VerticalNet Solutions
A Division of VerticalNet

VerticalNet OSM Platform™

OSM Marketplace Implementation Guide

Preview Release

Software Version number

0.10

Publication date

December 31, 2000

Credits

Writer: Adam Cheyer
Design: Avi Metcalfe

Contents

About This Manual

This chapter introduces the features and organization of ***OSM Marketplace Implementation Guide***.

This document includes instructions for all the steps required to build a new market:

1. Define the screen flows and presentation layer for a marketplace.
2. Choose the types of services that will be offered in the marketplace, and configure the trading vocabularies and APIs that define them.
3. Compose services into process flows that elaborate the business logic required to run the marketplace.
4. Utilize the Service Selection wizard to specify requirements that will be incorporated by the Service Coordination Advisor component as it helps choose optimal business partners from the Service Registry.

Audience

The ***OSM Marketplace Implementation Guide*** is intended for software engineers or a professional service team working for a marketplace operator desiring to build and deploy an OSM-enabled public or private marketplace.

Conventions

The following typographic conventions are used in this manual to convey special types of information.

Code

Code fragments and code listings are shown as follows:

```
code {
  code appears in the courier monospaced
font
     }
```

Command Lines

Information that a user must type on a command line is represented as follows:

```
javac -switch
```

Tips

Important information about a specific topic is presented as a "tip":

Be sure that the java files are in the same directory as the include files!

Related Documents

The following related documents are included in the Open Services Marketplace (OSM) documentation set.

OSM Platform Overview

Gives a basic description of the Open Services Marketplace platform. It includes an overview of the approach, benefits, tools, and applications of the OSM for public and private marketplaces.

OSM Marketplace Implementation Guide (this document)

Using this guide, a marketplace operator can describe the business processes, trading vocabularies, and rules for creating an adaptable private or public marketplace in a specific domain. Operation instructions are given for OSM's business process management tools, the E-Service Selection Wizard, the OSM B2B Service Ontology, and the OSM Service Coordination components.

Service Publisher User's Guide

Third party service providers employ Service Publisher to register descriptions of their services in OSM-enabled marketplaces.

Programmer's Reference Manual

Describes low-level programmer's interfaces for the key functional components of the OSM platform.

Ontology Builder User Guide

Describes how to use the Ontology Builder to develop and share ontologies.

Marketplace Administration Guide

Describes administration, logging and auditing tools for an OSM marketplace.

OSM Installation Guide

Describes how to install the OSM platform and tools.

OSM Platform Release Notes

Contains the latest information that is not included in any of the other documents in the documentation suite.

Chapter 1: OSM Platform Architectural Overview

VerticalNet's Open Services Marketplace (OSM) Platform consists of a broad set of server-class software products used for building public or private marketplaces leveraging a just-in-time integration approach. The OSM Platform enables the marketplace builder to:

- Create and maintain a flexible electronic marketplace with easy customization of presentation and underlying functionality.
- Design the business process flows for the marketplace application using a graphical process editor.
- Integrate the marketplace both with internal systems and with external third party services.
- Define the business rules for the system that govern how service providers are selected (at runtime) to best fulfill the needs of the marketplace.

A Layered Architecture

The OSM Platform comprises a suite of server-level components arranged in a 3-tier hierarchy to deliver electronic marketplace functionality. Each layer is implemented using well-defined interfaces that promote flexible integration through clean layers of program abstraction.

Presentation Layer

Through the Presentation Layer, a marketplace implementor can define the look and feel and the screen sequencing of the user interface, with only a loose coupling to the business logic of the underlying marketplace. Although marketplace developers are free to incorporate any commerce site presentation strategy they choose (e.g., XSL over XML, Microsoft .ASP), we believe VerticalNet's C2/SF Application Server offers numerous advantages as the front end to an OSM-enabled marketplace. The C2/SF Application Server is an open framework for integrating modular service components using an industry-standard JSP/J2EE approach.

Business Logic Layer

The Business Logic Layer provides multiple techniques for an implementor to implement the functionality behind the graphical user interface of the marketplace. User events can be tied to logic specified as Java code, or, to promote enhanced

flexibility, can be linked to business processes stored in a business process library. Instead of a one-size-fits-all approach, multiple processes can be rapidly created and customized for particular situations and user segments.

Although the OSM Platform supports just about any business process engine or process editor tool, out-of-the-box support is provided for Microsoft's BizTalk Orchestration process tool, and soon, for HP's Business Process Management system.

Services Layer

The Services Layer enables business process logic to access and integrate with component services defined by the marketplace or by external business partners. As part of the Services Layer, the OSM Platform provides a number of software tools and servers to manage the just-in-time integration and coordination of web-based services.

- VerticalNet Ontology Builder (aids the market builder in constructing ontologies through which services and processes are structured)
- OSM Service Registry (an ontology-based, UDDI-compliant directory of web-based services)
- OSM Service Coordination Advisor (an extensible rule-based expert system that helps manage optimal business partner selection)
- OSM Service Execution Engine (a high-performance server that manages reliable, scalable communications with remote business services)

In the following chapters, we will look at how to define each of these layers in order to construct an application based on the OSM Platform.

Chapter 2: Presentation Layer using C2/SF

This chapter describes how to use VerticalNet's Cooperative Commerce Service Framework (C2/SF) and the C2/SF Component Builder tool to create a front end to your OSM-enabled marketplace. Although any presentation layer tools or techniques can be used in conjunction with the core "logic" layer of the OSM Platform, C2/SF is the recommended approach because of its ease-of-use, scalability, and because it comes with VerticalNet technical support. For additional detail about using C2/SF, please consult the *Cooperative Commerce Service Framework User's Guide* supplied as part of the EMarketplace Suite 3.0.5 product.

Cooperative Commerce Service Framework

Cooperative Commerce Service Framework (C2/SF) is an electronic marketplace specific framework that leverages standard middleware technology to provide an integration environment and presentation layer for high-level business applications.

Features and Benefits

The C2/SF framework provides the following benefits as part of the OSM platform:

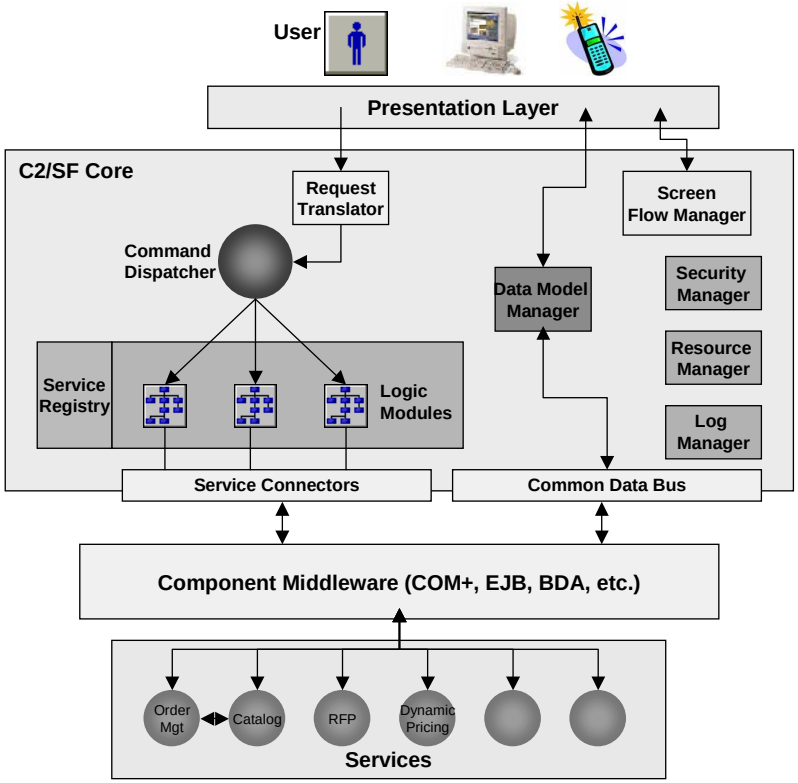
- A uniform integration & deployment standard for aggressive product roll-out
- A proven methodology for quick system integration and application prototyping
- Facilitates internationalization and localization
- Rapid integration of a diversity of service components and business processes:
  - Provides separation and coordination between presentation and business layers via MVC
  - Provides service configuration and management
  - Integrated security management

C2/SF Core Components

The core module of C2/SF comprises several components:

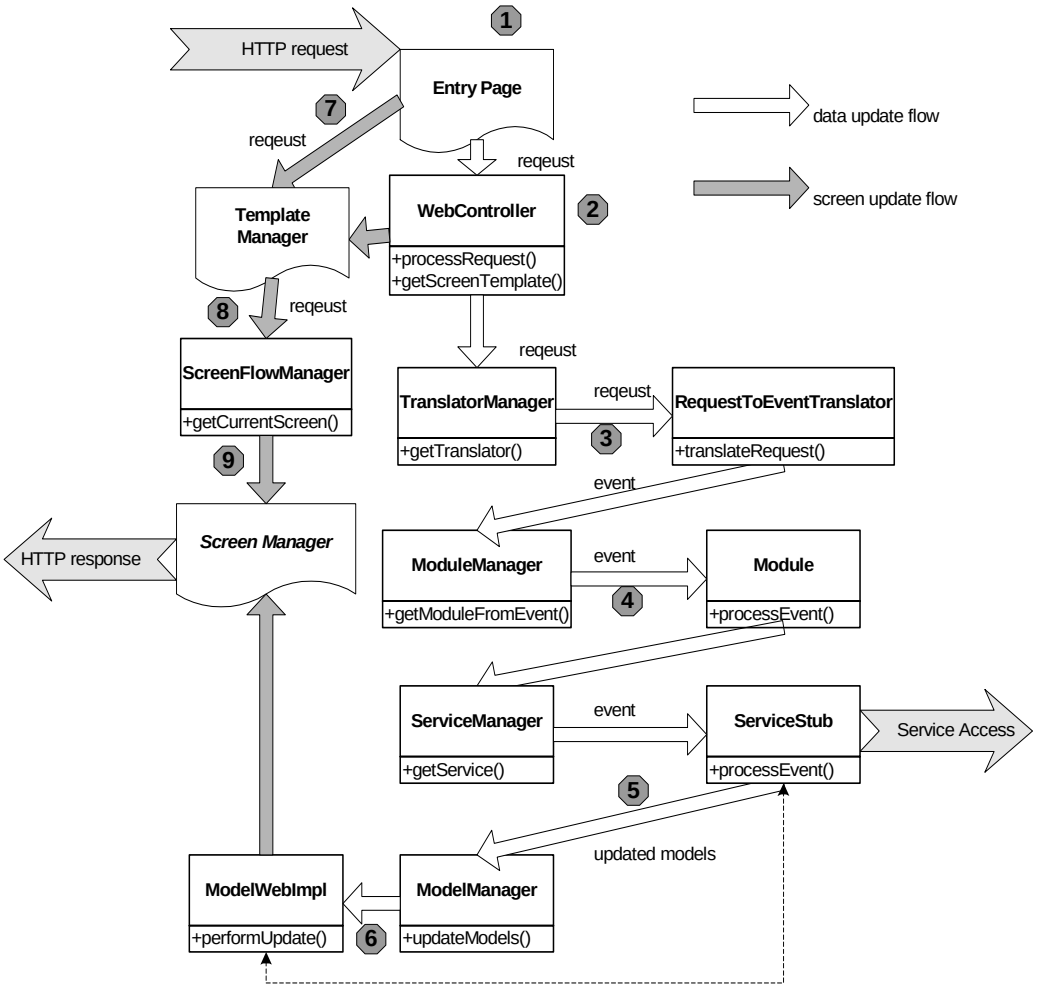
- Entry Page
- Screen Flow Manager
- Template / Screen Manager
- Web Controller
- Translator Manager
- Model Manager / Databus Manager
- Module Manager
- Service Manager
- Resource Manager
- Security Manager
- Log Manager

The following figure shows a high-level view of the C2/SF system architecture.



eMarketplace C2/SF Core System Architecture

The next figure shows the dataflow relationships of the C2/SF core components.



eMarketplace C2/SF Core Components Dataflow

The following sections describe each of the C2/SF core components in more detail. Refer to the previous figures as necessary.

