nodes that you can edit.

8 If the **Service Response** tab is enabled (this depends on the Synchronization Type setting in the Connection resource), click the **Service Response** tab at the top of the dialog box to bring the Service Response document pane forward. This pane shows a tree view of the Response document, similar to the one shown for the Request document. The same user interface features that applied to the Request document pane apply to the Response document pane.

9 Visit all of the nodes in the Response document tree and customize the settings as desired.

10 Click the **Case Expression** tab at the top of the dialog box to bring the Case Expression pane forward. This pane provides an Expression Builder that you use to build a case expression.
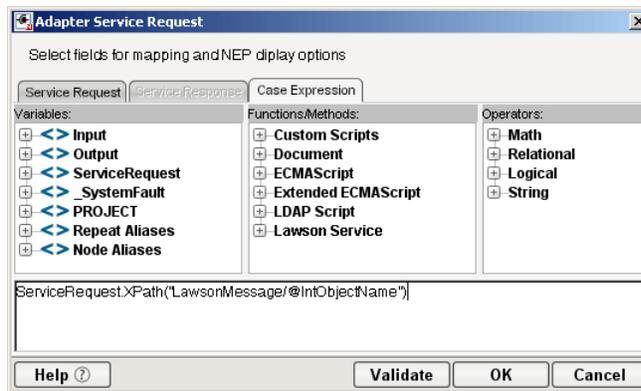
11 Type the static string values or the ECMAScript expressions that will be checked against the input DOM.

12 Select the OK button. The Service Request dialog box is displayed.

13 To add another case, repeat Step 4 through Step 12. The cases are listed from top to bottom in the order in which they are evaluated. Each Case value will be checked in turn, in the order you list them.

- ◆ For optimal performance, list the most likely matches first.

- ◆ You can change the order in which the cases are listed by clicking on the name of a case, then clicking the up or down triangle icons.

- ◆ To edit a case, select the name of the case, then press the **Mapping** button. You can also select the name of the case, then select the Expression Builder button.

14 When you have finished adding cases, select **OK.** The Service Request dialog box closes and a new action appears in the Action Model of your component. The Native Environment Pane changes to show the Request and Response (if the Synchronization Type is set to REQUEST_RESPONSE in the Connection resource) tabs along with tree views of the request and response documents.

15 In the Action Model, add actions to be performed to each case to create the desired business logic.

- ◆ To map a node of the Input document to a node in the Request document, drag a node from the Input pane and drop it on a node in the Request tab of the Native Environment pane. This automatically creates a new XML Map Action in the Action Model.

- ◆ To map a node of the Response document to a node in the Output document, drag a node from the Response tab of the Native Environment pane and drop it on a node in the Output pane. This automatically creates a new XML Map Action in the Action Model. You can cut, copy, or delete the actions once they appear in the Action Model pane.

- ◆ When all of the Actions for a case have been evaluated, Integration Manager sets the Service Response DOM.

- ◆ The final case in the Action model is always labeled Default. This case is generated automatically and cannot be removed. Actions placed under Default are executed if and only if the none of the other cases are matched. While you are not required to place actions under Default, it is good programming practice to have at least some kind of fallback logic for the Default case, even if it's only a Log action or a Raise Error action.

◆ The Request and Response document displays for Service Requests work similarly to the displays for Requests (see).

Managing Deployed Lawson Services

Once a project containing Lawson Services has been deployed (see "Deploying Your Project" in the *Novell Integration Manager User's Guide),* the Listener objects actively listen for messages each time you start your server. To manually start and stop these services you need to use the Lawson Services Console. This browser-based console allows you to see the list of Lawson Services, the status of each service (running or not running), the running tally (Count) of messages received, other administrative information, and a Start/Stop button.

➢ **To display the Lawson Services Console:**

1 Make sure that you have installed the Integration Manager Enterprise Server to your application server (see the *Novell Integration Manager Installation Guide*) and that your application server is running.

2 Start the Integration Manager Enterprise Server (for example, if you are using the JBoss application server, you would use a Web browser to access http://localhost:8080/exteNdComposer). A dialog box for entering the application server administrator ID and password is displayed.

