



SAS Publishing



# **SAS<sup>®</sup> 9.1.2 Integration Technologies**

## **Administrator's Guide**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS® 9.1.2 Integration Technologies: Administrator's Guide*. Cary, NC: SAS Institute Inc.

### **SAS 9.1.2 Integration Technologies: Administrator's Guide**

Copyright © 2002-2004, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**U.S. Government Restricted Rights Notice:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19, Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

May 2004

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/pubs](http://support.sas.com/pubs) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.

# Table of Contents

<b>SAS® 9.1.2 Integration Technologies: Administrator's Guide.....</b>	<b>1</b>
<b>Getting Started.....</b>	<b>2</b>
<b>SAS Foundation Services.....</b>	<b>4</b>
<b>Understanding Service Deployments.....</b>	<b>9</b>
<b>Understanding Service Deployment Configuration.....</b>	<b>12</b>
<b>Defining Service Deployments.....</b>	<b>13</b>
<b>Importing Service Deployments.....</b>	<b>16</b>
<b>Exporting Service Deployments.....</b>	<b>17</b>
<b>Duplicating Service Deployments.....</b>	<b>18</b>
<b>Redistributing Service Deployments.....</b>	<b>19</b>
<b>Understanding How Applications Deploy Foundation Services.....</b>	<b>20</b>
<b>Understanding How Applications Locate Foundation Services.....</b>	<b>22</b>
<b>Scenario: Stand-alone Application.....</b>	<b>24</b>
<b>Scenario: Remote-accessible Services.....</b>	<b>26</b>
<b>Scenario: Local and Remote-accessible Services.....</b>	<b>28</b>
<b>Understanding How Applications Share Foundation Services.....</b>	<b>30</b>
<b>Modifying Service Configurations.....</b>	<b>31</b>
<b>Understanding the Event Broker Service.....</b>	<b>32</b>
<b>Understanding Events and Process Flows.....</b>	<b>35</b>
<b>Modifying an Event Broker Service Configuration.....</b>	<b>39</b>
<b>Creating Events and Process Flows.....</b>	<b>40</b>
<b>Modifying the Information Service Configuration.....</b>	<b>42</b>
<b>Modifying the Logging Service Configuration.....</b>	<b>46</b>

# Table of Contents

<b>Pattern Layouts.....</b>	<b>48</b>
<b>Modifying the Session and User Service Configurations.....</b>	<b>49</b>
<b>Monitoring Applications.....</b>	<b>52</b>
<b>Stored Processes.....</b>	<b>53</b>
<b>Defining Servers Used By Stored Processes.....</b>	<b>54</b>
<b>Registering a New Stored Process.....</b>	<b>55</b>
<b>Modifying an Existing Stored Process.....</b>	<b>57</b>
<b>Publishing Framework.....</b>	<b>59</b>
<b>Planning Your Publishing Solution.....</b>	<b>60</b>
<b>Managing Subscribers.....</b>	<b>63</b>
<b>Delivery Transports.....</b>	<b>65</b>
<b>Filters.....</b>	<b>68</b>
<b>Managing Channels.....</b>	<b>71</b>
<b>Persistent Stores.....</b>	<b>74</b>
<b>Publishing to Secure Servers.....</b>	<b>78</b>
<b>Example: Creating a Subscriber.....</b>	<b>80</b>
<b>Example: Creating a Channel.....</b>	<b>86</b>

# SAS® 9.1.2 Integration Technologies: Administrator's Guide

This is the Administrator's Guide for SAS Integration Technologies. It is provided for SAS Integration Technologies customers who use the SAS Open Metadata Architecture.

This guide provides detailed instructions for administration of the following SAS Integration Technologies features:

- SAS Foundation Services
- SAS Stored Processes
- SAS Publishing Framework

Many of the administrative tasks can be performed using the SAS Management Console application. SAS Management Console is a graphical user interface that enables you to easily enter and modify metadata on your SAS Metadata Server.

Before you begin the SAS Integration Technologies administration tasks, refer to the [Getting Started](#) chapter for introductory information and references.

Use this Administrator's Guide in conjunction with the following guides:

- [SAS Integration Technologies: Server Administrator's Guide](#) provides detailed information for administering SAS Workspace servers and spawners, and SAS Stored Process servers and spawners. It also provides general information for administering all IOM servers.
- [SAS Integration Technologies Developer's Guide](#) provides details about using SAS Integration Technologies to develop and integrate applications.

**Note:** If you are implementing SAS Integration Technologies by using the Lightweight Directory Access Protocol (LDAP) instead of the Open Metadata Architecture, refer to the [Administrator's Guide \(LDAP Version\)](#).

*Getting Started*

# Getting Started

To utilize the features of SAS Integration Technologies, you must define the appropriate resources on the SAS Metadata Server (SAS Foundation Services, SAS Stored Processes, and SAS Publishing Framework) or in a configuration file (SAS Foundation Services only). You can then access the resource information as required for your implementation. You should have already determined the appropriate authorization (access controls) for the resources that you will define and access. (To understand and implement the Open Metadata Architecture security, refer to the security administration information in the [SAS Intelligence Architecture: Planning and Administration Guide](#)).

To set up and access SAS Integration Technologies resources, see the following topics:

- **SAS Foundation Services.** To implement SAS Foundation Services service deployment configurations, define the service deployment on the SAS Metadata Server (or in an XML file) and then access the service deployment configuration to deploy and access the services:
  - ◆ **Define Service Deployments.** Use the Foundation Services Manager to define service deployments for local and remote SAS Foundation Services. For details, see [Service Deployment Configuration](#).
  - ◆ **Deploy and Access Service Deployments.** Code your applications to retrieve the service deployment configuration from one of the following locations:
    - ◇ SAS Metadata Repository on a SAS Metadata Server
    - ◇ XML file that contains the service deployment configurationCode one application to retrieve the service deployment configuration and deploy and access the services as local services. Code other applications to retrieve the service deployment configuration and access and use the remotely deployed services. For details about local and remote services and coding client applications to deploy and access services, see the [SAS Foundation Services](#) chapter in this guide, the [SAS Foundation Services](#) topic in the *SAS Integration Technologies Developer's Guide*, and the SAS Foundation Services class documentation for [com.sas.services.discovery](#) in the *SAS Integration Technologies Developer's Guide*.
- **SAS Stored Processes.** To implement SAS Stored Process definitions on the SAS Metadata Server, define and then access the stored process definitions:
  - ◆ **Define Stored Processes.** Use the Stored Process Manager plug-in to SAS Management Console to define stored processes. For details, see [Stored Processes](#).
  - ◆ **Access Stored Process Definitions.** Access stored process definitions by running applications or stored processes that connect to the SAS Metadata Server and access stored process definitions. For details, see the [Stored Processes](#) chapter of the *SAS Integration Technologies Developer's Guide*.
- **Publication Channels.** To implement publication channel definitions on the SAS Metadata Server, define the publication channels and then publish or subscribe to the publication channel.
  - ◆ **Define Publication Channels.** Use the Publishing Framework plug-in to SAS Management Console to define publication channels. For details, see [Publishing Framework](#).
  - ◆ **Publish and Subscribe to Publication Channels.** You can access the publication channel definitions on the SAS Metadata Server, and publish and subscribe to the defined publication channels, in two ways:
    - ◇ SAS Integration Technologies Publishing Framework. For details, see the [Publishing Framework](#) chapter of the *SAS Integration Technologies Developer's Guide*.
    - ◇ SAS Information Delivery Portal. For details, see the online Help for the SAS Information Delivery Portal.

## SAS® 9.1.2 Integration Technologies: Administrator's Guide

After you set up your resources, ensure that the appropriate authorization (access control) is specified for the resource definition.

*SAS Foundation Services*

# SAS Foundation Services

SAS Foundation Services 1.1 includes tools to enable application development and service administration for the SAS Foundation Services. Depending on the components you choose to install, SAS Foundation Services 1.1 includes one or more of the following components:

- **SAS Foundation Services**, which is a set of platform infrastructure and extension services for programmers who want to write applications that are integrated with the SAS platform.

For information about coding applications that use the SAS Foundation Services, see [SAS Foundation Services](#) in the Java client chapter of the *SAS Integration Technologies Developer's Guide* and the Java class documentation for SAS Foundation Services.

The following table presents the function and related documentation for each of the SAS Foundation Services:

SAS Foundation Services			
Service	Class Documentation	Function	Related Documentation
<b>Connection Service</b>	<a href="#">com.sas.services.connection.platform</a>	IOM connection management	For details about administering the SAS servers that you connect to with the Connection Service, see the <a href="#">SAS Integration Technologies Server Administrator's Guide</a> .  For development information and coding examples, see <a href="#">Using the Connection Factory</a> in the <i>SAS Integration Technologies Developer's Guide</i> .
<b>Discovery Service</b>	<a href="#">com.sas.services.discovery</a>	locating and binding to deployed services	For details about how applications use the Discovery Service, see <a href="#">Understanding How Applications Locate Services</a> .
	<a href="#">com.sas.services.events.broker</a>		



<p><b>Event Broker Service</b></p>		<p>asynchronous event notification and request management to support dynamic, event-driven processes</p>	<p>For details about editing the Event Broker Service configuration, see <a href="#">Modifying the Event Broker Service Configuration</a>.</p> <p>For information about using the Publishing Framework to generate and publish events, see <a href="#">About Events in the SAS Integration Technologies Developer's Guide</a>.</p>
<p><b>Event Broker Discovery Services</b></p>	<p><a href="#">com.sas.services.events.discovery</a></p>	<p>locates event brokers</p>	
<p><b>Information Service</b></p>	<p><a href="#">com.sas.services.information</a></p>	<p>repository federation, searching repositories, a common entity interface, and creating personal repositories</p>	<p>For details about editing the Information Service configuration, see <a href="#">Modifying the Information Service Configuration</a>.</p>
<p><b>Logging Service</b></p>	<p><a href="#">com.sas.services.logging</a></p>	<p>runtime execution tracing, response metric and resource utilization reporting, and error tracking.</p>	<p>For details about editing the logging service configuration, see <a href="#">Modifying the Logging Service Configuration</a>.</p>
<p><b>Publish Service</b></p>	<p><a href="#">com.sas.services.publish</a></p>	<p>access to the publication framework</p>	<p>For details about configuring and administering channels and subscriptions for the Publishing Framework, see the <a href="#">Publishing Framework</a></p>

			<p>chapter in the <i>SAS Integration Technologies Administrator's Guide</i>.</p> <p>For information about using publish and subscribe software components and SAS CALL routines, see the <a href="#">Publishing Framework</a> chapter in the <i>SAS Integration Technologies Developer's Guide</i>.</p>
<b>Security Service</b>	<a href="#">com.sas.services.security</a>	<p>user authentication, propagation of user identity context across distributed security domains, and protected-resource access policy administration and enforcement</p>	<p>For detailed information about implementing security in your environment, see the <a href="#">Security</a> chapter in the <i>SAS Integration Technologies Server Administrator's Guide</i>.</p>
<b>Session Service</b>	<a href="#">com.sas.services.session</a>	<p>context management, resource management, and context passing</p>	<p>For details about editing the session service configuration, see <a href="#">Modifying the Session and User Service Configurations</a>.</p>
<b>Stored Process Service</b>	<a href="#">com.sas.services.storedprocess</a>	<p>access to stored process execution and package navigation</p>	<p>For details about administering stored processes, see the <a href="#">Stored Processes</a> chapter of this guide.</p> <p>For information about developing stored processes,</p>

			see <a href="#">SAS Stored Processes</a> in the <i>SAS Integration Technologies Developer's Guide</i> .
<b>User Service</b>	<a href="#">com.sas.services.user</a>	access to authenticated user context, access to global, solution-wide, and application-specific profiles, and access to personal objects	For details about editing the User Service configuration, see <a href="#">Modifying the Session and User Service Configurations</a> .