Entry Page

The Entry Page component handles HTTP requests entering the system. It implements the following features:

- Entry point for the entire marketplace
- Asks the Web Controller to process the request
- Receives a screen template based on the request from the Web Property Manager (via the Web Controller)
- Redirects the request to the screen template to generate an HTTP response based on the request and changed data.
- Handles system-level exceptions on the page

Template Manager

The Template Manager attaches the appropriate screen template to a request and passes the request along to the Screen Flow Manager. The Template manager implements the following features:

Screen Flow Manager

The Screen Flow Manager component receives packaged requests from the Template Manager and passes a sequence of screens to be displayed to the Screen Manager component. There are two types of Screen Flow Manager instances:

- Default Screen Manager. Reads the screen flow information from a web property file
- Customized Screen Flow Manager. Implements a state-machine of the screen flow for complex situations which cannot be handled by the default Screen Flow Manager.

Web Controller

The Web Controller component handles requests in parallel to the screen control logic described above. The Web Controller forward requests to the internal C2/SF core logic so that any required service that is a part of the request can be invoked. The Web Controller component implements the following methods:

- Instantiates a RequestToEventTranslator to translate the request into an EMarketEvent object.
- Sends the event to a corresponding Module for processing
- Tells the Module Manager to notify all interested model web implementations based on the information returned by the Module about which models were changed during event processing

Translator Manager

The Translator Manager component receives requests from the Web Controller. Its responsibility is to instantiate a RequestToEventTranslator object per request to be processed. The Translator Manager implements the following features:

- Given a request (with URL), the Translator Manager returns an object that conforms to the `IRequestToEventTranslator` API

- The Translator Manager ensures that the framework supports multiple types of clients

Integration Module Manager

The Integration Module Manager receives the events generated from requests by `Request-ToEventTranslator` objects. The Integration Module Manager in turn instantiates a Module object for each event. Each Module object is responsible for handling the preparation of an event for the Service Manager (described below). The Integration Module Manager implements the following features:

- Given an event, it returns an Integration Module object that conforms to the `IModule` API
- The Module Manager does a look-up in a configuration file to determine which module(s) can process an incoming event.
- Event processing can be both synchronous and asynchronous

Service Manager

The Service Manager receives processed events from the individual Integration Module objects that were created to process them. The Service Manager maps the event types to service connectors that can process them. It then instantiates Service Stub objects that bridge out of the C2/SF system environment to access external services. The Service Stub objects are non-opaque and can receive return information at the conclusion of the external processing of the events.

The Service Manger implements the following features:

- Given an event, the Service Manager can return a service connector that conforms to `IService` API.
- Supports both remote service and local service.
- Handles service connection granularities.
- Handles connection exceptions.
- Can be extended for integration of intelligent service discovery software (such as mSpace).

Data Model Manager

The Data Model Manager component handles updated data models that are generated by the activity of the Service Stub objects described above. The Data Model Manager notifies all relevant model web implementations that their underlying model has been changed, and that they need to respond accordingly (dynamic screen updates, etc.).

The Data Model Manager implements the following features:

- Creates and manages a collection of personalized data model implementations that conform to `IModel` API
- Can notify all relevant model web implementations that need to be updated (due to their underlying models having changed)
- Data models are sharable through the common data bus

- Security control can be maintained at the bus level

Security Manager

The Security Manager is a low-level service that is available across the C2/SF core implementation. It provides the following features:

- Security is defined declaratively, and conforms to the J2EE 1.2 (and J2SE 1.2) security models
- Allows both anonymous (guest) user and registered users
- When a guarded resource (a web page, an EJB connection, a DB connect, etc.) is being accessed without proper credential, a login page is automatically displayed
- Unfortunately, different vendors may build their security mechanism differently
- A security adapter is used to encapsulate the differences in implementation

Resource Manager

The Resource Manager is a low-level service that is available across the C2/SF core implementation. It provides the following features:

- Internationalization. Since the web-tier is mainly built using unicode-compliant technology, internationalization (via the use of unicode standard) can be achieved relatively easily. New modules should be added with internationalization in mind (i.e. try to be locale independent as much as possible).
- ResourceBundle can be used to facilitate localization
- The framework provides uniform handling of resource bundles
- Other services such as translation, locale management, etc. can be introduced if needed

Log Manager

The Log Manager is a low-level service that is available across the C2/SF core implementation. It provides the following features:

- A standard API to which new components can direct log information.

C2/SF Component Standards

One of the premier benefits of using the C2/SF framework is its component API standards. Once the C2/SF framework has been integrated in a development effort, all programmers can interact with each other's and the system's native components using a single standard, dramatically reducing the learning curve for new project personnel, and unburdening the project leadership from having to invent a standard.

C2/SF implements a standardized component API in the following areas:

- A set of property files for defining resources, configurations, screen flows

- A set of standard API for new modules and services that need to be integrated into the framework
- A core implementation that ensures that components implemented conforming to these APIs will function properly
- A set of service components that conform to C2/SF API

C2/SF Property Files

The standardized use of property files allows market builders and system administrators flexibility in the implementation of a marketplace. C2/SF implements the following system property files:

- Web Property File
  - Managed by Web Property Manager.
  - Screens, templates, URLs, screen flow information, request to event translation information
- Module Property File
  - Managed by Module Property Manager.
  - Models, modules, event-to-module mapping information
- Service Property File
  - Managed by Service Property Manager.
  - Services, connection, granularity information

C2/SF Component Builder

C2/SF Component Builder tool is a wizard that allows you to quickly set up a project to build a new service for integration within the C2/SF framework.

Note: *C2/SF Component Builder requires Java 1.3 to be installed on the developer's machine. Refer to* VerticalNet eMarketplace Suite Installation Guide *for instructions on specifying the pathnames required to interface to the Java 1.3 runtime environment.*

C2/SF Component Builder builds and maintains a project source tree, and saves the state of the project in a project file. It allows you to set properties that control which services, screens, and modules the C2/SF deployment will access. Using C2/SF Component Builder you can maintain multiple service component projects at the same time with a minimum of project file tracking overhead.

The finished output of a C2/SF Component Builder session is a shell source tree that contains all the java source files that define the new service executable and its required interfaces. There are options for setting the *scope* of a service component to "application" (one service for all users) or "session" (new service for each user), as well as the *type* of a service component to be "Generic" (service stub handles all processing) or "EJB Delegated" (uses EJB interfaces).

After building a service component project with C2/SF Component builder, all that remains to be done is to code the specific functionality of the

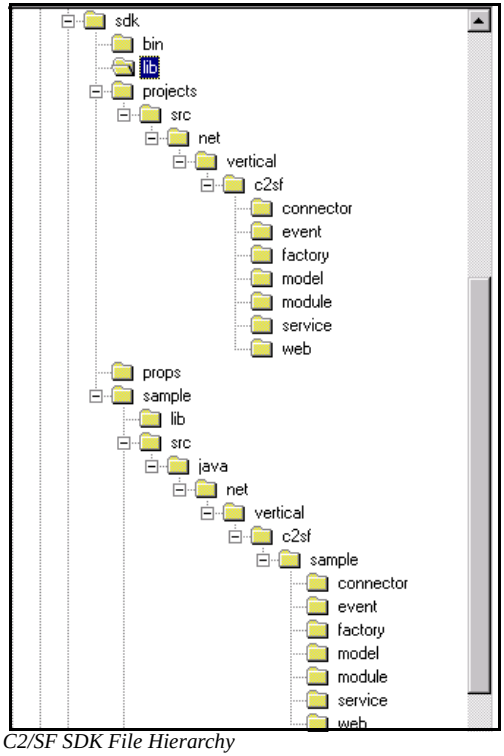
service in the shell source. The generated source is clearly marked with comments that flag where the internal functionality needs to be coded.

C2/SF Component Builder SDK Organization

Refer to *VerticalNet eMarketplace Suite Installation Guide* for instructions on how to specify the location for the installation of the C2/SF Component Builder SDK hierarchy. This discussion will pick up all file paths beginning at the SDK root directory.

Application File Organization

The C2/SF Component Builder application is installed in a directory hierarchy as shown in the following figure:



C2/SF SDK File Hierarchy

- **bin** contains the go.bat MS-DOS batch file which launches the application
- **lib** contains the application Java executables in JAR files
- **projects** is the recommended default location for your project files

- **sample** contains an already-build service component source shell and project file
- **templates** contains the Java and XML source templates that the tool uses to build a service component shell

Project File Organization

The **projects** directory contains a source tree that is organized in a Java package hierarchy that a compiler would expect to see when integrating a new component into the C2/SF environment. The service component sources are divided into a set of directories that define the various interface modules and the executable module.

- **connector** contains the generated source file that will have the most developer authored logic at the completion of the implementation
- **event** contains the generated source files that define the event objects for the new service
- **factory** contains the generated source code that defines a factory subclass to be used to generate instances of your new service in the C2/SF server environment. The factory object also generates the (Web) RequestToEvent-Translator which translates an incoming request from the Web to an event object. See Chapter 1 for more details.
- **model** contains the generated source code that defines the web-tier representation of your new service component model
- **module** contains the generated source code that defines the interface of your new service component to the Integration Module Manager
- **service** contains the source files that define the EJB interfaces if your service is of type “EJB Delegated”.
- **web** contains the generated source code that defines a class for the RequestToEventTrans-lator objects that will be instantiated by the Translator Manager during processing of requests to your new service component

Starting up C2/SF Component Builder

To launch the C2/SF Component Builder tool, open an MS-DOS command window. Change directories to **...\sdk\bin** and type:

```
go
```

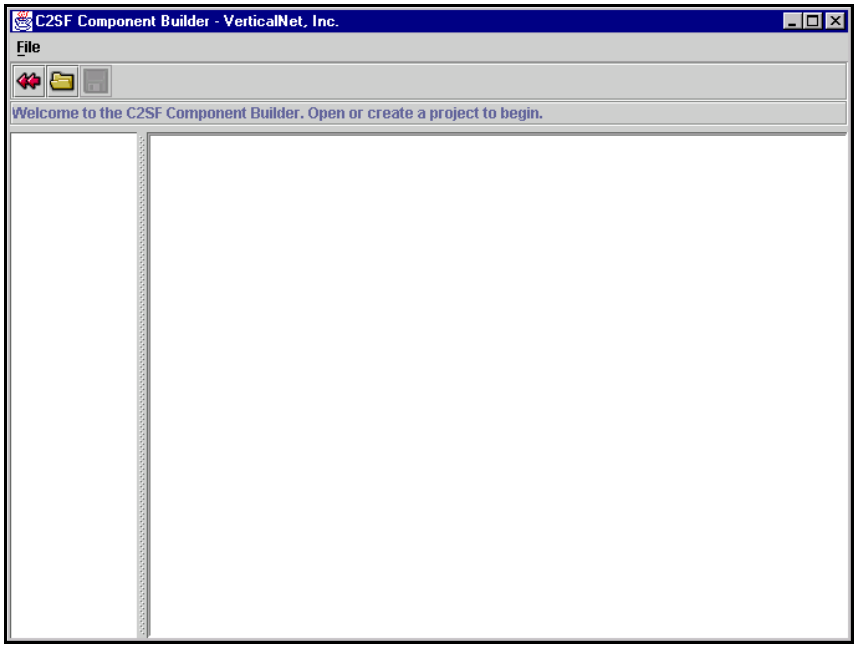
to launch the tool. The Java 1.3 JVM is invoked and the tool is launched.

Example Service Component Project

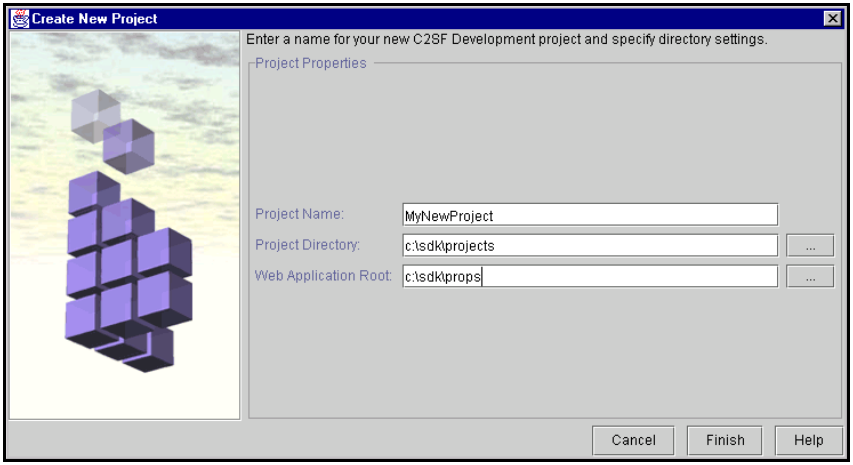
The following sections guide you through the process of using C2/SF Component Builder to create a service component project. The example used is “MyCatalogPing”, a simple service object that, when accessed from an HTML page, pings an externally connected catalog server to test for connectivity and activity.