3 Type your administrator ID and password, then select **OK**. The Integration Manager Server Console page is displayed.

4 Scroll down the **About Products** list on the left side of the page until you see the link titled "lwsnservice".
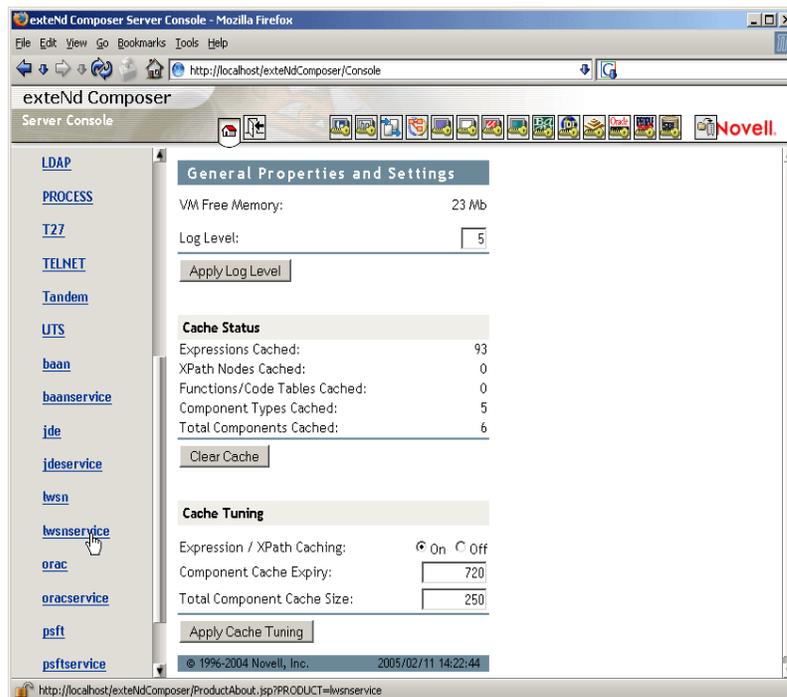
5 Select the "lwsnservice" link. A page is displayed that provides information (e.g., version, license number, copyright) about the Connect. It also displays a Console button.

6 Select the Console button. The Lawson Service console page is displayed. This page lists any Lawson Services that have been deployed.

7 To stop a Lawson Service, select the appropriate **Stop** button (the button will then change to Start).

NOTE: If messages are being handled by a service at the time of the **Stop** command, there may be some delay before the service actually exits. Select the **Refresh** button periodically until the "Running" column of the console says No for the service.

➢ **To undeploy a Lawson Service:**

1 Make sure that you have installed the Integration Manager Enterprise Server to your application server (see the *Novell Integration Manager Installation Guide*) and that your application server is running.

2 Start the Integration Manager Enterprise Server (for example, if you are using the JBoss application server, you would use a Web browser to access http://localhost:8080/exteNdComposer). A dialog box for entering the application server administrator ID and password is displayed.

3 Login as an administrator.

4 Select the **Deployment** icon ( )from the toolbar.

5 Select **Deployed Objects**:

6 Expand the database containing the deployed objects that you want to manage:
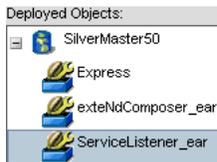
7 Select the object that you want to undeploy.

8 Select the **Undeploy** button.

ECMAScript Extensions

See Appendix A to this guide for a list of ECMAScript extension methods that can be used with the Lawson Connect.

A ECMAScript Methods

The J2EE Connector framework of the Integration Manager Connect for Lawson Software implements a number of ECMAScript extension methods that are available for use through the Expression Builder (which is available in Function Actions and elsewhere). You will find the methods listed in the Functions/Methods pane of the Expression Builder:
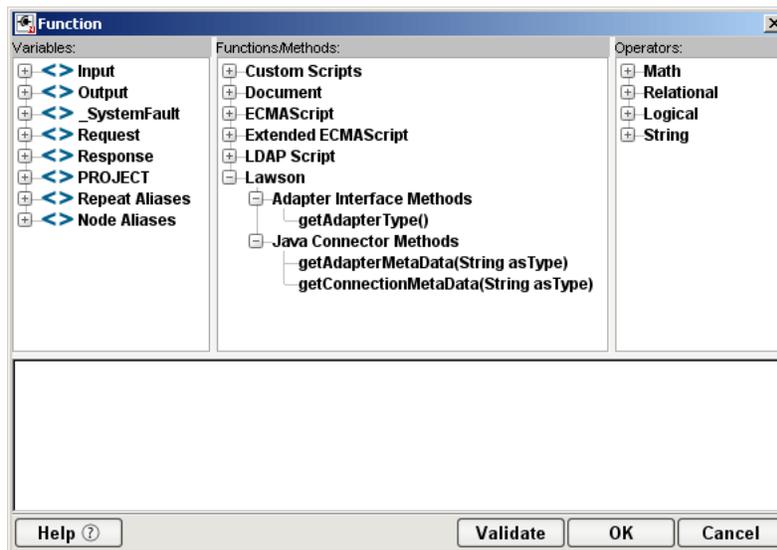
ToolTip help is available for each method.