In addition, you use the deployment utilities ( [com.sas.services.deployment](#) ) to deploy the services.

- **SAS Management Console plug-ins**, which enable you to administer configuration metadata in a metadata repository. The following plug-ins can be installed with the SAS Foundation Services:

- ◆ **Application Monitor**, which enables administrators to monitor the performance and activities of a foundation service-enabled application.
- ◆ **Foundation Services Manager**, which enables administrators to define and manage service deployments and service configurations.
- ◆ **Publishing Framework Manager**, which enables administrators to set up metadata for users and applications to do the following:
  - ◇ publish SAS files to a variety of destinations
  - ◇ receive and process published information.
- ◆ **Stored Process Manager**, which enables administrators to register and manage metadata for stored processes.

For further information about SAS Foundation Services administration, see the online Help for the appropriate administrative plug-in.

This chapter covers the following SAS Foundation Services topics:

- **Service Deployments**. In order to use the foundation services in your applications, you must deploy the services. To deploy the services, you must configure a service deployment. To understand service deployments and service deployment configuration, see [Understanding Service Deployments](#) and [Understanding Service Deployment Configuration](#).
- **Service Deployment Definitions**. To define and manage service deployments, see the Managing Service Deployments topics.
- **SAS Foundation Service-Enabled Applications**. To understand how applications deploy, locate, and share services, see the following topics:
  - ◆ [Understanding How Applications Deploy Foundation Services](#)
  - ◆ [Understanding How Applications Locate Foundation Services](#) and related scenarios.
  - ◆ [Understanding How Applications Share Foundation Services](#)
- **Service Configurations**. To understand how to modify the configurations of certain foundation services, see [Modifying Service Configurations](#).
- **Application Monitoring**. For information about how to monitor foundation service-enabled applications, see

Monitoring Applications.

*Foundation Services*

# Understanding Service Deployments

A service deployment is a configuration of a collection of SAS Foundation Services that specifies the data necessary to instantiate the services, as well as dependencies upon other services. You create service deployments for applications that will deploy or access the services. You can store the service deployment configuration in either one of the following locations:

- **SAS Metadata Repository:** you can use the Foundation Services Manager plug-in (of SAS Management Console) to administer service deployment metadata that is stored in a SAS Metadata Server repository. The SAS Metadata Server also controls access to the metadata.
- **XML file:** you can export service deployment metadata from the SAS Metadata Server to an XML file. You can then use the XML file to import service deployment metadata into another SAS Metadata Server repository. If you use an XML file to store service deployment metadata, there is no administration or access control for the metadata in the XML file.

**Note:** It is recommended that you store the service deployment metadata on a SAS Metadata Server; storing the service deployment metadata in a SAS Metadata Server enables it to be updated and queried from one centralized location.

To enable your application to deploy and access the foundation services, you can create local or remote service deployments:

- **Local service deployment:** a local service deployment supports exclusive access to a set of services deployed within a single Java Virtual Machine (JVM). Use a local service deployment when you want your application to have its own exclusive set of foundation services.
- **Remote service deployment:** a remote service deployment supports shared access to a set of services that are deployed within a single JVM, but are available to other JVM processes. Use a remote service deployment when you want to share a foundation service deployment among multiple applications. When you create services for remote service deployments, you must specify that the services will be accessed remotely. In order to allow remote access to the services, you must also create a service registry and associate named services with the named components for the remote services.

A service deployment contains:

- **service deployment group(s).** When you create service deployments (local or remote), you can also create groups within the service deployment in order to organize services within a deployment hierarchy.
- **services and service initialization data.** Within each service deployment group, you must define the services for that group. Service definitions contain the following information:
  - ◆ **Service types (interfaces):** service types designate which service interfaces are implemented by the service. The Discovery Service is used to locate services based upon their service interfaces. For example, if you want to locate a service that implements a Logging Service interface, have the Discovery Service search for a service that implements the `com.sas.services.logging.LoggingServiceInterface`.

**Note:** All SAS Foundation Services (including local services) implement the `RemoteServiceInterface`.

- ◆ **Service configuration:** the service configuration specifies the Java class used to create the service, the service's optional configuration data, and the service's configuration user interface. The service configuration user interface defines the Java class used by the Foundation Services Manager to

configure the service's configuration details.

- ◆ **Service dependencies:** when they are deployed, foundation services might depend on the availability of one or more other foundation services. When you define a service, you must specify the other services upon which that service depends. For example, the Authentication Service uses the Logging Service. Therefore, when you define the Authentication Service in a service deployment, you must specify the Logging Service as a dependency.
- ◆ **Service names (for remote access only):** if a foundation service is to be made available for remote clients, you must enable the service for remote access and define named services (service names) that specify the service's name bindings to one or more service registries.
- ◆ **Authorization permissions:** authorization parameters allow you to specify which user or group identities can perform which actions on a particular resource.

**Important Note:** If a service is dependent upon other services, you must define those services before defining the service that depends on them. For details about service dependencies and order of definition, see the [Service Dependencies Table](#).

- **service registries and associated named services (remote service deployments only).** To enable services for remote access, you must define a service registry to use in locating remote services. (A service registry is a searchable registry of service descriptions that is used to register named service bindings).

You must then register the services with the service registry by creating or associating named services that define how each service is to be used within the context of the Discovery Service.

To understand where service deployments are defined, see [Understanding Service Deployment Configuration](#).

## Service Dependencies

If a service has a dependency on another service, you must first create the service upon which it depends. The following table shows the service dependencies and the relative order in which you must define the services.

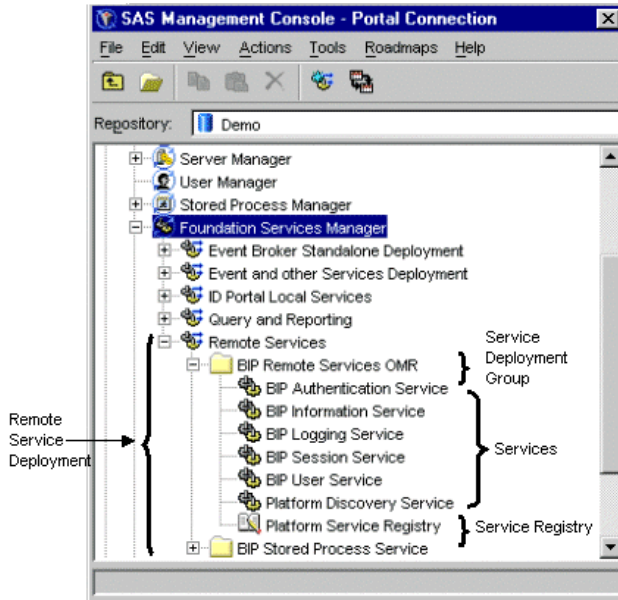
Service Dependencies Table	
Service	Service Dependencies
<b>Logging Service</b>	
<b>Authentication Service</b>	Logging Service
<b>Information Service</b>	Logging Service
<b>User Service</b>	Logging Service Authentication Service Information Service
<b>Session Service</b>	Logging Service Authentication Service Information Service User Service
<b>Discovery Service</b>	Logging Service
<b>Event Broker Discovery Service</b>	Logging Service Discovery Service
<b>Event Broker Service</b>	Logging Service Authentication Service Information Service

	User Service Session Service
<b>Stored Process Service</b>	Logging Service

*Foundation Services*

# Understanding Service Deployment Configuration

The following diagram shows the SAS Management Console Foundation Services Manager connected to a SAS Metadata Repository that contains a service deployment named Remote Services. The diagram also points to the service deployment group, services, and service registry defined within the Remote Services service deployment.



You can define a service deployment in a SAS Metadata Repository in one of the following ways:

- use the Foundation Services Manager Plug-in of SAS Management Console to create a service deployment. For details, see [Defining Service Deployments](#).
- import an XML file containing the service deployment. If your application's service deployment configuration is contained in an XML file, you can import it into a SAS Metadata Repository. For details, see [Importing Service Deployments](#)

After you import or create a service deployment, you can do the following:

- export the service deployment to an XML file. If an application does not have access to a SAS Metadata Repository in its runtime environment, you can export the service deployment configuration to an XML file that the application can access for service deployment configuration information. For details, see [Exporting Service Deployments](#).
- duplicate the service deployment. If you need to use a service deployment that is similar to an existing service deployment, you can duplicate an existing service deployment configuration. For details, see [Duplicating Service Deployments](#)

In addition, you might need to update the prototypes that define the foundation services. (To update prototypes, select the Foundation Services Manager and select **Actions** ► **Update Prototypes**). For further information about using the Foundation Services Manager to create service deployments, see the Foundation Services Manager Help.

## *Foundation Services*



# Defining Service Deployments


You create service deployments for applications to deploy and access SAS Foundation Services. Applications deploy service deployments using the service deployment name configured in a SAS Metadata Repository or XML file.

To create a service deployment, follow these steps:

1. Create a service deployment
2. Create service deployment groups for your service deployment
3. Create services within each service deployment group.
4. For Remote-Accessible services, create a service registry and associated named services.

## Step 1: Create a Service Deployment

To create a service deployment using the Foundation Services Manager, follow these steps:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, click the  next to **Foundation Services Manager** to expand the Foundation Services Manager view.
3. Right-click the **Foundation Services Manger** folder and select **New Service Deployment** from the pop-up menu. The New Service Deployment window appears.
4. Enter a **Name** and, optionally, a **Description** for the service deployment.
5. Click **Finish** to define the service deployment

You may now [define service deployment groups](#) for your service deployment.

## Step 2: Create Service Deployment Groups

After you have defined a service deployment, you can define service deployment groups as follows:

1. In the SAS Management Console navigation tree, select the service deployment in which you want to create a new service deployment group. Right-click the service deployment and select **New Service Deployment Group** from the pop-up menu. The New Service Deployment Group window is displayed.
2. Enter a **Name** and, optionally, a **Description** for the service deployment group.
3. Click **Finish** to create the new service deployment group

After you create the deployment group, you can select the deployment group and do one of the following:

- for local service deployments, create new services within that service deployment group
- for remote service deployments, create the services registry within that service deployment group.

## Step 3: Create a Service

If a service is dependent upon other services, you must define those services before defining the service which depends on them. For details about service dependencies and order of definition, see the [Server Dependencies Table](#).

To create a new service:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, expand the Foundation Services Manager tree to locate and select the service deployment group where you want to create a new service. Right-click the service deployment group and select **New Service** from the pop-up menu. The New Service wizard – Prototype window appears.
3. Select a service to use as a prototype. Click **Next**. The New Service wizard – Names window appears.
4. Enter a **Name** and, optionally, a **Description** for the service. Click **Next**. The New Service wizard – Service Interfaces window appears and displays the associated service interfaces.
5. Click **Next**. The New Service wizard – Service Details window appears and displays the **Service Factory** for the service. If the service has customizable configuration data, you can click **Edit Configuration** to supply the configuration information.
6. Click **Next**. The New Service wizard – Remote Clients window appears.
7. To make this service a remote service, select the **Enable remote clients to access service capabilities** check box and click **Service Names**. The Service Names window appears.
8. Click **New** to define a new named service. The New Named Service wizard – Name window appears.
  - a. Enter the **Name** and optionally, a **Description** for the named service. Click **Next**. The New Named Service wizard – Details window appears.
  - b. Enter the **Name** of the binding, select the **Type** of binding (bind or rebind). If you are creating an Event Broker Service, enter a **Codebase**. If you are creating a new named service for a service registry, click **Select** to select the named component associated with this named service. Click **Next**. The New Named Service wizard – Finish window appears.
  - c. Review the named service definition.
  - d. Click **Finish** to save the named service definition in a metadata repository.