Starting a Project

When you first launch the C2/SF Component Builder tool, the C2/SF Component Builder main window is displayed as shown in the following figure.



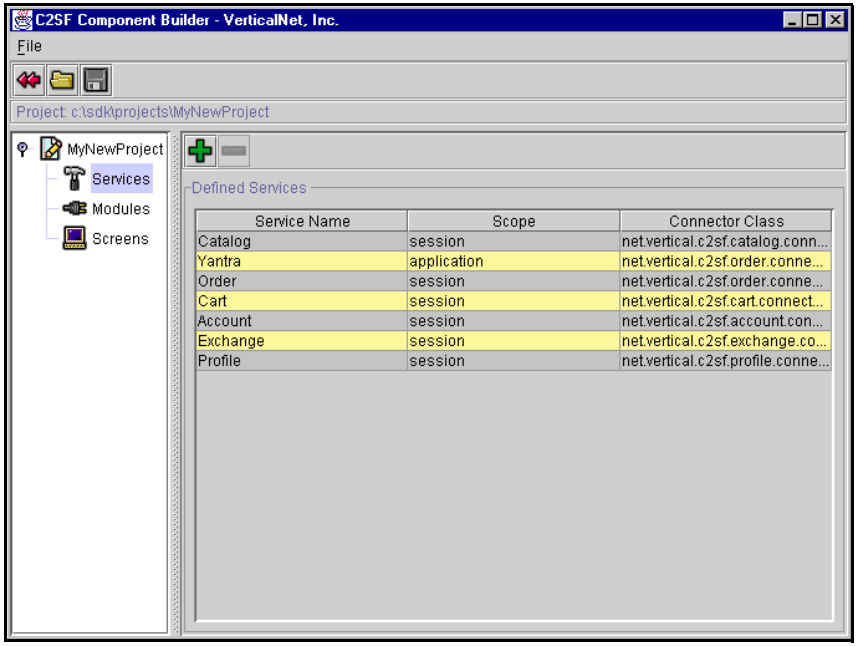
To start a new service component project select File>New Project. The Create New Project panel is displayed as shown in the following figure:



Enter the following data in the fields:

- **Project Name** takes the name of your new project. when you save the project state, it will be stored in a project file with this name.
- **Project Directory** takes the path to a directory under which C2/SF will build a hierarchy of project folders to contain the generated source as described above. It is recommended that use use the default projects directory in the C2/SF SDK hierarchy.
- **C2SF Properties Files** takes the path to a directory that contains the definitions of the service components C2/SF server environment requirements. These definition files are specific to the C2/SF environment and are installed with C2/SF. It is recommended that you use the default properties directory prop supplied with the SDK installation.

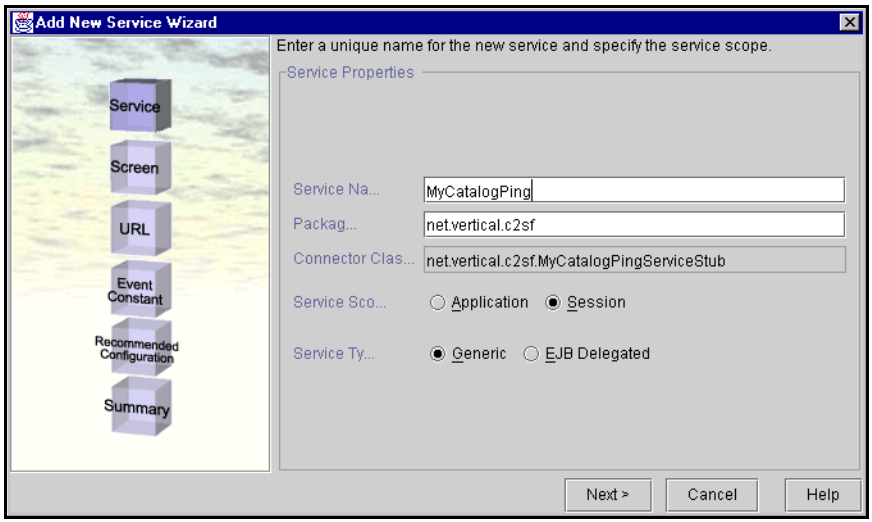
When you have completed specifying these general characteristics for your project click on the Finish button. The Create New Project panel is dismissed, and you can now see the entry for your new project in information line of the C2/SF Component Builder main window as shown in following figure.



Now you are ready to start defining the various interface characteristics of your new service component.

Defining the Service Component

To launch the New Service Wizard, click on the green plus icon "+" in the C2/SF Component Builder main window. The first panel of the New Service Wizard is displayed as shown in the following figure.



Enter appropriate data in the following fields:

- **Service Name** takes the name of your new service component. This string will be used as a preface in many objects throughout the generated source (see the Connector Class field in the figure above), so choose it with care.
- **Package Name** takes the path fragment that will be used to organize the generated source tree on a "base" Java package
- **Connector Class** is built automatically as you enter the Service Name. It shows you the Java package structure that will be created when the source files are generated.

Select the appropriate setting for your service component's scope and type.

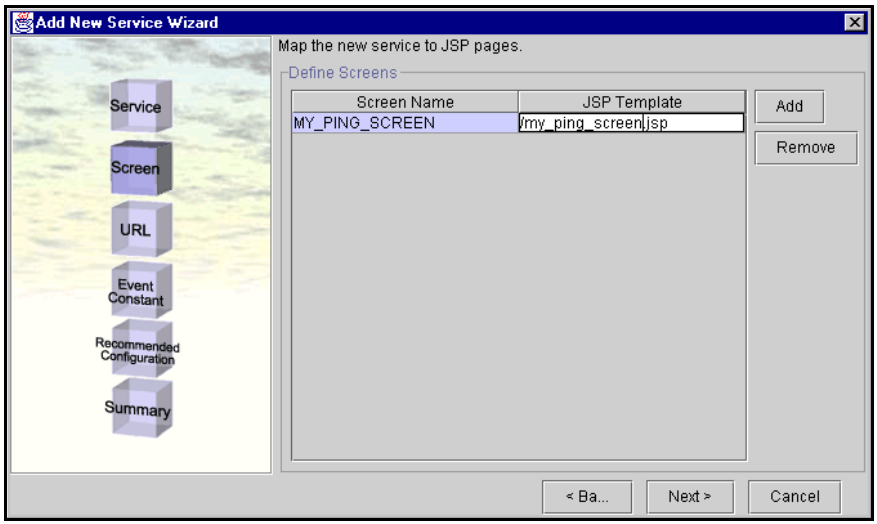
- **Service Scope** can be Application or Session. This defines the transaction scope of the service component.

- **Service Type** can be Generic or EJB Delegated. In a generic service component, the service stub handles all processing either internally or through communication with an external process or server. In an EJB Delegated service component, a service will delegate the processing of a request to an Enterprise Java Bean (EJB). If the EJB Delegated option is checked, EJB interface source files (Home, Remote, Implementation) will be generated for the new service component.

When you finished defining the service click on the Next button.

Defining Service Component Screens

The next panel of the New Service Wizard is used for defining HTML screens that can be used to display results produced by the service component.



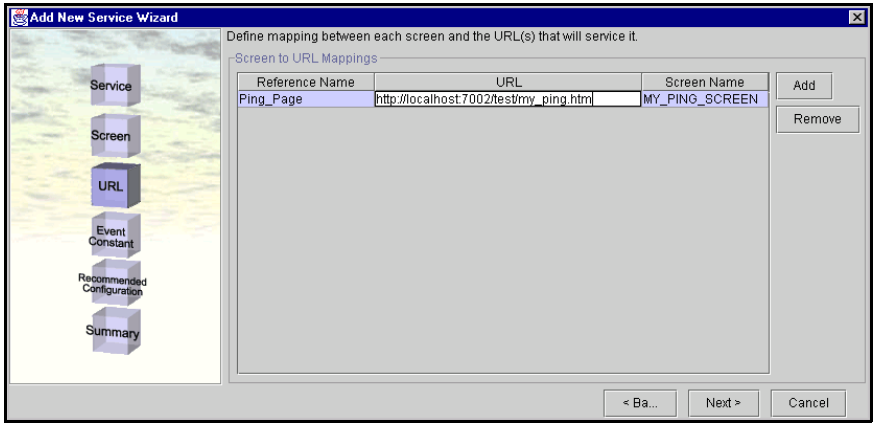
Each screen is uniquely identified by a developer-assigned name. Because C2/SF is a Java 2 Enterprise Edition (J2EE) based framework, the rendering of each screen is actually performed within a developer-implemented Java Server Page (JSP).

In some cases, a service may not have any screens associated with it. For example, a service that is only accessed by other services, and never directly by the user through a web browser, would not require any screens to be defined for it.

When you have completed the process of defining screens for your service object, click on the Next button.

URL to Screen Mapping

The next panel of the New Service Wizard is used for mapping URLs to the screens defined for the service component.



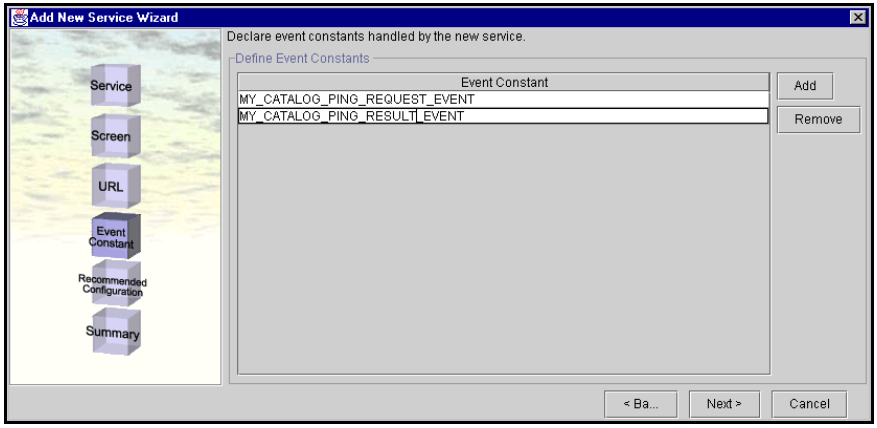
A screen defined in the C2/SF system can be referenced by more than one URL. In the typical scenario a screen would be accessed via exactly one URL.

However, in some cases, a single screen may handle the rendering for a sequence of web pages. For example, if the user were required to go through a multi-step configuration wizard for creating a new account, each web page in the series might actually be rendered by the same underlying screen (i.e. one JSP page).

After you have completed the specification of URLs, click on the Next button.

Event Constant Definition

The next panel of the New Service Wizard is used for defining the events that will be processed by your service component.



The C2/SF system invokes the methods of a service component by delivering events to it. Service components generally do not perform only one task, but rather are implemented to handle several different "types" of requests.

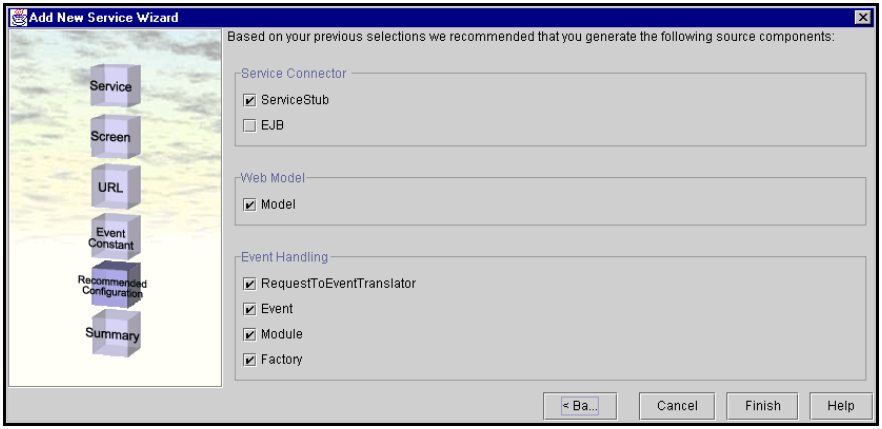
A service component can quickly and easily identify what needs to be done by asking the event delivered to it for its type. When you specify an individual event type in this Wizard panel, a section of generic event-handling code will be inserted and flagged with comments in the generated source files.

When you have finished defining events, click on the Next button.

Generating the Source Code

The next panel of the New Service Wizard is used for defining the source modules that will be generated.

The C2/SF Component Builder will display a list of recommended source modules to be generated, based on the selections and specifications you have made in the previous New Service Wizard panels.