For more information on working with the Expression Builder, see "Custom Scripting and XPath Logic in Integration Manager" in the *Novell Integration Manager User's Guide*.

Adapter Interface Methods

getAdapterType

This method calls the IGNVAdapterInterface.getAdapterType(), and returns the adapter type name (e.g., "lwsn").

Syntax:
getAdapterType()

Connection Interface Methods

The following CCI API metadata-wrapped methods are published through the Expression Builder.

getAdapterMetaData

This method wraps calls on the javax.resource.cci.ResourceAdapterMetaData class.

Syntax:

getAdapterMetaData(*parameter*)

Parameter	Description
NAME	Returns the name of the adapter (e.g., "iWay JCA adapter").
EXECUTEWITHI	Returns true if the implementation class for the Interaction interface implements public Record execute method; otherwise the method returns false.
EXECUTEWITHIO	Returns true if the implementation class for the Interaction interface implements public boolean execute method; otherwise the method returns false.
VENDOR	Returns the name of the vendor that has provided the resource adapter (e.g., "iWay Software, Inc.").
VERSION	Returns the version of the resource adapter (e.g., "5.5").
INTERACTIONSPECS	Returns the fully-qualified name of the InteractionSpec type supported by this resource adapter (e.g., "com.ibi.afjca.IWAFInteractionSpec").
SPECVERSION	Returns a string representation of the version of the connector architecture specification that is supported by the resource adapter (e.g., "1.0").
LOCALTX	Returns true if the resource adapter implements the LocalTransaction interface and supports local transaction demarcation on the underlying EIS (Enterprise Information System) instance through the LocalTransaction interface.

getConnectionMetaData

This method wraps calls on the javax.resource.cci.ConnectionMetaData class.

Syntax:

getConnectionMetaData(*parameter*)

Parameter	Description
EISPRODUCTNAME	Returns the product name of the underlying EIS instance (e.g., "Lawson").
EISPRODUCTVERSION	Returns product version of the underlying EIS instance (e.g., "Version: 7.3.3.ESP9, Version: 8.0.2.ESP6 (8.0.2.485)").
USER	Returns the user name for an active connection as known to the underlying EIS instance.

Additional Methods

The following ECMA-wrapped methods are not displayed by the Expression Builder user interface, but you can use them in the Expression Builder by typing them directly into expressions.

getWarnings()

Returns line separated list of javax.resource.cci.ResourceWarning for the javax.resource.cci.Interaction.

clearWarnings()

Calls javax.resource.cci.Interaction.clearWarnings().

getLastError()

Returns last error, such as a javax.resource.ResourceException.

Index

A

actions
    After Execute 34
    Before Execute 34
    creating 26
    Lawson Service Request, creating 37
    Request 26
    XML Map 31, 32, 38
Adapter Service Request 37
Add Case 37
administrative console, Integration Manager 16
After Execute actions 34
AGS 12
application servers
    supported 11
    version, supported 9

B

BEA WebLogic 11
Before Execute actions 34

C

Case Expression 37
CCI 13
CGI Root 21
clearWarnings 43
Common Client Interface 13
component, Lawson Software
    creating 23
    Editor window 25
connection parameter
    CGI Root 21
    Connection Type 21, 22
    constant 19
    Error Directory 23
    expression-driven 19
    File Mask 22
    Host or IP Address 21
    HTTPS 21
    invalid 27
    Lawson Port 21
    Maximum Pool Size 21, 23
    Metadata Connection 23
    Poll Interval 23
    Polling Location 22

Processing Mode 23
    Response/Ack Directory 23
    switching from constant to expression-driven 20
    Synchronization Type 23
    Thread Limit 23
    User ID 21
connection resource
    File service 22
    Lawson connection 20
    types of 19
Connection Type 21, 22
Connects
    description 11
    Lawson, about 12

D

Default case 38
Deployed Objects 40
DOM, output, selecting XML template for 24, 35

E

ECMAScript
    Integration Manager, technology, enabling 11
    methods 40, 41
Edit Document 32
Edit License 16
EIS 13
EISPRODUCTNAME 42
EISPRODUCTVERSION 42
Error Directory 23
EXECUTEWITHI 42
EXECUTEWITHIO 42
expression-driven 20

F

Fault Message 25, 36
File Mask 22
Filter Request 37
Filter Response 31

G

Get Schemas 28

ToolTips 28, 29

Lawson Software 9

U

Undeploy 40
User ID 21

V

variables, project 20
versions, supported
    application servers 9
    Integration Manager 9

X

XML Map Action 32, 33
XML Map Component Editor 11, 25
XML templates 19, 23
xObject
    description 19
    New 20, 22, 23, 34
XPath 11
XSLT 11