When you are finished creating new named services, click **OK** to return to the New Service wizard – Remote Clients window. Click **Next**. The New Service wizard – Service Dependencies window appears.
9. Select the services that your new service requires. Click **Next**.
10. If you are creating a new Event Broker Service, complete the following steps:
  - a. In the New Service wizard – Defaults window, enter the default event name for the Event Broker Service. Click **Next**.
  - b. In the New Service wizard – Resources window, specify the resources for the Event Broker Service. Click **Next**.
  - c. In the New Service wizard – Connections window, specify the administrator port and transport monitors for the Event Broker Service. Click **Next**.
11. In the New Service wizard – Finish window, review the service definition.
12. Click **Finish** to save the service definition in a metadata repository.

You can create additional services for your service deployment. If the service is enabled for remote access, you must create a new service registry and associate named services.

## Step 4: Create a Service Registry and Named Services

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, expand the Foundation Services Manager tree to locate and select the service deployment group where you want to create a new service registry. Right-click the service deployment group and select **New Service Registry** from the pop-up menu. The New Service Registry wizard – Type window appears.
3. Select the type of service registry you wish to define. Click **Next**. The New Service Registry wizard – Name window appears.
4. Enter a **Name** and **Description** (optional) for the service registry. Click **Next**. The New Service Registry wizard – Service Interfaces window appears and displays the service interfaces satisfied by this definition of a

service registry. Click **Next**. The New Service Registry wizard – Host window appears.

5. Specify the **Host Name** and **Port Number** to use to bind to the service registry. The only currently supported application protocol is RMI. Click **Next**. The New Service Registry wizard – Named Services window appears.
  6. If you have not already defined the appropriate remote accessible service, Click **New** to define a new named service. The New Named Service wizard appears.
    - a. Enter the **Name** and optionally, a **Description** for the named service. Click **Next**. The New Named Service wizard – Details window appears.
    - b. Enter the **Name** of the binding, select the **Type** of binding (bind or rebind). If you are creating an Event Broker Service, enter a **Codebase**. If you are creating a new named service for a service registry, click **Select** to select the named component associated with this named service. Click **Next**. The New Named Service wizard – Finish window appears.
    - c. Review the named service definition.
    - d. Click **Finish** to save the named service definition in a metadata repository.
- When you are finished creating new named services, click **OK**. Click **Next**. The New Service Registry wizard – Finish window appears.
7. Review the service registry definition.
  8. Click **Finish** to define the service registry in a metadata repository.

After you create the service registry, you can select the service deployment group for the registry and create the services, including the named services associated with the service registry.

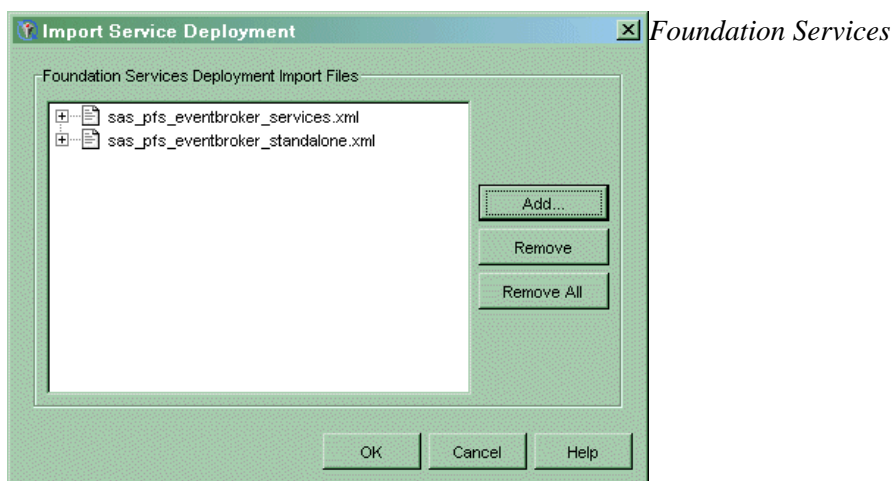
### *Foundation Services*

# Importing Service Deployments

The Foundation Services Manager enables you to import an XML file that contains the metadata necessary to create a service deployment in the Foundation Services Manager. To import a service deployment:


1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, right-click **Foundation Services Manager** and select **Import Service Development** from the pop-up menu. The Import Service Development window appears.
3. The **Foundation Services Deployment Import Files** field lists the files you have selected to import. To select a new file, click **Add**. To remove the file from the import list, click **Remove** or **Remove All**.
4. Click **OK** to import the files and close the window.

The following SAS Management Console screen shot shows the Import Service Deployment window:

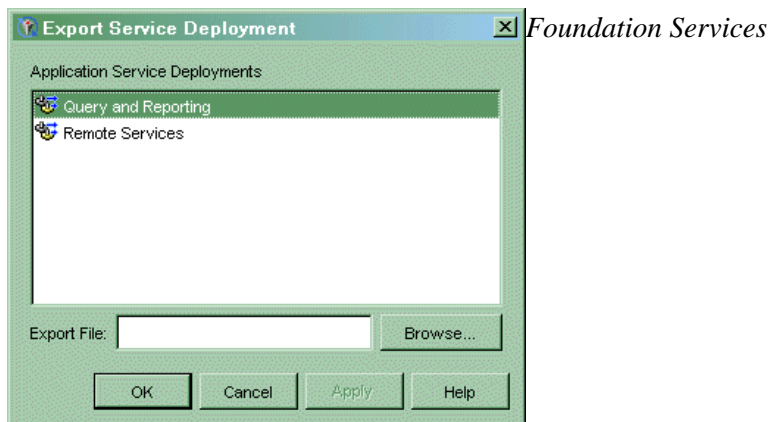


# Exporting Service Deployments

You can export the metadata for a service deployment and its contained objects to an XML file. To export a deployment:


1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, click the  next to **Foundation Services Manager** to expand the Foundation Services Manager view.
3. Right-click a service deployment in the navigation tree and select **Export Service Deployment** from the pop-up menu. The Export Service Deployment window appears.
4. In the Export Service Deployment window, select the deployments whose data you want to export from the list in the **Application Service Deployments** field.
5. In the **Export File** field, type the name of the file (including the **.xml** extension) to which you want to export the data. You can also click **Browse** to interactively select a file. **Note:** You must specify the **.xml** file extension with the file name.
6. Click **OK** to export the service deployment to a file and close the window.

The following SAS Management Console screen shot shows the Export Service Deployment window:

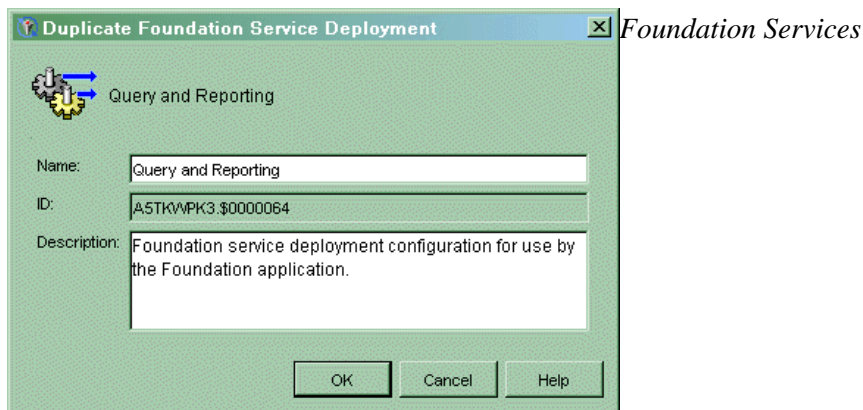


# Duplicating Service Deployments

The Foundation Services Manager plug-in of the SAS Management Console enables you to duplicate an existing service deployment under a new name. To duplicate a service deployment:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, click the  next to **Foundation Services Manager** to expand the Foundation Services Manager view.
3. Right-click a service deployment in the navigation tree and select **Duplicate Service Deployment** from the pop-up menu. The Duplicate Service Development window appears.
4. The **Name** field contains the name of the service deployment you are duplicating. You must change this name to a unique deployment name. Enter a new **Description** if needed.
5. Click **OK** to duplicate the service deployment. The new deployment appears in the navigation tree under the Foundation Services Manager.

The following SAS Management Console screen shot shows the Duplicate Foundation Services Deployment window:



# Redistributing Service Deployments

After you have configured a remote service deployment, you might need to move your remote service deployment or service registry to a different machine.

**Note:** Before you can redistribute service deployments or service registries, if the service deployment exists in an XML file instead of on the SAS Metadata Server, you must first import the service deployment into a SAS Metadata Repository.

You can use SAS Management Console to reconfigure parameters as follows:

- move the remote service deployment to another machine. To move a remote service deployment to another machine, you must use the Foundation Services Manager to reconfigure any machine-specific service configuration data. For example, the logging service might be configured to send its output to a file in the directory `c:\original\log.txt` on a Windows machine. If you move the remote service deployment to a UNIX machine, you must edit the logging service configuration and change the log file directory to `/newmachine/log.txt`

**Note:** If the application that deploys the remote services is starting the service registry, the service registry must be located on the same machine as the remote services deployment.

- move the service registry to another machine. To move a service registry to another machine, follow these steps:
  - ◆ reconfigure the Service Registry definition in the service deployment. To reconfigure the service registry, use the Foundation Services Manager to update the service registry's host name and port number.

**Note:** If the service registry's host name is configured as `localhost`, you do not need to update the configuration when you move the service registry to a different machine.

**Note:** You must ensure that the port configured for the service registry does not conflict with a port that is already in use on the new machine.

- ◆ for Event Broker Service definitions only, reconfigure any codebase property changes in the Named Services definition. To reconfigure the codebase properties, use the Foundation Services Manager to update the named service definitions on the service registry or in the service definition.
- ◆ ensure that the application that starts the service registry is coded to call the correct host name. For details, see the SAS Foundation Services class documentation for the Deployment Service.

After you have finished using SAS Management Console to re-configure the service deployment, if the service deployment was imported into the SAS Management Console from an XML file, use SAS Management Console to export the service deployment back to an XML file. You must export or copy the file to the location where the application accesses the XML file.