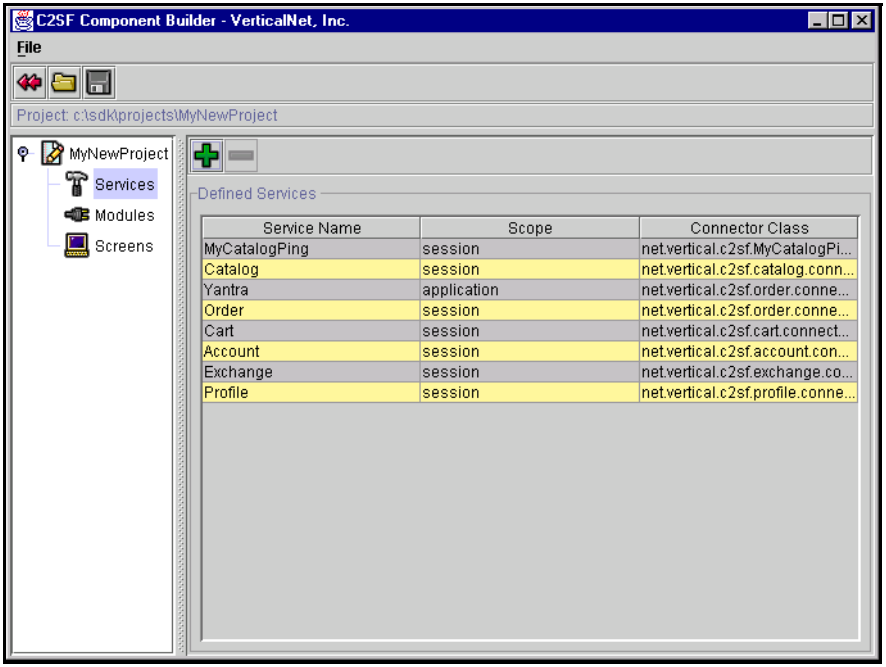


In general, only advanced users should alter the default recommendation. Deselecting recommended options will result in required source files not being generated, and this can prevent the new service component from compiling.

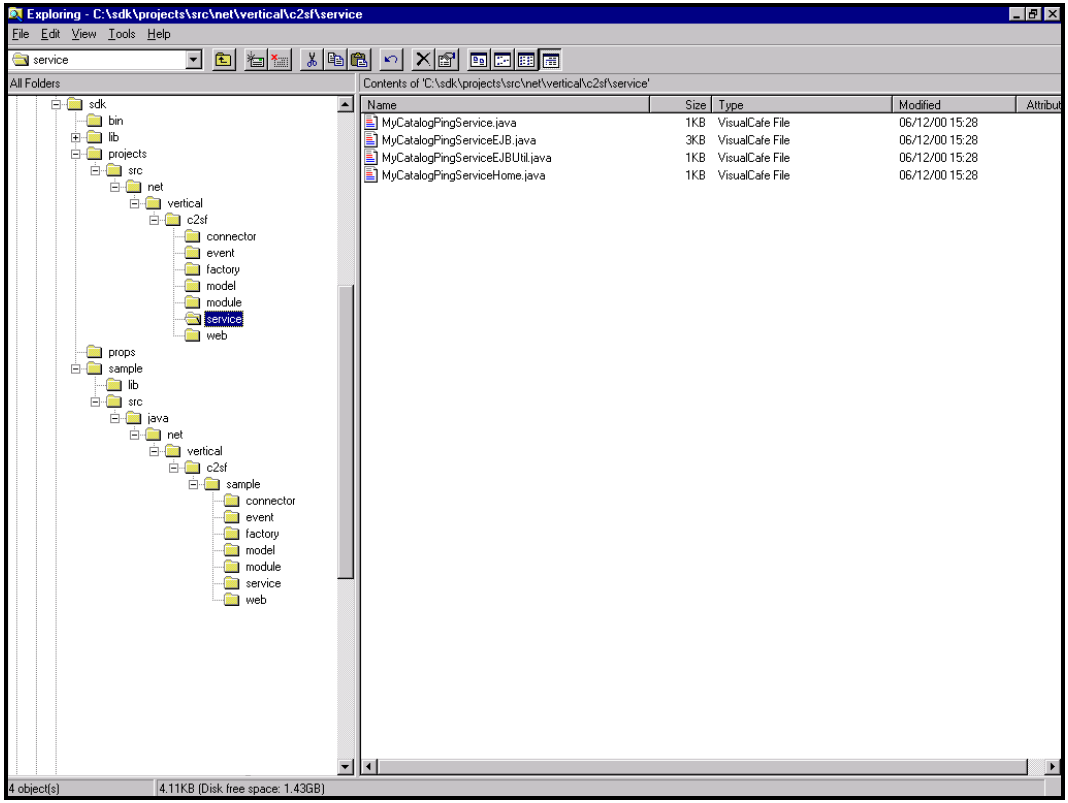
When you have finished specifying the source modules to be generated, click on the Finish button. C2/SF Component Builder tool now generates the source files according to the specifications that you have entered. When the tool has completed the New Service Wizard display is dismissed and you are returned to the C2/SF Component Builder main window.

Verifying the Generated Source Code

You can verify that the new service component has been registered in the C2/SF Component Builder tool by checking for its entry in the Defined Services list as shown in the following figure.

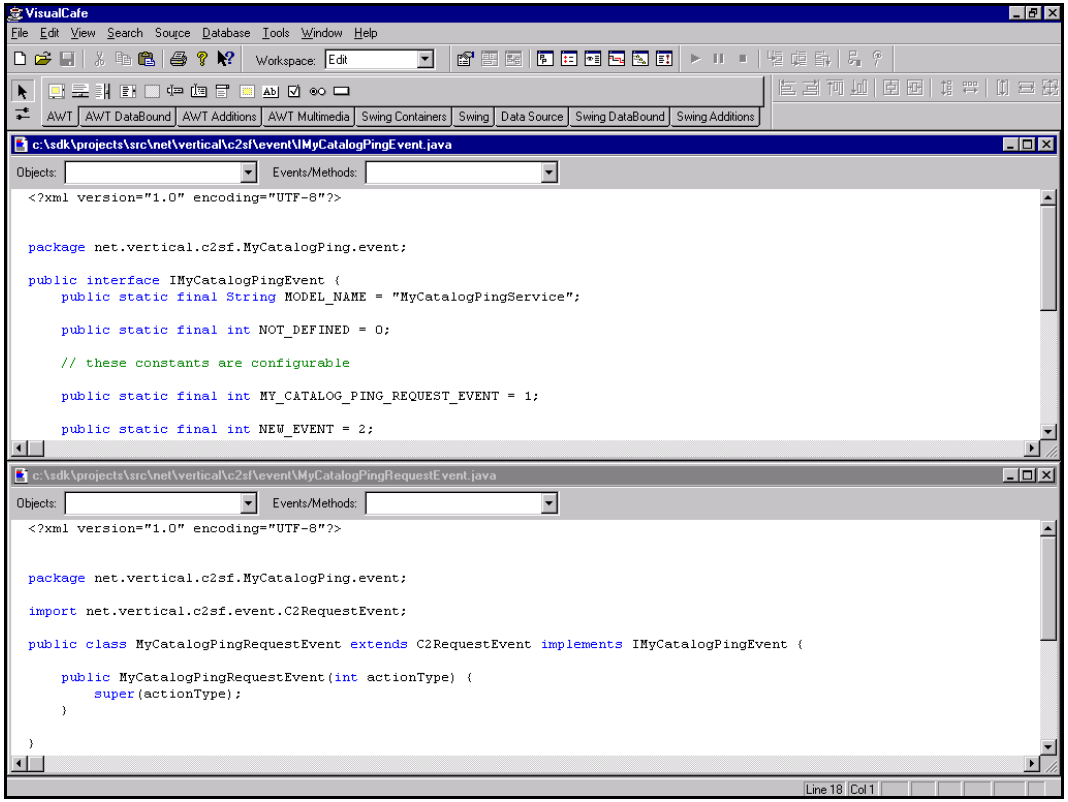


You can also verify that the project file hierarchy has been populated with the generated source file as shown in the following figure.



Finally, you can examine the generated source code to see the implementation of the specifications that you made in the New Service Wizard.

The following figure shows the source code that was generated for the MY_CATALOG_PING_REQUEST_EVENT.



Here you can see the constant token that was defined for the event, and the handler method that will process the event when it is received by the MyCatalogPing service component.

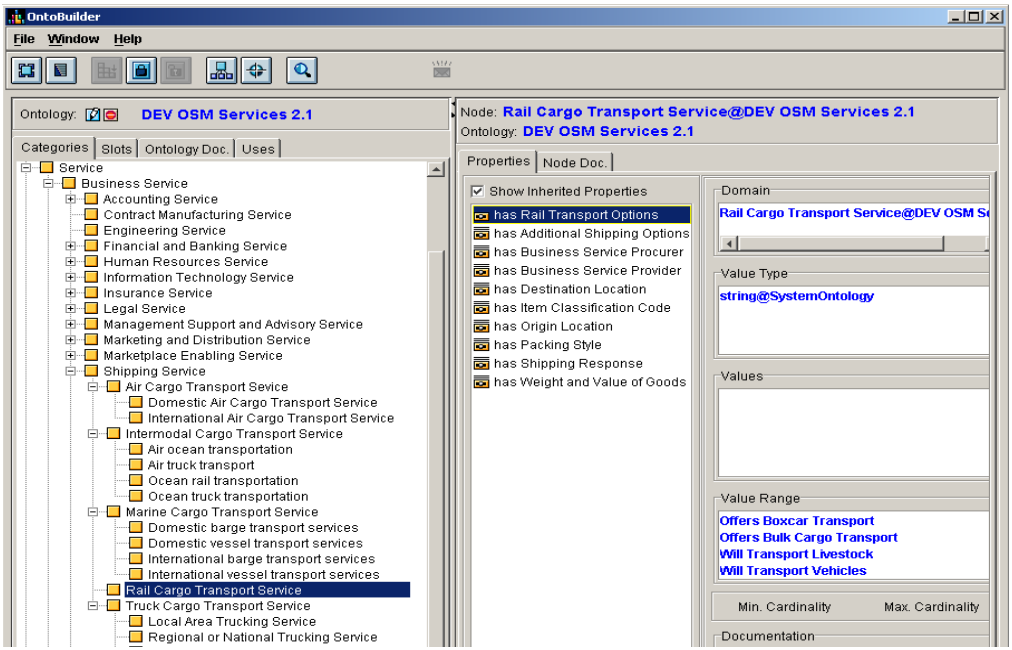
Chapter 3: Service Specification

In this chapter, we will discuss the OSM Services Ontology, and provide information about how to extend the ontology to meet the specific needs of individual marketplaces or enterprises.

Introduction to Service Ontologies

The Service Ontology organizes the information necessary to buy and sell services. All services are sub-classes of the "Business Service" class (see figure). A business entity that *provides* a given services is known as a Business Service Provider. Likewise, business entity that *procures* a service is known as a Business Service Procurer.

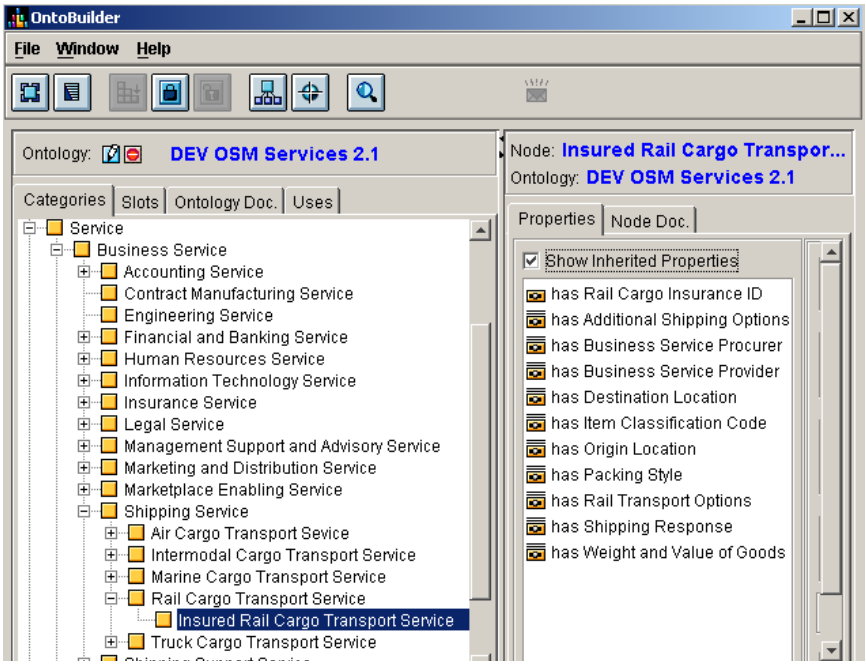
As illustrated below, the OSM ontology utilizes an "indented text view" to illustrate parent-child relationship between classes. A child is also known as a **subclass** of its **parent** class. For instance, here we see that the class `Rail Cargo Transport Service` is a subclass of the class `Shipping Service`.



Each class generally has associated with it one or more attribute groups that describe the nature of the service These attribute groups are listed in the "Properties" sub-window. For example, `Rail Cargo Transport Service` has an attribute group called "`hasRailTransportOptions`." As figure 1 shows, `Rail Cargo Transport Service` is associated with a total of ten different property groups. Some of these properties are defined at the specific class level (e.g. `hasRailTransportOptions` is specific to the class `Rail Cargo Transport Service`), others are defined higher up in the taxonomy and inherited from the parent classes (e.g. `hasOriginLocation` is defined as an attribute of the class `Shipping Service`). Select the "**Show Inherited Properties**" checkbox to toggle whether all inherited properties for a class are displayed or only those defined specifically for that class.

Extending the Service Ontology

The OSM Service Ontology can be extended to accommodate services relevant to a particular market or enterprise. When adapting the ontology to fit your needs, to encourage interoperability with other marketplaces, it is good practice to do so by adding additional subclasses and attributes rather than deleting or modifying existing classes.



Creating A New Subclass

To create new service classes, proceed by finding the class in the tree that most closely matches the service you with to add. For instance, suppose that you need a new subclass of the class `Rail Cargo Transport Service` specifically to accommodate insured cargo. We will call this new class "`Insured Rail Cargo Transport Service`" and add an additional "`insuranceID`" property.

Begin by right-clicking on the `Rail Cargo Transport Service` class, and select "**Lock**" from the pop-up menu. Right -click on this node again, and this time select "**Add Child(ren)...**" Give the new child node a meaningful name, in this case "`Insured Rail Cargo Transport Service`". Click Add to close the dialog box, and you will now have a new subclass (see Figure 2).

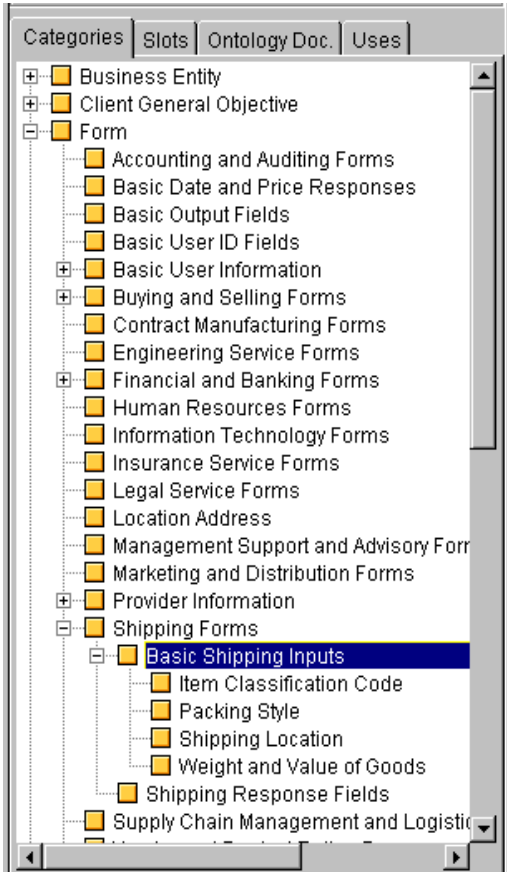
Creating A New Attribute Group

Now we will add an attribute indicating whether the cargo has a rail insurance identification number. To achieve this, we must first create a new attribute group, add a new attribute to this group, and then link the attribute group to the desired service.

Begin by creating the new attribute group. Create this group in a logical place, for example under the `Basic Shipping Inputs` folder, which you will find in the Form folder, in the "`Shipping Forms`" sub folder (see Fig 3).

Create a new subclass by right clicking on this node, locking it, right-clicking again and selecting "**Add Child(ren)…**". In this case, name the new subclass "Insurance Parameters".

Now we will attach an attribute to the Insurance Parameters group. Proceed by right-clicking Insurance Parameters class, and select "**Lock**." Right-click again and choose "**Create Slot(s)…**"

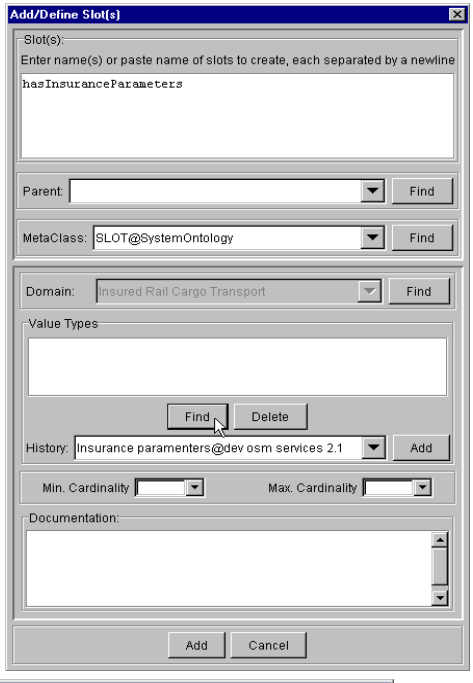


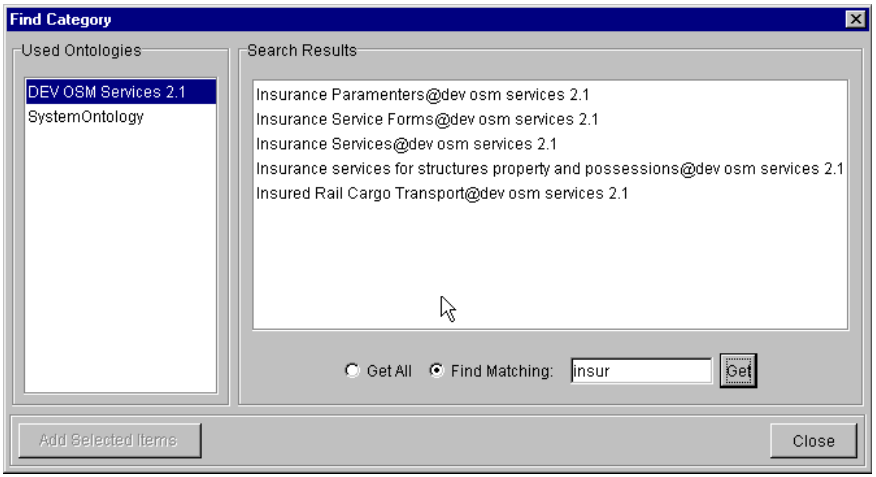
Now enter a name such as “hasInsuranceID” and click on the “**Add**” button. You can confirm that the slot has been added by clicking on the node.

Finally we need to link the new attribute group to our “`Insured Rail Cargo Transport Services`” class. Right-click on the `Insured Rail Cargo Transport` node, and select “**Create Slot(s)…**”. Name the new slot “`hasInsuranceParameters`”. Now click the “**Find Button**” in the Value Types” section of this dialog box (see Figure 4).

Choose the “**Find Matching:**” radio button in the Find Category dialog box (see Fig 5 below), and enter the first few letters of the target form group (in this case “Insurance Parameters”). Then click the “**Get**” button. From the resulting list, select “Insurance Parameters@dev osm services 2.1” Click the **Add Selected Items** button, and then click the **Modify** button of the Add/Define Slot(s) dialog box.

Additional attributes can be added at any time to the Insurance Parameters group to further define the Insured Rail Cargo Transport Service.





Chapter 4: Business Process Specification

In this chapter, we look at how to encode the business logic for your marketplace application by specifying a library of business process flows.

Motivation

In Chapter 2, we saw how OSM tools enable a marketplace builder to cleanly separate the presentation layer from the underlying business logic. When a particular interface button is pushed or form filled, the appropriate response can be calculated using logic coded in Java beans, potentially calling out to the OSM Service Coordination Management components to include services defined by the OSM Service Registry.

For some tasks, however, encoding business logic as a process flow instead of compiled code can offer enhanced flexibility. A general purpose flow can easily be tailored by adding or removing a step or redefining how the step should be executed. The modified process flow can be saved into the process library as a new instance appropriate to some user or specific class of users. This offers a very modular methodology for creating a solution that offers "different strokes for different folks" instead of a "one-size fits all" approach.

Business Process Tools

Commercial business process tools typically provide the following components:

- A graphical process editor to enable a business analyst to intuitively and rapidly design process and data flows for the system.
- A business process execution engine that interprets the designed process flows. Some products are capable of insuring transactional integrity of the executed process.
- A means of monitoring the status of running processes.

The OSM platform has been designed to minimize dependencies on a particular business process vendor. What is required is that the presentation layer can trigger the execution of a process from among those defined in the process library, and that a sub-step in the process can delegate a request to the OSM Service Coordination Advisor and other management components.

The OSM platform currently comes out of the box with connectors for Microsoft's BizTalk Orchestration product, and a future release will add connectors for the HP Process Manager and BEA's Process Integrator.

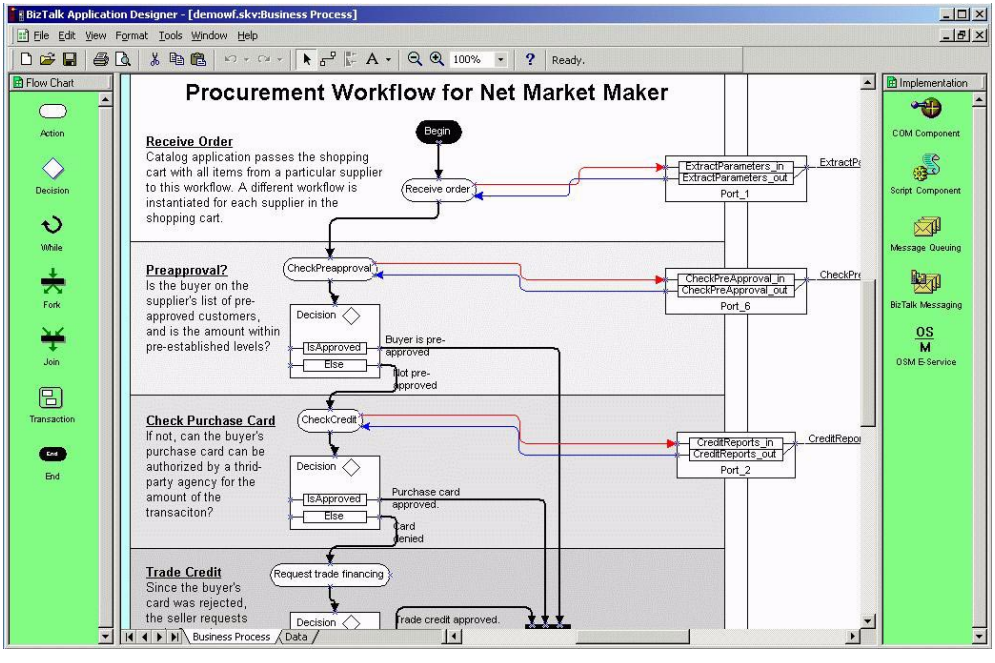
The following documentation provides an overview of how to use MS BizTalk Orchestration in the context of the OSM Platform. For more detailed documentation on the operation of BizTalk Orchestration, please consult its accompanying documentation.

Microsoft BizTalk Orchestration

Microsoft BizTalk Orchestration, also referred to as the BizTalk Application Designer, offers a graphical process editor based on Microsoft's Visio 2000 product. Process drawings are compiled into an XML format called XLANG which can then be interpreted by BizTalk's process manager.

Executing Orchestration

To execute Microsoft BizTalk Orchestration, choose "BizTalk Application Designer" from the "Start | Programs | Microsoft Biztalk Server 2000" menu on your Windows task bar. Orchestration's process designer, based on MS Visio, will appear as illustrated below.