*Foundation Services*

# Understanding How Applications Deploy Foundation Services

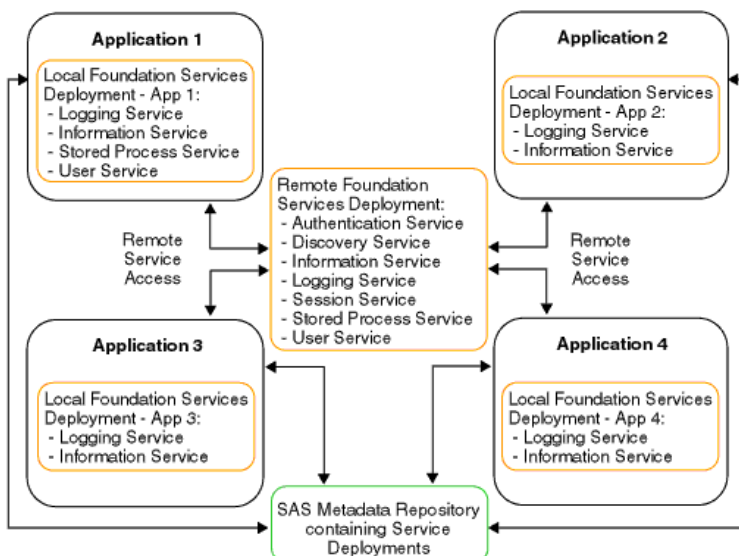
Applications can access service deployments from a SAS Metadata Repository or an XML file (that contains exported metadata). Applications deploy services as follows:

- **For a local service deployment**, the application uses a service loader utility to instantiate and initialize the SAS Foundation Services for a local service deployment, and register the deployed services with a local Discovery Service. The application then has exclusive access to these locally deployed services. For a stand-alone deployment, you do not need to configure a Discovery Service.
- **For a remote service deployment** that is shared between applications, one of the applications must deploy the remote service deployment. The application uses a service loader utility to instantiate and initialize the foundation services for a remote service deployment, and register the deployed services with a local Discovery Service. The application then has local access to the services. To enable the services for remote access, the remote service deployment specifies a remote Discovery Service which registers with the service registry. The remote service deployment also contains a distributable configuration for any service that remote clients will access. These remote services are registered with a remote Discovery Service. Other applications can then use the remote Discovery Service to access the remote services.

**Note:** A foundation service-enabled application can be either a standard client application or a Web client application that runs in a servlet container.

Your application must install the appropriate JAR files (for example, sas.svc.core.jar) in a location that is only accessible to its own classloader. This installation restriction is due to the inheritance hierarchy of classloaders. This inheritance hierarchy enables multiple applications to access classes that are available to higher level class loaders. Therefore, each foundation service-enabled application should NOT install the required JAR files in a location that is accessible to a class loader that might be shared amongst multiple applications. For details about coding client applications for service deployment, see the SAS Foundation Services class documentation for [com.sas.services.deployment](#) and [com.sas.services.discovery](#) and the [SAS Integration Technologies Developer's Guide](#).

The following diagram shows these components and how they work together.





## SAS® 9.1.2 Integration Technologies: Administrator's Guide

In the diagram, Applications 1 through 4 all access their local and remote service deployment configurations from a SAS Metadata Repository.

If Application 1 deploys the remote service deployment, the services are registered with a local Discovery Service and a remote Discovery Service. Applications 2, 3, and 4 can then use the remote Discovery Service to locate and access the deployed remote services. All of the applications share the same remote service deployment. In addition, each application has exclusive access to its own local service deployment. For information about how applications locate and access services, see [Understanding How Applications Locate Services](#).

The different components in the diagram might exist on the same Web server, or on different Web servers. You can install your applications and deploy your services on separate machines as required by the needs of your implementation. For information about distributing service deployments, see [Redistributing Service Deployments](#).

*Foundation Services*

# Understanding How Applications Locate Foundation Services

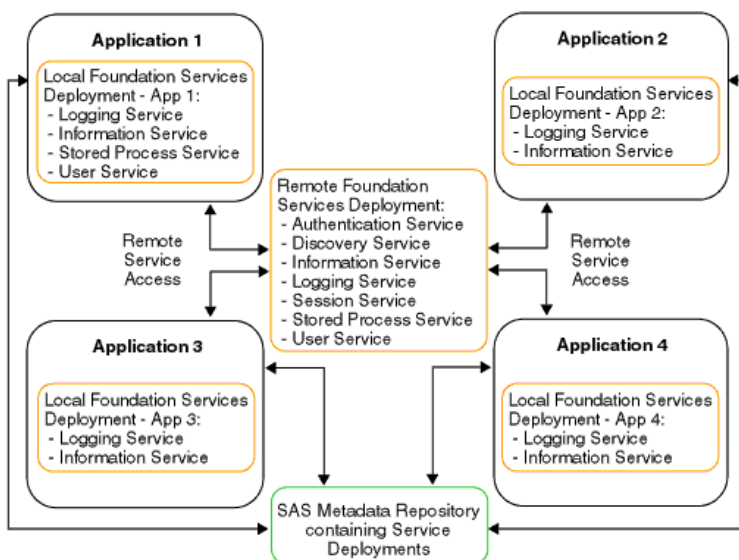
Applications can access services that are deployed locally or remotely.

**Note:** Your foundation service-enabled application can be either a standard client application or a Web client application that runs in a servlet container.

To locate local and remote services:

1. The application uses a service loader to instantiate and initialize local services, including its local Discovery Service.
2. The application initializes and registers the local Discovery Service with a remote Discovery Service. The application locates the remote Discovery Service by obtaining the Remote Method Invocation (RMI) registry location from a SAS Metadata Repository (or XML file that contains exported metadata) and performing an RMI name lookup on the remote Discovery Service. The remote Discovery Service enables the client to locate remotely deployed SAS Foundation Services.
3. When the application requests a service, its local Discovery Service first checks to see if the service is a locally registered service.
  - ◆ If the requested service is a locally registered service, the application binds to the local service.
  - ◆ If the requested service is not a locally registered service, then the local Discovery Service uses the remote Discovery Service to search the remote services deployment for the requested service.
    - ◇ If the requested service is not registered with the remote Discovery Service, an error is returned.
    - ◇ If the requested service is registered with the remote Discovery Service, a stub to the remote service is returned and the application can then use the remote service.

For example, in the following diagram, if an application requests the Logging Service, the application will bind to the local Logging Service. If an application requests the Session Service, the application will use the remote Discovery Service to locate and bind to the remote Session Service.



**Note:** If the application that deploys the remote services also starts the service registry, the service registry must exist on the same machine as that application.

The following scenarios show examples of local and remote service deployment and access.

- Scenario: Standalone Application
- Scenario: Remote-Accessible Services
- Scenario: Local and Remote-Accessible Services

*Foundation Services*

# Scenario: Stand-alone Application

A stand-alone application deploys services locally, uses the services, and terminates the services when they are no longer needed. If an application does not need to interact with any other applications, then it can be a stand-alone application with its own exclusive local service deployment. Services locally deployed by this application are not available to any other application; in addition, no remote services are available.

**Note:** A foundation service-enabled application can be either a standard client application or a Web client application that runs in a servlet container.

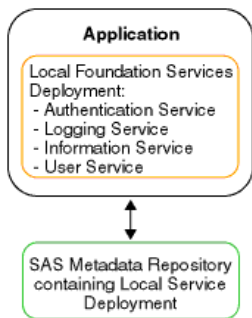
To deploy local services for its own exclusive use, the application:

1. Uses the service loader to query service deployment metadata from either a SAS Metadata Server or XML file (that contains exported metadata).
2. Uses the service loader to instantiate services defined in the service deployment metadata and registers them with the local Discovery Service.
3. Uses the local Discovery Service to find services based upon their service interfaces and optionally, their service attributes.

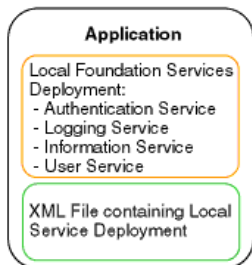
When the application no longer needs the services or is ready to exit, it terminates the local Discovery Service; the local Discovery Service then destroys all locally instantiated services.

Figures 1 and 2 show standalone applications that access their service deployments from a SAS Metadata Repository or XML file respectively. Figure 3 shows two standalone Web applications that access their service deployments from a SAS Metadata Repository and each deploy their own local services for their own exclusive use.

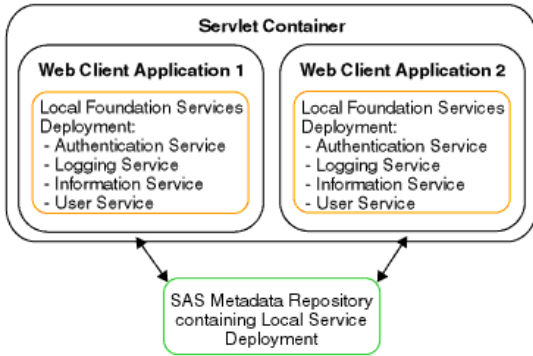
**Figure 1: Standalone Application accessing Local Deployment from a SAS Metadata Server**



**Figure 2: Standalone Application accessing Local Deployment from an XML File**



**Figure 3: Two Standalone Web Applications accessing Local Deployments from a SAS Metadata Repository**

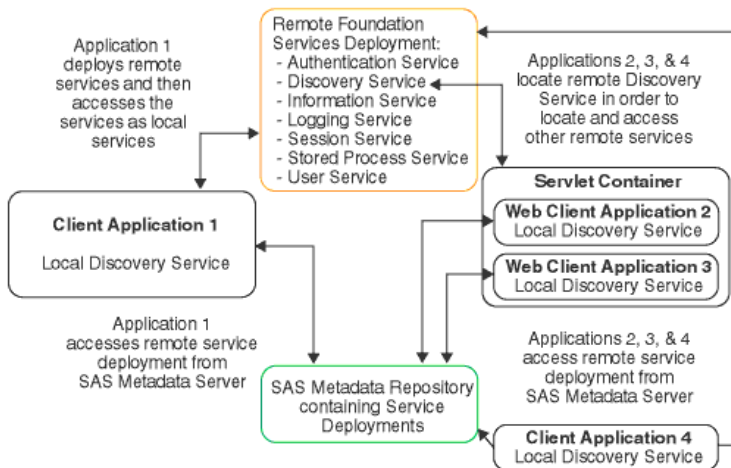


*Foundation Services*

# Scenario: Remote-accessible Services

To enable applications to access remote services, one application must deploy the remote services. (The application that deploys the remote services can then access the services as local services). Instead of deploying their own set of local services, other applications can access the remote services. To access the remote service deployment, applications locate the deploying application's remote Discovery Service in order to locate and access the deployed remote services. This scenario is useful if one or more client applications need to use the same set of services.

In this scenario, Application 1 deploys the remote services and accesses them as local services. Applications 2, 3, and 4 locate Application 1's remote Discovery Service in order to access the remote services. Note that Applications 2 and 3 are Web client applications that run in the same servlet container and each deploy their own local services for their own exclusive use.



## To deploy remote services, Application 1 does the following:

1. Uses the service loader to query service deployment metadata from either a SAS Metadata Server or an XML file (that contains exported metadata).
2. Uses the service loader to instantiate services defined in the service deployment metadata and register them with the local Discovery Service.

**Note:** In this scenario, these services must be configured as remote-accessible.

3. Uses its local Discovery Service to find services based upon their service interfaces and optionally, their service attributes.

## To locate the remote-accessible services (that were deployed by Application 1), Applications 2, 3, and 4 do the following:

1. Use the service loader to query service deployment metadata from either a SAS Metadata Server or an XML file (that contains exported metadata).
2. Use the service loader to obtain a name binding to the remote-accessible Discovery Service instantiated by Application 1.
3. Register the remote Discovery Service with their own local Discovery Service.
4. Use their own local Discovery Service to find services based upon their service interfaces and optionally, their service attributes. The local Discovery Service uses the remote Discovery Service to locate the remote-accessible services.

## SAS® 9.1.2 Integration Technologies: Administrator's Guide

**Note:** In this scenario, Applications 2, 3 and 4 do not deploy any services themselves; they only locate remote-accessible services instantiated by Application 1.