Designing a Process Flow

A business analyst uses Orchestration's Visio tool to create or modify a process flow. To do this, he or she creates a business process drawing by dragging and dropping flowchart shapes into the workspace and connecting them to indicate the appropriate flow logic. MS Orchestration provides several primitive shapes from which to construct process flows:

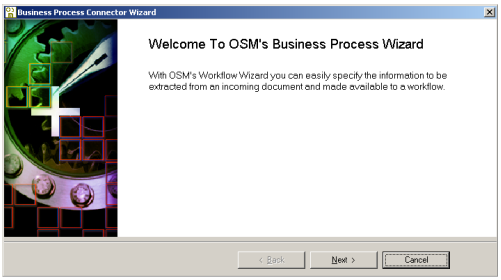
- *Action*: Actions can be bound to "ports" which are responsible for the execution of the action step. An action may call COM objects, windows scripts, send a message using BizTalk or MSMQ, or can be bound to a service defined in the OSM Service Registry.
- *Decision*: Decision implement "if-then" branching rules.
- *While*: While shapes implement looping logic.
- *Fork*: BizTalk Application Designer supports concurrent actions. Use the fork shape to create a concurrent branch in the process flow.
- *Join*: Join shapes unify any branched actions that are not terminated by an End shape.
- *Transaction*: Using the transaction shape, portions of the workflow can be segmented into groups for which all steps much succeed if the transaction block is to succeed. Aborting any step will abort the entire block.
- *Begin*: the starting node for the process flow. Only one begin node may exist.
- *End*: A termination node. Multiple end nodes may exist in a flow.

OSM Business Process Connection Wizard

The first step in a workflow should specify the input arguments that will come into the workflow. To help in this process, OSM provides a Business Process Connection wizard (BPCW), which accepts arguments from the workflow designer and then generates the necessary connection scripts required for integrating MS Orchestration with the presentation layer.

Running BPCW

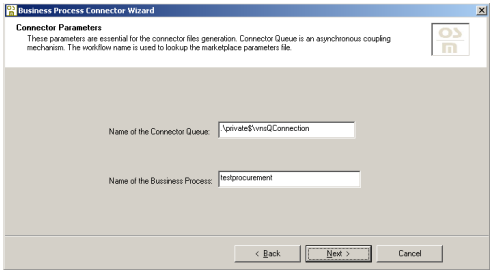
To execute BPCW, run BPCW.exe from the **bin** directory of the OSM distribution. To move forward through the steps defined by the Workflow Connection Wizard, use the **next** button on the lower panel. To move backwards and re-edit information from a previous frame, use the **back** button. **Cancel** aborts the process.



Specifying Connector Parameters

For each business process created, two parameters are required:

- The name of the MSMQ connector queue on which to send input.
- The name of the business process. This name should uniquely identify your process on your system.



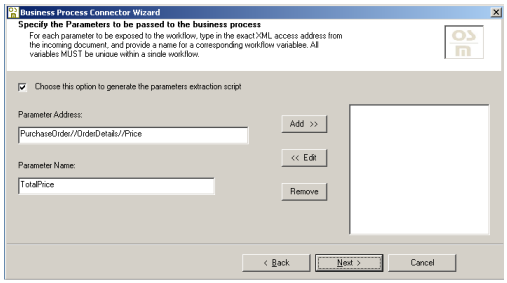
Once you have entered suitable values for these parameters, press the **next** button.

Defining Input Arguments

The next panel allows you to input the parameters to extract from an incoming request message that will be used by the BizTalk Orchestration Workflow. For each parameter to be exposed to the workflow, type in the XPATH information specifying how to identify the information from an incoming XML document, and provide a name for a corresponding workflow variable, then hit **Add**. If you want to edit any of them, select the parameter and press **edit**. When you select a parameter for editing, it is removed from the parameters window and added to the input windows. Make sure that you do an add after you finish editing. You can also remove an existing parameter using the **remove** button.

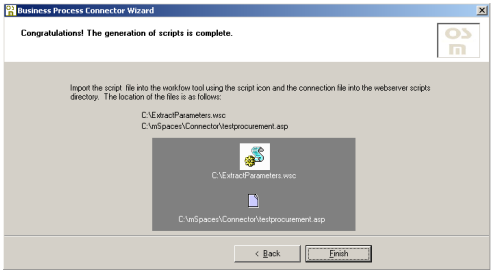
Variable names can be the same as tag names but are not required to be so. All variable names must be unique within a single workflow.

There is a choice button near the top of this panel that allows you to turn on or turn off the extract parameters script generation. The connector ASP file is always generated.



Adding Definitions to your Workflow

Hit the **next** button to generates the files. The pathnames of the generated files is shown on the panel. Move the ASP files to the scripts directory under your web server directory. For example, on this machine it is “C:\Inetpub\scripts”. The Windows Scripting File needs to be dragged into your workflow specification. For more information on this see the documentation on setting up workflows in BizTalk Orchestration.



Connecting Actions to Ports

Conceptual action steps need to be bound to a concrete implementation of the action. Do do so, BizTalk Orchestration provides four built-in component types that each specify what Microsoft terms a "port"; these component types include:

- COM component: access functions stored in compiled components on your Windows machine.
- Windows Script: a component implemented in Visual Basic, best used for calculating simple values.
- BizTalk Message: BizTalk messaging provides a reliable framework for sending messages that can be translated to various formats
- MS Message Queue: provides loosely-coupled and reliable network communications based on a message queuing model.

To bind an action to one of the above component types, drag and drop an instance of the component type onto your workspace. A wizard will popup, guiding you through the process of entering the information required by the component type. The result will be a Port shape that can be connected to your action step using the MS Orchestration connector tool.

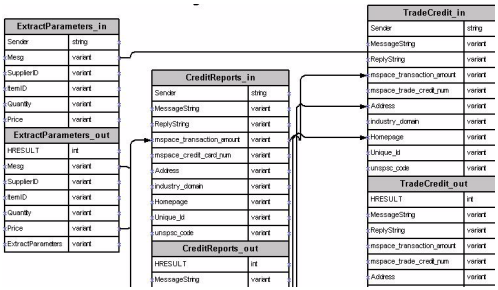
OSM Service Coordination

Ports created using the built-in connector types provided by MS Orchestration are useful for integrating a workflow with existing systems such as databases or ERP systems. However, it is often advantageous to bind the action step not to a known system, but rather to delegate the execution of the action to external service providers defined in an OSM service repository. Doing so allows the possibility of defining business rules to select the appropriate partner for the current context or user, and enables an enterprise to reconfigure business partners over time without having to recode business processes to accommodate a new partner's APIs.

To bind an action step to a service provided by external business partners defined in the OSM Service Registry, add an "OSM E-Service" to your workflow by dragging and dropping the icon into your workspace. As detailed in Chapter 5, "Service Coordination", an OSM Service Selection wizard will guide you through the steps of defining the required information for your new service.

Designing Data Flow

The Data page contains message shapes, each with a unique name, that correspond to one or more Port Message shapes that are contained within the Port shapes on the Business Process page. The Data page represents the flow of all the messages used by the scheduler during its lifetimes. Using the Data page, a process designer must connect the appropriate input and output fields to specify such a flow.



Compiling the Business Process

Before the process can be run, it must be first compiled. From the “File” menu, choose “Make XLANG Filename”. As the process is compiled, any errors will be displayed. Once the process has been successfully built, you will be ready to insert the process into your marketplace application.

Connecting the Presentation Layer to Process Logic

Once a business process has been constructed and compiled, you can access from Java (e.g., from the presentation layer) using the connector class *WorkflowConnector*. The constructor of the class is initialized with the ID for the workflow (as specified in section “Specifying Connector Parameters” on page 33) and the main functionality for the class transmits an XML message to use as input for the flow.

Using the ID for the workflow, the corresponding URL where the workflow listener has been created is obtained. The file OSM.config is searched for entries of the form *workflowURL.xxx=http://yyy*, where *xxx* is the id of a workflow and *http://yyy* is a URL via which the workflow is triggered: an appropriate XML string is posted to this URL, and it is assumed that this XML is then forwarded on to the workflow.

For example, if the config file contains the entry

```
workflowURL.osm1432=http://localhost:90
01/scripts/czone.asp
```

then the workflow started by this .asp page can be started as follows:

```
WorkFlowConnector wflow = new
      WorkFlowConnector("osm1432");
String reply =
   wflow.startorkFlow(
     "<BuyerOrder> ... </BuyerOrder>")
```

Chapter 5: Service Coordination

In this chapter, we shall walk through the Service Selection wizard frame by frame, explaining how to use the tool to define a delegated service within an OSM business process.

Executing Service Select

To execute Service Select, simply double click on its icon in the OSM Folder. The Service Select wizard appears as illustrated below.

Navigation

Service Select uses a wizard metaphor for simplified use. To move forward in the process, use the **next** button on the lower panel. To move backwards and re-edit information from a previous frame, use the **back** button. **Cancel** aborts the selection process.

Welcome Panel

The purpose of the first panel is simply to welcome you to the OSM Service Select wizard. To begin the service selection process, press **next**.

Choosing a Service Type

From this panel, you should choose a service type from among those listed in an OSM Service Ontology.

Browsing the Service Ontology

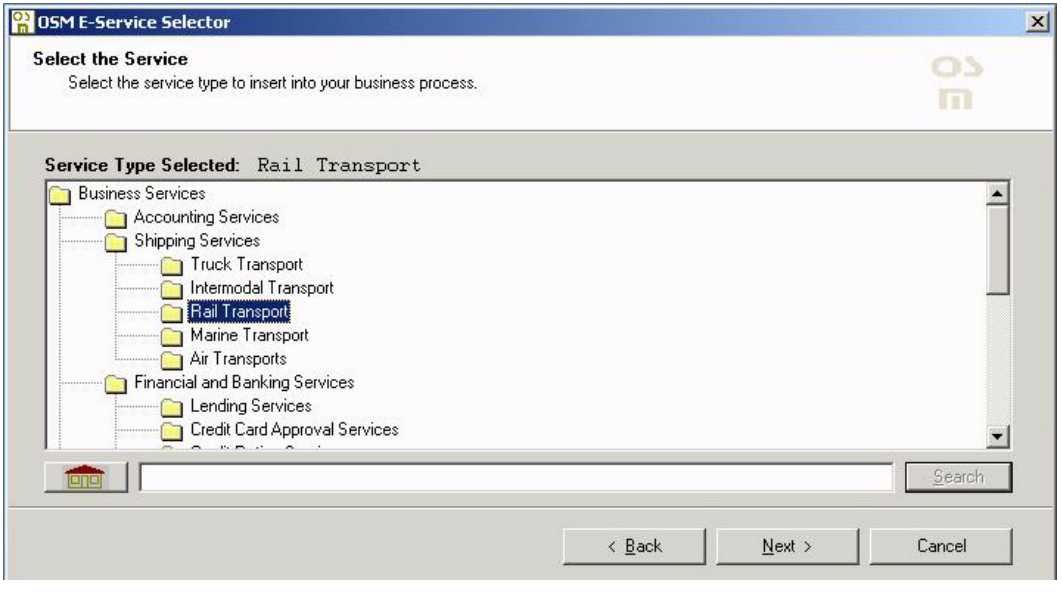
Services types in the OSM Service Ontology are listed in a hierarchical structure, starting with the root "Business Services", and gradually becoming more specialized. One way to find the appropriate service type for your new service is to click on nodes in the tree, opening up the services folders to browse successively more detailed levels.

Searching the Service Ontology

For faster access, you may want to search directly for your service type using keywords. Type a keyword into the provided textfield and press the **search** button. A dialog box appears with all service types containing your term. Clicking on an item causes the main service browser to position the choice as the root node, where you may continue browsing. To reset the browser to the "Business Services" root, click on **home**.

Selecting a Service Type

As you click on service types displayed in the browser, the field "`Service Type Selected`" updates accordingly. Press **next** when you are finished specifying a service type.