5. When Applications 2, 3, and 4 no longer need the services, they each terminate their own local Discovery Service.

When Application 1 exits, it terminates its local Discovery Service; the local Discovery Service then terminates all locally instantiated services. After all services are terminated, no services are available to any other applications.

### *Foundation Services*

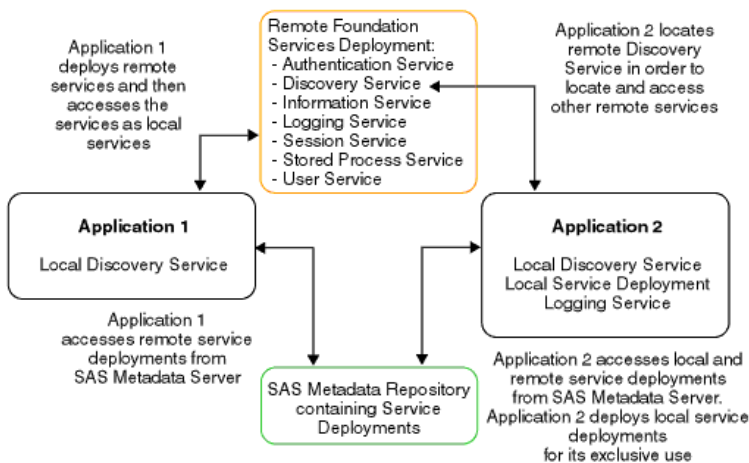
# Scenario: Local and Remote-accessible Services

To enable other applications to access remote services, one application must deploy the remote services. (The application that deploys the remote services can then access the services as local services). Instead of deploying their own set of local services, other applications can access the remote service deployment. To access the remote service deployment, applications locate the deploying application's remote Discovery Service in order to locate and access the deployed remote services. In addition, these applications can each have their own set of locally deployed services to which each application has its own exclusive access. This example is useful when client applications need to have both of the following:

- services deployed locally for exclusive use
- use of the same set of remote services

**Note:** A foundation service-enabled application can be either a standard client application or a Web client application that runs in a servlet container.

In this scenario, Application 1 deploys the remote services and accesses them as local services. Application 2 locates Application 1's remote Discovery Service in order to access the remote services. Application 2 also deploys local services for its own exclusive use.



**To deploy remote services and access these services locally, Application 1 does the following:**

1. Uses the service loader to query service deployment metadata from either a SAS Metadata Server or an XML file (that contains exported metadata).
2. Uses the service loader to instantiate services defined in the metadata and register them with the local Discovery Service.

**Note:** These services must be configured for remote access.

3. Uses its local Discovery Service to find services based upon their service interfaces and optionally, service attributes.

**To deploy local services and access remote services, Application 2 does the following:**

1. Uses the service loader to query service deployment metadata from either a SAS Metadata Server or an XML file (that contains exported metadata).



## SAS® 9.1.2 Integration Technologies: Administrator's Guide

2. Uses the service loader to instantiate services defined in the metadata and register them with the local Discovery Service.

**Note:** Because these services are only used by Application 2, they are not configured for remote access.

3. Uses the service loader to query service deployment metadata from either a SAS Metadata Server or an XML file (that contains exported metadata).
4. Uses the service loader to obtain a binding to the remote Discovery Service instantiated by Application 1.
5. Uses its local Discovery Service to find services based upon their service interfaces and optionally, service attributes.

**Note:** Application 2 has access to both local services and remote services. When services are located, the local Discovery Service first tries to find a service locally before it looks for a remote-accessible service.

6. When Application 2 no longer needs the services it terminates its local Discovery Service. This will cause its locally instantiated services to be destroyed and its bindings to Application 1's remote services to be terminated.

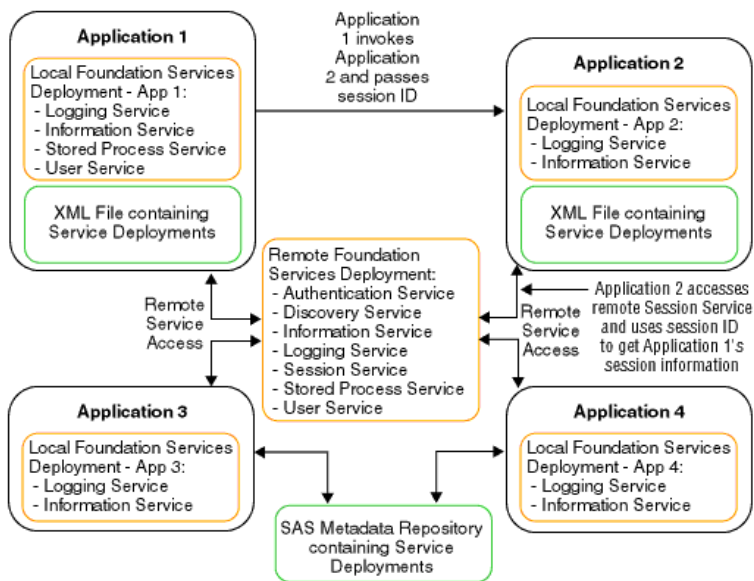
When Application 1 exits, it terminates the local Discovery Service; the local Discovery Service then terminates all locally instantiated services. After all services are terminated, no services are available to any applications.

### *Foundation Services*

# Understanding How Applications Share Foundation Services

An application can use the SAS Foundation Services to access another application's session context.

**Note:** A foundation service–enabled application can be either a standard client application or a Web client application that runs in a servlet container.



In the diagram, Applications 1–4 use the same remotely deployed Session Service. When Application 1 launches Application 2, it passes its session ID to Application 2. Application 2 can then bind to the remote Session Service and obtain and use Application 1's session and user context information. This allows the user to seamlessly pass-through to Application 2 without requiring a separate login definition.

## Foundation Services

# Modifying Service Configurations

After you define a service deployment and its associated services, you might want to edit the configuration information for particular services. You can use the Foundation Services Manager to modify the configuration data for the following services:

- Event Broker Service. For details, see [Understanding Events and Process Flows](#) and [Modifying the Event Broker Service Configuration](#).
- Information Service. For details, see [Modifying the Information Service Configuration](#).
- Logging Service. For details, see [Modifying the Logging Service Configuration](#).
- Session Service. For details, see [Modifying the Session and User Service Configurations](#).
- User Service. For details, see [Modifying the Session and User Service Configurations](#).

*Foundation Services*

# Understanding the Event Broker Service

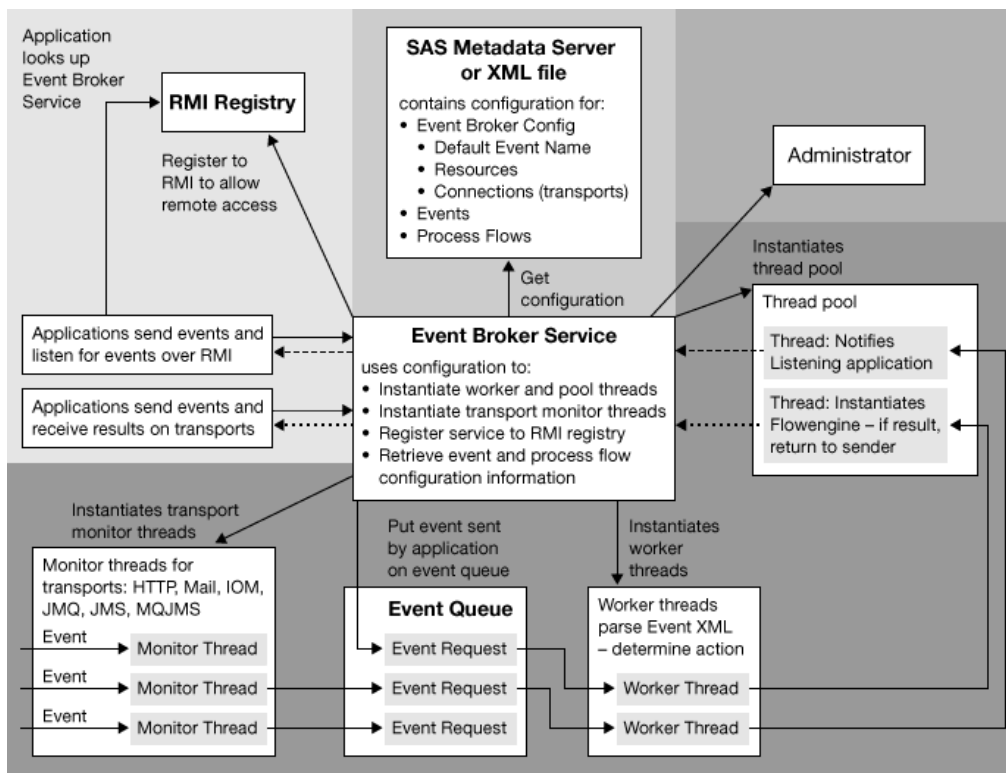
The Event Broker Service enables you to receive external event notifications and process them based on the name of the event that is received. Events can be structured or unstructured as follows:

- A structured event is specified as well-formed XML and adheres to the event message specification. (For details about the event message specification, see the SAS Foundation Services class documentation for the Event Broker Service.) It contains information such as the name of the event, the associated properties, and the message body.
- An unstructured event must also be specified as well-formed XML; however, it does not adhere to the event message specification. For unstructured events, the entire event is parsed as the message body.

**Note:** The Event Broker Service only handles unstructured events if default event handlers have been configured.

For details about the event message specification, see [com.sas.services.events.broker](#) in the Foundation Services Class Documentation.

The following diagram shows the components of the Event Broker Service:



The Event Broker Service works as follows:

1. **Listens for incoming events via transports or applications.** The Event Broker Service can monitor for and receive events via the following transports:
  - ◆ **RMI:** if the RMI transport is enabled, the Event Broker Service registers itself to one or more RMI registries.

To enable the event broker service to be accessed via RMI, you must enable remote access in the service configuration. Enabling remote access registers the service to the RMI Registry. Remote access to the Event Broker Service enables the following:

- ◊ Java clients that are *sending* can use the appropriate RMI (remote method invocation) registry to locate the Event Broker Service in order to send events
  - ◊ Java clients that are *listening* can use the appropriate RMI (remote method invocation) registry to locate the Event Broker Service and register to listen for particular events.
  - ◆ **HTTP:** the HTTP transport listens for events sent from HTTP clients. Clients can also be SOAP-enabled.
  - ◆ **JMS:** the JMS transport listens for events sent from any JMS-compliant messaging client. This transport uses administered objects to isolate client applications from the proprietary aspects of a provider. When you configure this transport, you specify whether the administered objects are on the local file system or an LDAP directory server. The transport then uses JNDI to look up the administered objects on the local files system or LDAP directory server.
  - ◆ **MQJMS:** the MQJMS transport listens for events sent from WebsphereMQ (formerly MQSeries) messaging clients.
  - ◆ **JMQ:** the JMQ transport listens for events sent from SunONEMQ (formerly iPlanet Message Queue) messaging clients.
  - ◆ **Mail:** the mail transport listens for events sent to IMAP or POP3 mail servers.
  - ◆ **IOM:** the IOM transport listens for events sent from SAS servers.
2. **Determines the event name to use for event configuration information.** The Event Broker Service parses the event XML to determine the event name (or names if a naming hierarchy is used) to use for event configuration information. If an unstructured event is received, the Event Broker Service uses the service configuration information to map the unstructured event to a default event name.
  3. **Forwards the event to the appropriate event handling agent(s) based on the configured event type.** The Event Broker Service uses configuration information defined for the event name or default event names to determine appropriate actions to take for the event.