Specifying Delegation Policies

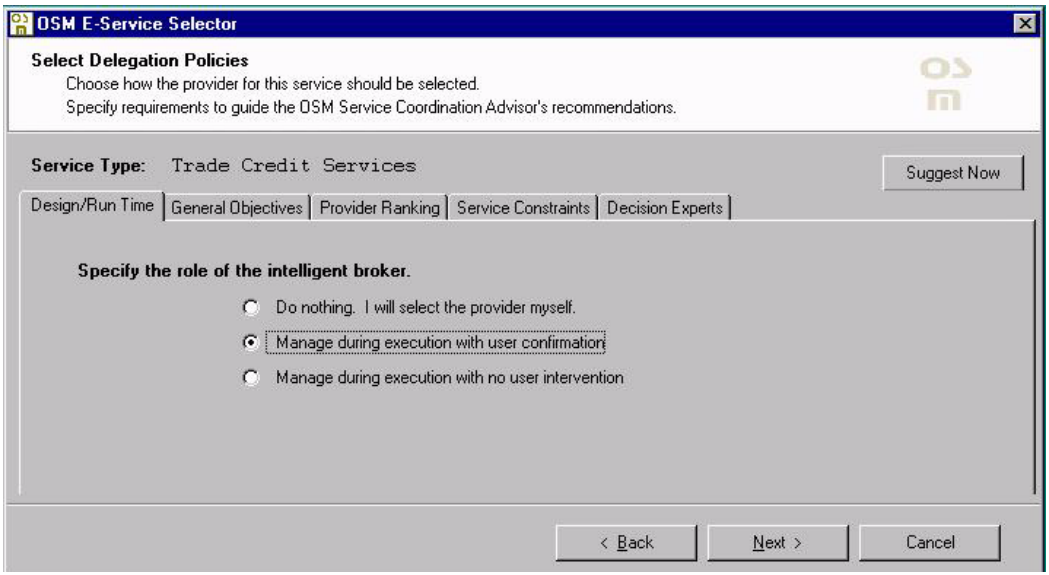
Service Coordination Advisor

The Service Coordination Advisor (“Advisor”) is a tool that makes recommendations about which service provider from among the business partners listed in the Service Repository would optimally fit the needs defined using the Service Select wizard. The Advisor can be thought of as an extensible, distributed, business rule engine that fluidly combines requirements set by the market-place, recommendations from numerous sources including domain-relevant third party rating services, and policies from the workflow designer.

Design vs. Run-Time Selection

In the Design/Run-Time panel, the user can specify the role of the Service Coordination Advisor. The Advisor can be disabled for this service by selecting the option “Do nothing, I will select the provider myself”, in which case the user will be presented with a list of service providers from which to choose. By clicking the “Suggest Now” button, the Advisor will make a suggestion based on input criteria, as described in section “Reviewing Recommended Providers” on page 44. From the resulting panel, the suggested provider can be set as the design-time choice by clicking the “Use Suggestion” button.

Maximum flexibility and robustness can be obtained by enabling the OSM Advisor to make runtime decisions, either with or without user confirmation. If a preferred provider cannot



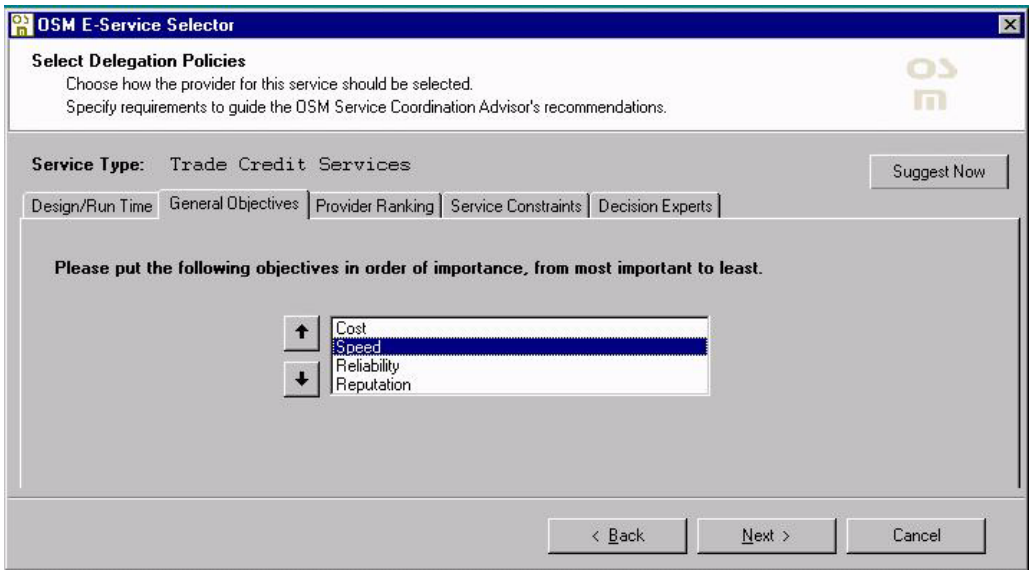
accomplish a task for some reason, the Advisor will be able to suggest a reasonable alternative and take appropriate action.

Prioritizing General Objectives

Individual tabs in the "Select Delegation Policies" panel enable a user to enumerate constraints and preferences for how service providers should be recommended by the Service Coordination Advisor.

In the "General Objectives" tab, a list box displays a number of domain-independent objectives, such as cost, speed, reliability, reputation, and others. This list is obtained by querying the Services Ontology.

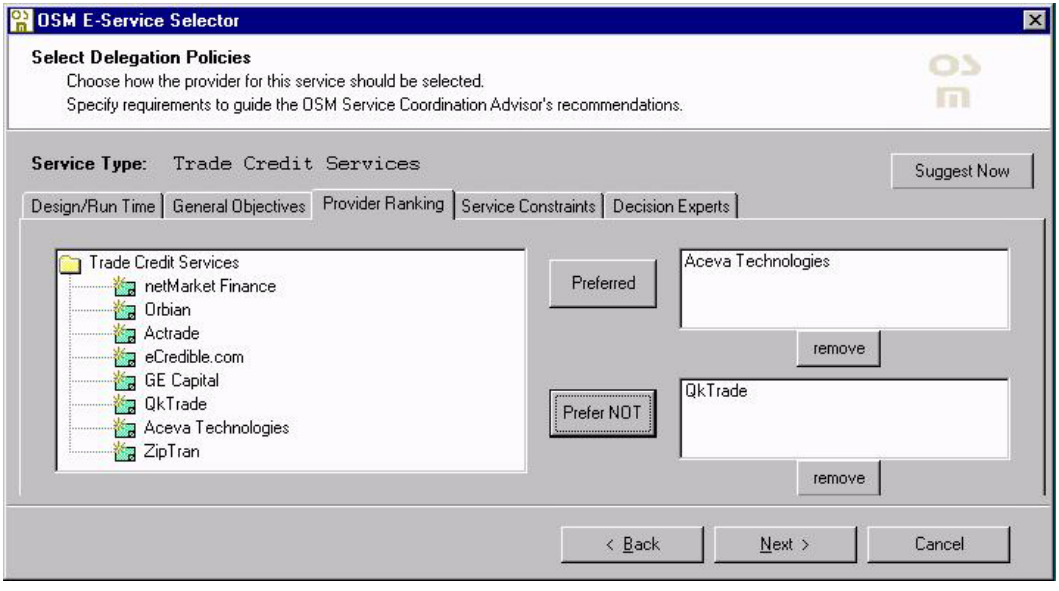
Using the arrows, prioritize the relative importance of the objectives by selecting an objective in the list box and raising (up arrow) or lowering (down arrow) its ranking. Objectives at the top of the list will be highly considered by the Advisor, whereas objectives at the bottom will not impact the Advisor's suggestions much at all.



Preferred Providers

In the “Provider Ranking” tab, a user can specify preferences, or weights, either positive or negative, that will impact the provider’s rankings as suggested by the Service Coordination Advisor.

Browse the list of available service providers for the selected service type. After selecting one in the list box on the left, add the provider to the “Preferred” or “Not Preferred” sections by using the appropriate buttons. If a provider is classified in error, you may remove them from the classification lists by selecting provider and clicking **remove**.



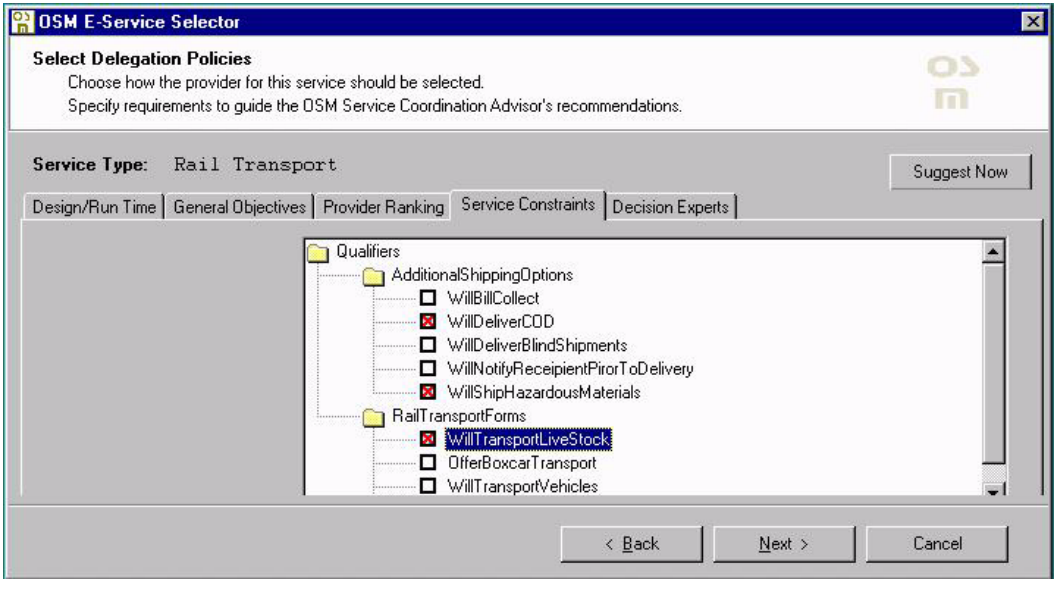
Filtering using Service Qualifiers

For certain service types in an OSM ontology, service qualifier attributes will have been defined. These attributes are used by the marketplace to filter prospective service providers down to those who most aptly suit a given request. Service qualifiers are specific to a service type: an example of a qualifier for a shipping service might be “Will ship hazardous materials.”

When service qualifiers are defined for the selected service type, the “Service Constraints” tab will contain a list box that enables a user to specify required attributes for their prospective service providers. If no qualifiers have been defined for the service type, a text message indicating this fact will be displayed instead.

Examine all service qualifiers indicated by the panel, and consider whether the property indicates a feature that is essential to fulfill your service need. Clicking on the check box next to each qualifier will toggle the value from not selected to selected, meaning the attribute is a requirement to meet your request.

For any selected service qualifier, only those providers who have indicated they can perform this capability will be considered in the selection process.



Prioritizing Decision Experts

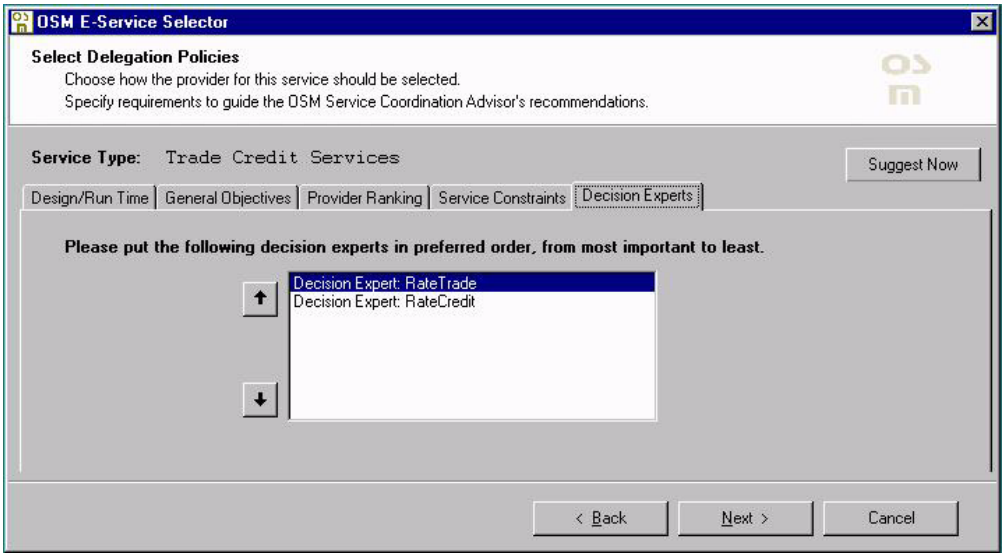
The role of the Service Coordination Advisor is to provide intelligent recommendations when appropriate about which service provider from among those in the OSM Service Registry would best meet requirements of the current situation.

To come up with its conclusions, the Advisor incorporates business rules elaborated by a community of "decision experts", each expert providing expert options and rationale about the pros and cons of various providers in various situations. Decision experts can be selectively added by the marketplace to enforce policies and promote guidelines for how partner selection should occur.

There are many types of decision experts. Samples include:

- Individual user preferences
- Heuristic-based or learning-enabled rule engines
- Third-party rating or credentialing services
- Market-based mechanisms such as reverse auctions, structure negotiation, and so forth.
- System level goals, such as balancing workload among multiple providers.