For a broadcast event type, the Event Broker Service notifies all handling agents (process flows and listening applications) of the event as follows:

- ◆ If an application is a registered listener for an event, the Event Broker Service notifies the listening application of the event.
- ◆ If the event configuration contains process flows, the Event Broker Service instantiates a flow engine for each configured process flow in order to process the event message.

For a request/response event type, the Event Broker Service notifies only one handling agent (listening application or process flow) as follows:

- ◆ If an application is a registered listener for an event, the registered listener has precedence over a process flow (only one process flow can be defined for request/response types). Therefore, the Event Broker Service forwards the event to the listening application.
  - ◆ If the event configuration contains a process flow and there is no application that is a registered listener, the Event Broker Service instantiates a flow engine to process the event message.
4. **Sends a response based on the event response type.** The Event Broker Service uses the event configuration to determine whether to send a response to an event.
    - ◆ If the event sender does not require a reply, the event request should specify a response type of *none* or *ack* (acknowledge). To configure an event for no response or acknowledge, you specify *broadcast* as the event type. For acknowledge response types, the Event Broker Service sends an acknowledge receipt to the event sender.
    - ◆ If the event sender requires a reply, the event request should specify a response type of *result*. To configure an event for a response, you specify *request/response* as the event type. For request/response types, the Event Broker Service sends a response to the event sender.

**Important Note:** Unstructured event requests are automatically assigned a response type of *none*. Therefore, for event definitions that will be used to handle unstructured event requests, you must configure the response type as *broadcast*.

**Important Note:** An event is completely qualified by its name and type. Therefore, the Event Broker Service will view events as separate events if they are sent or configured with the same name, but different event types. For example, if you send an event named `AlertHigh` with a response type of *none* to an event broker that contains an event definition named `AlertHigh` that is configured as a *request/response* type of event, an error is returned.

Applications can send and receive events using either of the following:

- transport monitors
- RMI (remote method invocation).

## Overview of Event Broker Discovery Service

The Event Broker Discovery Service provides the ability to locate one or more Event Broker Services that can process a particular event.

By default, an Event Broker Discovery Service can locate any Event Broker Service that is part of its same deployment. However, to locate an Event Broker Service outside of its deployment, the Event Broker Service must be remote-accessible and its location must be defined to the Event Broker Discovery Service. You can define Event Broker Service location information as part of the Event Broker Discovery Service configuration. Format the configuration with XML initialization data as follows:

```
<EventBrokers>
  <Location url="//host:port/name" />
  <Location url="//host:port/name" />
    ...
</EventBrokers>
```

Specify the appropriate RMI URL specification for each remote-accessible Event Broker Service. *Foundation Services*

# Understanding Events and Process Flows

The Event Broker Service configuration allows you to configure one or more events. When an event is received, the Event Broker Service maps the event name to a configured event name. If an unstructured event is received, the Event Broker Service maps the unstructured event to a configured default event name.

An Event configuration consists of the following information:

- **Name:** the name of the event in the incoming XML request maps to the configured event name. You can also name events so that they are part of a naming hierarchy. Events in a naming hierarchy are separated by a period. For example: `Animals`, `Animals.Dogs`, `Animals.Dogs.Retriever`. Naming hierarchies are handled differently based on the event type:
  - ◆ If a broadcast event for `Animals.Dogs.Lab` is received, the event is delivered to all handling agents (process flow or application) that are registered for `Animals.Dogs.Lab`, `Animals.Dogs`, and `Animals`.
  - ◆ If a request/response event is received, it is delivered to a single handling agent. If the incoming request contains an event name that does not exactly match an event name in the Event Broker Service configuration, the naming hierarchy is searched for the best possible event name match that is also configured as a request/response event type.
- **Type:** events can be one of the following types:
  - ◆ *Broadcast*, where a notification is sent to all handling agents, and either no response or an acknowledge receipt, is sent to the originating client.
  - ◆ *Request/Response*, where notification is sent to one handling agent and a response is sent to the originating client.

Configure the event type as follows:

- ◆ If the incoming XML request specifies a response type of *none* or *ack* (acknowledge), the event sender does not require a reply. To configure an event for no response or acknowledge, you specify *Broadcast* as the event type. For unstructured events, specify *Broadcast* as the event type.
- ◆ If the incoming XML request specifies a response type of *result*, the event sender requires a reply. To configure an event for a response, specify *Request/Response* as the event type.

The following table summarizes information about the incoming event request/response type and configured event type.

Event Request/Response Type	Configured Event Response Type	Event Notifications	Event Response
none	Broadcast	Notification sent to all process flows configured for the event and all listening applications registered for the event.	No response sent
ack	Broadcast	Notification sent to all process flows configured for the event and all listening applications registered for the event.	Acknowledge receipt sent to the event sender.
result	Request/Response	Notification sent to only one handling agent (listening application or process flow). If there is a listening application, it takes	Response sent to the event sender.

		precedence over the process flow.	
--	--	-----------------------------------	--

**Note:** If the event configuration does not match the incoming event request response type, then an error is returned (`Event not configured`).

- **Security:** you can specify different security attributes for each event:
  - ◆ To authenticate and authorize the sender's credential, select the **Check sender's authorization**. If you select the **Check sender's authorization** property, the event's process flows will not run unless the sender's credentials are successfully authenticated by the SAS Metadata Server's authentication provider and then authorized by the SAS Metadata Server's authorization facility as having the *Execute* permission for the event.

**Note:** The sender's event request must contain the sender's user ID and password, and optionally, the authentication domain. You can configure a default authentication domain in the configuration for the User Service (see [Additional Security Configuration](#)); if you configure a default authentication domain in the User Service, then the sender is not required to specify the authentication domain in the event request.

- ◆ To run event process flows under a particular identity, you must configure the events to run under one of the following:
  - ◇ the sender's identity
  - ◇ the broker's identity
    - Note:** You can only configure event process flows to run under the broker's identity if the Event Broker Service is deployed using a SAS Metadata Server (instead of an XML file) as the metadata source.
  - ◇ an identity that you supply in the configuration
 You can also specify that the event run with no security.

## Additional Security Configuration

To set up security for sender's credentials or event process flows, you must

- use the User Manager plug-in to SAS Management Console to define user or group identities in the SAS Metadata Repository.
- create, configure, and deploy the User Service (of the SAS Foundation Services). You must configure and deploy the User Service as part of the Event Broker Service's service deployment; the User Service must be available to the Event Broker Service at run-time.

To authenticate users, the User Service requires an appropriate login module configuration file. In addition, other Java 2 policy and JAAS policy files might be required. For example, to run an event's process flows under a particular security context, you must set up subject-based security with the JAAS policy configuration file in order to restrict access to the appropriate resources.

For details about required User Service configuration, see the SAS Foundation Services class documentation for the User and Security Services. For details about additional User Service configuration in the Foundation Services Manager, see [Modifying the Session and User Service Configurations](#).

In addition, to set up authorization for sender credentials, you must grant the sender the *Execute* permission for the event. To grant the *Execute* permission:

1. Use the Authorization Manager plug-in to SAS Management Console to define the *Execute* permission.



2. From the Foundation Services Manager, open the event properties.
3. On the event's Authorization tab, click **Add** to add the appropriate user or group for the sender.
4. Also on the event's Authorization tab, select the sender's user or group identity and grant the *Execute* permission.

After you define an event, you can define your process flows.

## Understanding Process Flows

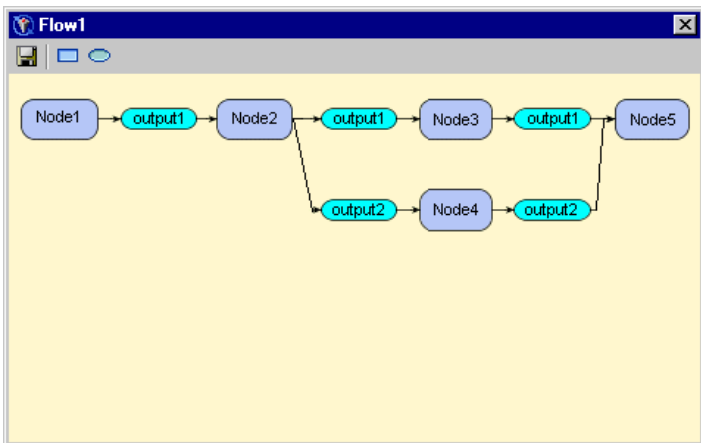
Process flows are used to process event messages. Process flows contain process nodes, which contain logic to process messages, and message nodes, which encapsulate the inputs and outputs for the process nodes.

- **For broadcast events**, you can configure one or more process flows for an event.
- **For request/response events**, you can only configure one process flow for an event.

You can configure a process flow by using the Process Flow Editor to define a Process Flow Diagram (PFD). A process flow configuration consists of:

- **Name and description:** the process name and optionally, a description.
- **Process nodes:** a process node is a Java class that can have one or more inputs and outputs. You diagram these inputs and outputs as message nodes. When an event is received and a process flow needs to be instantiated for the event, a runtime flow engine is instantiated. The runtime flow engine calls a process node by instantiating the Java class associated with that node. Currently, all process nodes are executed synchronously. A process node configuration consists of:
  - ◆ **Name and description:** the process node name and optionally, a description.
  - ◆ **A class:** the Java class is used to instantiate the process node. You can then generate the skeleton for the class, define your logic for the class, and compile the class.
  - ◆ **Attributes for the class:** attributes are name/value pairs for the class.
- Note:** If a process node has no predecessors, it is the starting node for the process flow. Each process flow can have only one starting node.
- **Message nodes:** a message node encapsulates the outputs and inputs to process nodes in a process flow. A message node configuration consists of:
  - ◆ **Name and description:** the message node name and optionally, a description.
  - ◆ **Details:** details specify whether a message is required from the previous process node in order to make a process node eligible for firing.

The following screen capture shows an example of a portion of a process flow diagram:



*Foundation Services*

# Modifying an Event Broker Service Configuration

After you create an Event Broker Service in your service deployment, you can modify its service configuration.

To modify the Event Broker Service configuration:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, expand the folders in the Foundation Services Manager until you find the Event Broker Service you want to modify.
3. Right-click the service you want to modify and select **Properties** from the pop-up menu. The object's properties are displayed.
4. Select the Service Configuration tab and click **Edit Configuration**. The EventBroker Service Configuration window appears.
5. On the Defaults tab, enter the **Default event name** to use for unstructured events. Select the Resources tab.
6. On the Resources tab, enter the event management and thread pool information for the service. Select the Connections tab.
7. On the Connections tab, specify an **Administrator Port** and click **Insert** to create a new transport or select a transport and click **Edit** to edit a transport's properties. For details about creating and editing transports, see the Foundation Services Manager help. When you are finished editing a transport, click **OK**.
8. To enable a service for remote access:
  - a. On the Connections tab, select the RMI\_Transport and click **Edit** to edit the RMI transport's properties. Select the General tab.
  - b. On the General tab, to configure a new default event name for RMI transports that overrides the default Event Broker Service event name, enter a **Default event name**. Select the RMI Details tab.
  - c. On the RMI Details tab, select the **Enable remote clients to access service capabilities** check box and click **Service Names** to define a new named service. When you are finished creating named services and editing the transport, click **OK**.
9. When you are finished creating or editing a transport, click **OK** to save the Event Broker Service configuration to the metadata repository.

After you edit the Event Broker Service configuration, you can select the Event Broker Service in the navigation tree and create event definitions for the Event Broker Service. The event definitions you create can then be used to hold process flow definitions that you create.