All decision experts may not be equally useful or relevant for all situations. Using the "Decision Expert" tab, the operator of the Service Select wizard can set the relative importance of available decision experts for the current task request. To do so, select a decision expert in the list box raise (up arrow) or lower (down arrow) its relative ranking. Recommendations from experts at the top of the list will be highly considered by the Advisor, whereas those from experts at the bottom will not impact the Advisor's suggestions much at all.



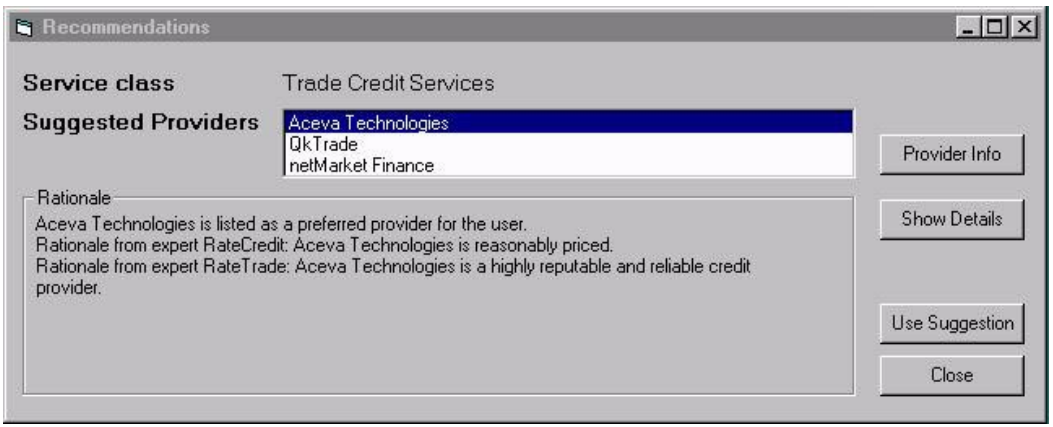
Reviewing Recommended Providers

Once all objectives, preferences, and constraints have been entered using the five configuration panels, you are ready to explore the recommendations offered by the Service Coordination Advisor that best suit your requirements. By clicking on the **Suggest Now** button, the recommendations panel is displayed, containing the Advisor's suggestions.

Recommended providers are presented in a list box in order of suitability, from highest recommendation on down. By clicking on a listed provider, the rationale behind the recommendation is displayed. Clicking on the **Show/Hide Details** button toggles the rationale presentation from brief mode (the default) to expanded detail mode, which provides the numerical values behind the calculated recommendation.

To perform additional due diligence, click on the **Provider Info** button. If information is available for the selected provider, it will be shown in a web browser window.

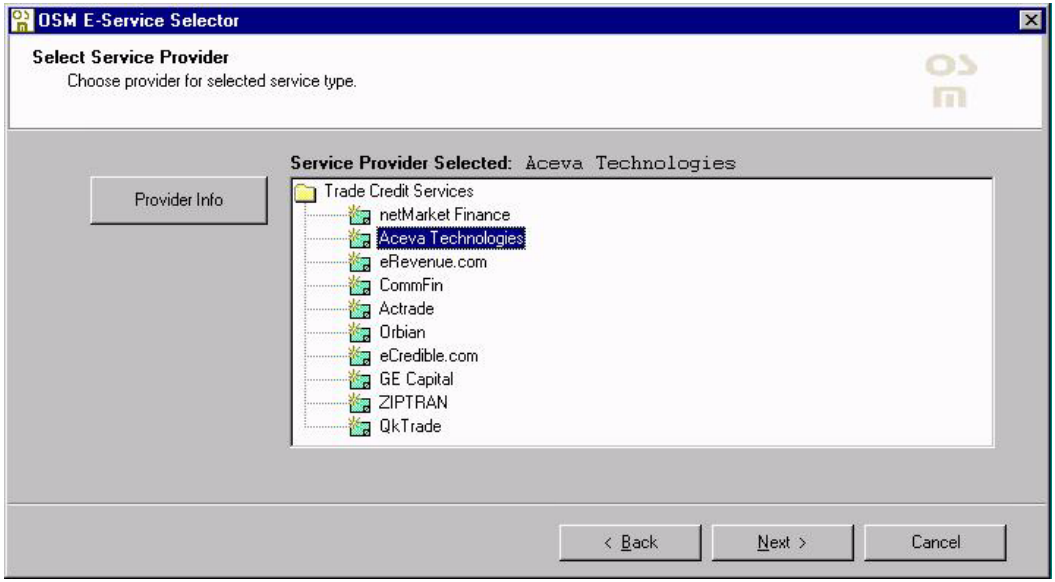
At design time, the recommendations of the Service Coordination Advisor can be considered a research tool. As discussed in section "Design vs. Run-Time Selection" on page 39, to specify that a given service provider should be used as the primary contractor for a service, highlight a provider and click on the **Use Suggestion** button.



Designating a Primary Provider

As described in “Design vs. Run-Time Selection” on page 39, the user of Service Select can specify whether or not the selection of the service provider should be managed by the Advisor. If the “Do nothing, I will select the provider myself” option is selected, the “Select Service Provider” panel will be displayed as illustrated below.

Browse the list of available providers for the selected service type. You may view detailed web-based information by selecting a provider and clicking on the **Provider Info** button. As you click on a provider in the list, they will become the selected provider, indicated by the label above the selection list box. When your final choice has been made, click on the **next** button to proceed with the Service Selection process.



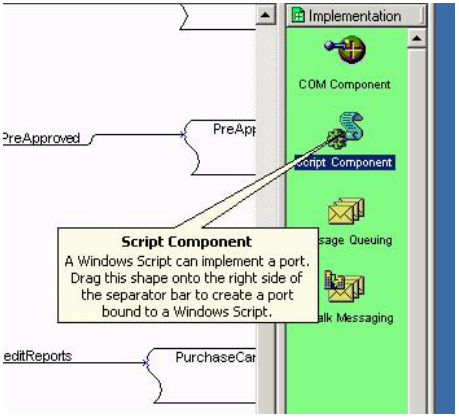
Workflow Integration

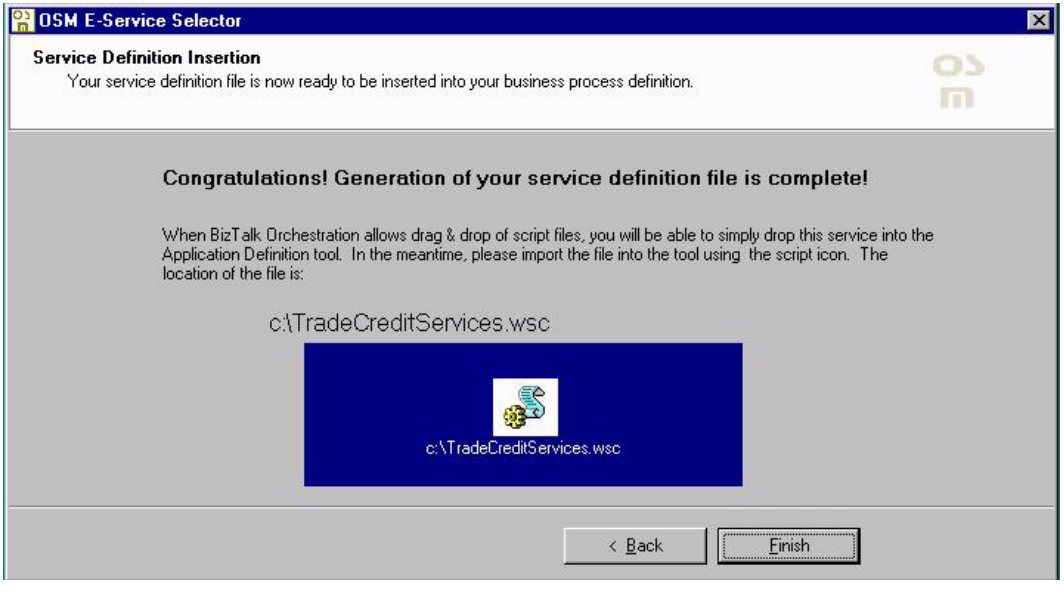
You have now completed the process of specifying constraints and requirements for your service request. The Service Select wizard generates a windows script file containing the directives you have specified and displays this as an icon in the wizard panel. What remains is to insert this script into your process workflow, binding it to the appropriate sub task step.

Inserting Service Request in BizTalk Orchestration

The current version of Microsoft BizTalk Orchestration does not support drag & drop of a script file, although this is documented as a feature for a future release. When this feature is added, you will simply be able to drag the script icon into your workflow and connect it through the connector tool.

Until this feature is supported, you should use Orchestration's Window Script import icon:





When you execute the Windows Script icon in MS Orchestration, a wizard will pop up. Answer the questions as follows:

1. Allow a new port to be automatically generated, as proposed. Press **next**.
2. Choose the option “Use the XLANG Scheduler Engine” to instantiate the script component. Press **next**.
3. Type in the path (or browse to the path) of the file generated by the OSM Service Select wizard. Click **next**.
4. Accept the proposed option “Use a moniker of the script file.” Click **next**.
5. Set your security level. Click **finish**.

The script and its associated port will now be added to the workflow. Bind the new port to the desired process flow step using the connector tool, and your integration will be complete!

Glossary

API See Application Programmers Interface.

Application Programmers Interface (API) A programmatic interface exposed by a service that provides invoking of the service from external applications.

ASP See Active Server Pages.

Active Server Pages (ASP) A technology from Microsoft for creating web pages that contain scripts, or “active content.”

BPCW See Business Process Connection Wizard.

Business Process Connection Wizard (BPCW) A tool to simplify the creation of a new business process for OSM, specifying the required inputs for the process.

C2/SF See Cooperative Commerce Service Framework.

C2/SF Component Builder A wizard that allows you to rapidly build a new service for integration within the C2/SF framework.

class A class is a representation for a conceptual grouping of similar terms. For example, a computer could be represented as a class that has many subclasses such as personal computers, mainframes, workstations, etc. Each class is described by a definition that specifies the slots and values that describe the class itself (not the members of the class), slots and values that describe the instances of that class, and logical statements (called axioms) that describe the class but can't be represented using slots and values.

Cooperative Commerce Service Framework (C2/SF) A framework for building electronic marketplace applications that leverages standard middleware technology to provide an integration environment and presentation layer for high-level business applications.

Decision Experts A decision expert provides expert options and rationale about the pros and cons of service providers in various situations. The Service Coordination Advisor’s job is to combine information from multiple decision experts to formulate a composite recommendation for the current context.

Microsoft Messaging Queue (MSMQ) Software from Microsoft that provides guaranteed message delivery functionality.

MSMQ See Microsoft Message Queue.

Ontology An ontology is an explicit, formal and declarative specification that provides a vocabulary (terms) and a set of relationships for representing and communicating knowledge about some topic.

Ontology Builder Aids the market builder in constructing ontologies through which services and processes are structured.

Open Services Marketplace (OSM) Platform An integration platform for B2B public and private markets based on advanced technologies for dynamic services coordination.

OSM Platform See Open Service Marketplace Platform.

Properties Attributes of a class that are defined by a slot.

Service Coordination Advisor An extensible rule-based expert system that helps manage optimal business partner selection.

Service Execution Engine A high-performance server that manages reliable, scalable communications with remote business services.

Service Registry An ontology-based, UDDI-compliant directory of web-based services.

Service Qualifiers Service qualifier attributes are used by the marketplace to filter prospective service providers down to those who most aptly suit a given request. Service qualifiers are specific to a service type: an example of a qualifier for a shipping service might be “Will ship hazardous materials.”

Slot A slot is used to describe a relationship between two terms. A slot is sometimes referred to as a binary relation.

Subcategory See subclass.

Subclass The child of an existing class.

Universal Description Discovery and Integration (UDDI) The UDDI consortium, consisting of more than 100 companies including Microsoft, IBM, Ariba, HP, and VerticalNet, are working to standardize the registration of business and web services. VerticalNet Solution’s OSM uses UDDI-compliant registries to store its registry information. `http://www.uddi.org`

XLANG Microsoft BizTalk Orchestration’s format for storing business process representations.

XSL eXtensible Stylesheet Language. A stylesheet format for XML documents.

XSLT XSL Transformations. The language used for converting XML documents into other XML documents. May be used independently of XSL.

Contacting VerticalNet

VerticalNet Solutions
301 Howard Street
Suite 1410
San Francisco, CA 94105

Tel: (866) 515.2040
Fax: (415) 995-9783
E-mail: contact_solutions@verticalnet.com
Web: http://www.verticalnetsolutions.com

Index

U

V

W

X