*Foundation Services*

# Creating Events and Process Flows

## Create a New Event



To create a new event:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, expand the folders in the Foundation Services Manager until you find the Event Broker Service for which you want to define a new event.
3. Right-click on the Event Broker Service and select **New Event** from the pop-up menu. The New Event wizard window appears.
4. Enter a **Name** and optionally, a **Description**. Click **Next**. The New Event wizard – Type window appears.
5. Select the type of event that you want to create. Click **Next**. The New Event wizard – Security window appears.
6. Select the type of security to use when running the event. Click **Next**.
7. Review the event definition and Click **Finish** to save the event definition in the metadata repository.


After you define an event, you can select the event definition in the navigation tree and create process definitions for the event.

## Create a New Process and Process Flow

To create a new process and process flow:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, expand the folders in the Foundation Services Manager until you find the event for which you want to define a new process flow.
3. Right-click on the event and select **New Process** from the pop-up menu. The New Process Wizard – Name window appears.
4. Enter a **Name** to use for the process flow. Click **Finish** to save the process flow definition in the metadata repository.
5. In the navigation tree, right-click the process flow that you just defined and select **Process Editor** from the pop-up menu. The Process Editor appears.
6. In the toolbar, select , hold down the mouse button, and drag the cursor into the drawing area of the Process Editor. The New Process Node Wizard appears.
7. Enter a **Name** and optionally, a **Description** for the process node. Click **Next**. The Process Node Wizard – Class Window appears.
8. Enter the **Class** to instantiate for this process node, and **Generate** and **Compile** the class as appropriate. Click **Next**. The Process Node Wizard – Attributes Window appears.
9. Click **Insert** to add a new row to the name/value columns. Select the added row and double-click the **Name** or **Value** field with the left mouse button in order to edit the field. When you have finished entering your name/value pairs, click **Finish**. The Process Node definition is saved. You can now create a message node for outputs and inputs.
10. In the toolbar, select , hold down the mouse button, and drag the cursor into the drawing area of the Process Editor. The Process Message Wizard appears.
11. Enter a **Name** and, optionally, a **Description** for the message node. Click **Next**. The Process Message Wizard – Format Window appears.
12. Select the **Usage** drop-down and choose whether the input is required or optional for downstream process nodes. Optionally, specify the **Format** for the node. Click **Finish** to define the message node.

## SAS® 9.1.2 Integration Technologies: Administrator's Guide

13. To create a connection between the process node and the message node, position your cursor on the process or message node so that a pencil icon appears. Click the left mouse button and drag the cursor to the node to which you are making the connection.
14. Create other process nodes, message nodes, and connections as required for the process flow.
15. Click  to save the process flow.

### *Foundation Services*

# Modifying the Information Service Configuration

The Information Service:

- provides a mechanism to perform a federated search of any repositories that a user has a connection to. The term federated means connected and treated as one. The classes in the Information Service package enable the creation of a single filter which can search disparate repositories (for example, SAS Metadata Repositories and LDAP repositories).
- allows repository-specific searches to be performed, so that efficient searching can be achieved.
- provides a convenience method for fetching an item from a repository using a URL.
- can be used in conjunction with the User Services and the Authentication Service to authenticate users, create User Contexts, locate servers that the user has access to, and create repository definitions to use in making server connections.

For more information about the Information Service, see [com.sas.services.information](#) in the Foundation Services class documentation.

The Information Service configuration consists of the following items:

- **Protocols:** the protocol definition maps the repository protocol to a Java class that implements the `com.sas.services.information.RepositoryInterface` interface. When connecting to a repository, the protocol class definition is used to create the new repository object.
- **Repositories:** a repository is a persistent storage mechanism for metadata and content. The repository definitions specify how to connect to the repository and how to allow client software to connect to a repository by name. You must create a repository definition for each repository your application is going to access. (You must also define a repository when using the `getPathUrl` method of the `MetadataInterface`.)
- **Repository groups:** a repository group identifies a set of repositories that can be searched together.
- **Smart objects:** smart objects are objects that act as wrappers for metadata entries in order to hide the details of repository-specific metadata types. A smart object definition consists of the following:
  - ◆ the protocol of the repository that contains the metadata
  - ◆ the interface for the smart object
  - ◆ the repository-specific type of metadata
  - ◆ the action to take to implement the object
  - ◆ the filter class to use to search for this type of object (object)

You can use smart objects to specify implementations (smart object action definition) for one or more repositories. You must specify an implementation (smart object action definition) for at least one repository type. In the smart object action definition, you can also specify a filter to use for implementing different smart objects for the same repository type.

- **Factories:** factories are objects that act as wrappers for metadata entries in order to hide the details of repository-specific metadata types. However, with factories, you can not specify an interface or filter to use when creating the object. In addition, within each factory, you can only specify implementations (factory object action definitions) for one type of repository. A factory definition consists of the following:
  - ◆ the protocol of the repository that contains the metadata
  - ◆ the repository-specific type of metadata
  - ◆ the action to take to implement the factory

**Note:** You must use smart object definitions if you wish to specify the following:

- ◆ an interface for the object
- ◆ a filter to use when implementing the object

- ◆ multiple repositories for the actions of an object

To configure the Information Service, follow these steps:

1. Open SAS Management Console and connect to a metadata repository.
2. In the navigation tree, expand the Foundation Services Manager tree to locate and select the Information Service that you want to modify.
3. Right-click the Information Service and select **Properties**. The Information Service properties window appears.
4. Select the Service Configuration tab. Click **Edit Configuration**. The Information Service Configuration window appears.
5. In the Protocols tab, click **New** to add a protocol or select a protocol and click **Edit** to edit a protocol. Enter the following information:

**Protocol**

specifies the protocol for the information service.

**Class**

lists the fully qualified Java class for the selected protocol. When requesting a connection to a new repository, this class is used in the `connect` method.

6. In the Repositories tab, click **New** to add a repository, or select a repository and click **Edit** to edit a repository. Enter the following information:

**Information Repositories**

lists the repositories for the specified protocol.

**Protocol**

specifies the protocol for the Information Service.

**Description**

specifies the repository description.

**Host**

specifies the fully qualified DNS name of the host where the repository server is running.

**Port**

specifies the TCP/IP port on which the repository server is listening.

**Domain**

specifies the authentication domain in which the repository server is running.

**Base**

specifies the base directory for the repository.

**Proxy**

specifies a URL for a proxy server.

**Auto-Connect**

when checked, specifies that the information service should automatically connect each authenticated user to the repository.

**Secure**

when checked (and if security is supported), specifies that the connection to the repository should be made using a secure protocol.

7. If you want to define repository groups, select the Groups tab. Click **New** to add a repository group, or select a repository group and click **Edit** to edit a repository group. Enter the following information:

**Name**

specifies the repository group name.

**Member Repositories**

specifies the repositories that are members of the repository group. Select a repository from the **Available Repositories** panel and click the arrow button to move it to the **Member Repositories** panel.

8. If you want to define smart objects, select the Smart Objects tab. Click **New** to add a smart object or select a smart object and click **Edit** to edit a smart object. Enter the following information:

**Name**

specifies the smart object type name. This string should exactly match the string returned from the smart object implementation's `getType()` method.

**Interface Class**

specifies the fully-qualified Java interface that objects of this type will implement.

**Filter Class**

specifies the fully-qualified Java class to use to most effectively search for objects of this type. This class will likely contain specific extensions to the `com.sas.services.information.Filter` class to make searches more efficient.

**Actions**

defines how and when objects of this type will be created. An action definition contains a protocol, a repository-specific type, a fully qualified Java class for the implementation to instantiate when that type is encountered, and an optional filter to run against an object which it must match for the action to be taken. Click **Add** to define a new action, or **Edit** to change an existing action and enter the following information:

**Protocol**

specifies the repository protocol that this action applies to. Select **omi** for Open Metadata Interface, **ldap** for LDAP directory server, or **dav** for WebDAV server.

**Type**

specifies the repository-specific type to look for when creating this type of object.

**Class**

specifies the fully qualified Java class to create when encountering this type in the repository.

**Filter**

specifies an optional filter which an object must validate against before this action is taken.

The format of the filter is

`[*association/]@attribute='value'`

*association*

specifies the name of an association from the specified repository type; the objects in the association will be tested against the attribute portion of the filter.

*attribute*

specifies an attribute to test for validation. The attribute can be an attribute on the objects in the association or, if no association is specified, an attribute can be an attribute on the object itself.

*value*

specifies the attribute value to test the object against to be sure it is the correct type.

9. If you want to define factory definitions, select the Factories tab. Click **New** to add a factory or select a factory and click **Edit** to edit a factory. Enter the following information:

**Protocol**

specifies the protocol for the Information Service.

**Types**

specifies the factory types associated with the Information Service and the selected protocol. You may select more than one factory type.

**Action**

specifies the action associated with the selected factory. The **Action** table lists the type, class, method, and filter for each action. Click **Add** to define a new action, or **Edit** to change an existing action and enter the following information:

**Type**



## SAS® 9.1.2 Integration Technologies: Administrator's Guide

specifies the action type. Select **Class** (to specify a class to generate the smart object), **Constructor** (to specify a constructor for a Java class that implements the smart object), or **Service** (to specify a Foundation Service).

### ***Filter***

specifies the fully qualified Java class to use to search for objects of this type. The class will most likely contain extensions to the `com.sas.services.information.Filter` class to make searches more efficient.

### ***Class***

specifies the fully qualified Java class to instantiate for the action.

### ***Method***

specifies the method for the action. This field is displayed only for action types of **Class** and **Service**.

10. Click **OK** to save the Information Service configuration to the metadata repository.

### *Foundation Services*

# Modifying the Logging Service Configuration

The Logging Service enables applications to:

- send runtime messages to one or more output destinations, including consoles, files, and socket connections.
- configure and control the format of information sent to a particular destination. Configuration can be performed through static configuration files or by invoking runtime methods that control logging output.
- perform remote logging, which involves sending log messages generated in one Java virtual machine (JVM) to another JVM.
- perform logging either by user session or by JVM.

For more information about the Logging Service, see [com.sas.services.logging](#) in the Foundation Services class documentation.

When a service deployment is defined and deployed, a base logging configuration is used to determine the appropriate output destinations. However, you can use SAS Management Console Foundation Services Manager to modify the Logging Service configuration and configure additional logging contexts and output destinations. The Logging Service configuration consists of the following items:

- **Contexts:** the logging context definition specifies the name and outputs for a specific logging context. In your application, you code the Logging Service to send information to a specific logging context. (The `RootLoggingContext` is used for any logging context that is not configured.)

When naming the logging context, you can specify the logging context name as part of a naming hierarchy. In a naming hierarchy, the logging context names are separated by a period (for example, `com.sas.services.event`). If a call to a logging context named `com.sas.services.event` is made and there is no logging context for `com.sas.services.event`, then the Logging Service looks for a logging context of `com.sas.services`. If there is no logging context for `com.sas.services`, then `com.sas` is used.

When you define a logging context, you associate outputs with the logging context in order to specify where to send logging messages for that particular logging context.

**Note:** To associate outputs with a logging context, you must first create the output definition.

- **Outputs:** the output definition specifies an output destination for the logging messages. The Logging Service can send the log messages to a file, console, or socket.

To configure the Logging Service:

1. In the SAS Management Console navigation tree, expand the Foundation Services Manager tree to locate and select the Logging Service that you want to modify. Right-click on the Logging Service and select **Properties**. The Logging Service properties appears.
2. Select the Service Configuration tab and click **Edit Configuration**. The Logging Service Configuration window appears.
3. On the Outputs tab, click **New** to add an output, or select an output and click **Edit** to edit an output. Enter the following information:

**ID**

specifies the name or identifier for the output.

**Type**

specifies the output type. Select **File**, **Console**, or **Socket**.

- If you select **File**, enter the following information:

**File**

specifies the file to use for output

**Append**

specifies whether to append the logging output to the existing output in the file. If the **Append** checkbox is unchecked, then any existing data in the file will be overwritten.

**ImmediateFlush**

specifies whether the log file is cleared after each logging statement.

- If you select **Console**, enter the following information:

**Target**

specifies the output destination. Valid values are **System.out** or **System.err**

**ImmediateFlush**

specifies whether the file is flushed after each logging statement.

- If you select **Socket**, enter the following information:

**Port**

specifies the socket port number.

**Host**

specifies the socket host.

**Layout Pattern**

specifies how to format the log message. For details about specifying layout patterns, see Pattern Layouts.

**Async**

specifies whether asynchronous logging is activated.

Click **OK** to return to the Logging Service Configuration window.

4. On the Context tab, click **New** to add a context, or select a context and click **Edit** to edit a context. Enter the following information:

**Name**

specifies the name of the context.

**Priority**

specifies the priority level of the logging context. The priority levels are

**DEBUG**

displays the informational events that are most useful for debugging an application.

**INFO**

displays informational messages that highlight the progress of the application.

**WARN**

displays potentially harmful situations.

**ERROR**

displays error events that might allow the application to continue to run.

**FATAL**

displays very severe error events that will probably cause the application to abort.

**Chained**

specifies whether the context is chained. Chaining designates that the log message is processed by both the current context and also by logging contexts higher in the logging context hierarchy.

**Outputs**

specifies the output destinations for the context. Click **Add** to add an output. The Add Logging Service Output window appears. Select an output and click **Add**. Click **OK**.

Click **OK** to return to the Logging Service Configuration window.

5. Click **OK** to save the new Logging Service configuration to the metadata repository.

# Pattern Layouts

The layout specifies how the output is formatted before it is sent to the output device. The layout is specified as a pattern string. The following table shows the characters available for use within layout pattern strings:

The following table shows the special conversion characters available for use within layout pattern strings:

Conversion Character	Result
<b>c</b>	Used to output the logging context. The logging context conversion specifier can be optionally followed by <i>precision specifier</i> , that is a decimal constant in brackets or braces. The precision specifier specifies the number of right most components of the logging context name that will be printed. For example, for the logging context name <b>a.b.c</b> the pattern <code>%c{2}</code> will output <b>b.c</b> .  If you do not specify a precision specifier, the logging context name is printed in full.
<b>d</b>	Used to output the date of the logging event. The date conversion specifier may be followed by a <i>date format specifier</i> enclosed between braces. For example, <code>%d{HH:mm:ss,SSS}</code> or <code>%d{dd MMM yyyy HH:mm:ss,SSS}</code> . If no date format specifier is given, then ISO8601 format is assumed.
<b>l</b>	Used to output location information of the caller that generated the logging event. The location information depends on the JVM implementation, but usually consists of the fully qualified name of the calling method followed by the caller's source, the file name, and line number all within parentheses. The location information can be very useful but its generation can cause performance issues.
<b>m</b>	Used to output the application supplied message associated with the logging event.
<b>n</b>	Used to output the platform dependent line separator characters. This conversion character offers similar performance to using non-portable line separator strings such as <code>"\n"</code> , or <code>"\r\n"</code> . Thus, it is the preferred way of specifying a line separator.
<b>p</b>	Used to output the priority of the logging event.
<b>r</b>	Used to output the number of milliseconds elapsed since the start of the application until the creation of the logging event.
<b>s</b>	Used to output the session ID associated with this logging event. The output for this conversion character will be an empty string if the Logger being used does not have an associated <code>SessionContext</code> .
<b>t</b>	Used to output the name of the thread that generated the logging event.
<b>u</b>	Used to output the user name associated with this logging event. The output for this conversion character will be an empty string if the Logger being used does not have an associated <code>SessionContext</code> , or if that <code>SessionContext</code> does not have an associated <code>UserContext</code> .
<b>%</b>	The sequence <code>%%</code> outputs a single percent sign.

# Modifying the Session and User Service Configurations

## Understanding and Editing the User Service

The User Service enables applications to:

- create, locate, maintain, and aggregate information about users of the SAS Foundation Services.
- store and retrieve User Context objects for sharing between applications. The User Context contains the user's active repository connections, identities, and profile.
- manage and access user profiles. A profile is a collection of name/value pairs that specify preferences and configuration or initialization data for a user for a particular application.
- access group profiles. A group profile specifies preferences and configuration or initialization data for a group of users for a particular application.

For more information, see [com.sas.services.user](#) in the Foundation Services class documentation.

The User Service utilizes a user context to hold the user's information for connections, identities, and profile. The profile then contains application profile data for the user. The User Service configuration consists of the following:

- **Users:** the user definition specifies the credentials that are associated with this User Service. The user definition consists of the user ID, password, and authentication domain of the user.
- **Profiles:** the profile definition contains a collection of name/value pairs that specify preferences and initialization data for a user of an application. The profile definition contains the name of the associated application, where the profile is located, the class and type of the profile, and a filter used to locate the profile.

To configure the User Service configuration:

1. In the SAS Management Console navigation tree, expand the Foundation Services Manager tree to locate and select the User Service you wish to modify. Right-click the User Service and select **Properties** from the pop-up menu. The User Service properties window appears.
2. Select the Service Configuration tab. Click **Edit Configuration**. The User Service Configuration window appears.
3. On the General tab, to add authentication domain or base LDAP information, enter the following information:

**Name**

specifies the default authentication domain name. If an entry for a user does not supply an authentication domain, this authentication domain name is used.

**People**

specifies the distinguished name (DN) for the context in LDAP that contains user metadata.

**Groups**

specifies the distinguished name (DN) for the context in LDAP that contains group metadata.

**Credentials**

specifies the location in LDAP that contains credential information.

4. On the Users tab, and click **Add** to add a user, or select a user and click **Edit** to edit a user. Enter the following information:

**ID**

specifies the user ID (for the SAS Metadata Server) or LDAP directory entry (for LDAP) of the user.

**Password**

specifies the password needed for the user to log on to the specified authentication domain.

**Domain**

specifies the authentication domain for which the user ID is valid.

Click **OK** to return to the User Service Configuration window.

5. On the Profiles tab, click **Add** to add a profile, or select a profile and click **Edit** to edit a profile. Enter the following information:

**Application**

specifies the application whose profile is specified.

**Domain URL**

specifies the location of the repository where the application profile is stored.

**Class**

specifies the class associated with the profile.

**Type**

specifies the profile type. If you are NOT using a custom profile class, leave this field blank.

**Filter**

specifies information to help locate the correct profile. If you are NOT using a custom profile class, leave this field blank.

Click **OK** to return to the User Service Configuration window.

6. When you are finished adding User Service configuration information, click **OK** to save the User Service configuration to a metadata repository.

## Understanding and Editing the Session Service

The Session Service enables applications to:

- create a session context. A session context is a control structure that maintains state information within a bound session, facilitating resource management and context passing.
- bind objects to a session context.
- use the session context as a convenience container for passing multiple contexts.
- use the session context as a convenience container for passing other services, such as User Services and Logging Services.
- notify bound objects when they are removed from the session context or when the session context is destroyed, so that objects can perform any necessary cleanup.

For more information, see [com.sas.services.session](#) in the Foundation Services class documentation.

When the Session Service initializes, it discovers the Logging Service, and obtains a default logging context. The Session Service then uses the Session Service configuration to determine whether to bind to a user context when creating the root session context:

- If the Session Service deployment configuration specifies a user context name, the Session Service discovers the User Service and obtains the default user context. The Session Service then creates a default root session context that is bound to this default user context.
- If the Session Service deployment configuration does not specify a user context name, the Session Service creates a default root session context that is not bound to any user context.

Applications can then use the root session context to track shared resources that are global to the application and to obtain the initialized logging context and default user context (if one was specified).

To configure a default user context name in the Session Service configuration:

## SAS® 9.1.2 Integration Technologies: Administrator's Guide

1. In the SAS Management Console navigation tree, expand the Foundation Services Manager tree to locate and select the Session Service that you want to modify. Right-click the Session Service and select **Properties** from the pop-up menu. The Session Service properties window appears.
2. Select the Service Configuration tab and click **Edit Configuration**. The Session Service Configuration window appears.
3. Specify the default **User Context Name**. Click **OK** to return to the Session Service Configuration window.
4. Click **OK** to save the Session Service configuration to the metadata repository.

### *Foundation Services*

# Monitoring Applications

The Application Monitor plug-in to SAS Management Console enables you to monitor the performance and activities of the various parts of a running application.

Before using the Application Monitor, you must first use the Logging Service that is supplied with the SAS Foundation Services installation package to code your applications to generate monitoring information. Then, you must configure the Logging Service in the Foundation Services Manager plug-in to SAS Management Console. You will then be able to monitor all of your applications' pertinent activities on demand from the Application Monitor.

In order to display monitor output using the Application Monitor, you will need to perform the following tasks:

1. Code your application using the Logging Service of SAS Foundation Services. For more information about the Logging Service provided by SAS Foundation Services, see [Logging Service](#) in the *SAS Integration Technologies Developer's Guide*.
2. Configure the Logging Service in the Foundation Services Manager to provide monitoring data. For more information about configuring the Logging Service, see the online Help for the Foundation Services Manager plug-in to SAS Management Console.
3. Add and display one or more monitors to the Application Monitor. For more information, see the online Help for the Application Monitor plug-in to SAS Management Console.
4. Edit monitor properties to customize how your output is displayed. For more information, see the online Help for the Application Monitor plug-in to SAS Management Console.

You can also edit a monitor's properties after adding it to the Application Monitor. *Stored Processes*



# Stored Processes

A stored process is a SAS program that is stored centrally on a server. A client application can then execute the program, optionally supply name/value parameters, and receive and process the results. For details about creating a stored process and processing the results, refer to Stored Processes in the *SAS Integration Technologies Developer's Guide*.

To make a stored process accessible to client applications, you can use the Stored Process Manager plug-in to SAS Management Console to create metadata that describes the stored process and its location. You can use the Stored Process Manager to do the following:

- register metadata for new stored processes
- modify metadata for existing stored processes

**Note:** The SAS Integration Technologies Administrator's Guide for stored processes provides information for administering SAS Stored Processes using SAS Open Metadata Architecture.

*Stored Processes*

# Defining Servers Used By Stored Processes

Before you set up a new stored process, you must define a server for the stored process to run on. The stored process server is the same machine that contains the stored process source code. You can [use the Server Manager](#) to define a stored process server.

The type of server you use depends on the type of results that you want the stored process to return. To get packages, you can use a SAS Stored Process Server or a SAS Workspace Server. If you want streaming results, then use a SAS Stored Process Server. For more information about how to choose which type of server to use, see [Choosing or Defining a Server](#) in the Stored Processes chapter of the *SAS Integration Technologies Developer's Guide*.

If you plan to publish a permanent package to a WebDAV server, you can also use the Server Manager to make sure that a WebDAV server is defined.

In terms of scalability, there are numerous server configurations that you can use in order to optimize load balancing. You can use the following server configurations:

- prestarted servers
- starting servers on demand, or reusable servers
- starting a new server for each client

The default server configuration uses the object spawner and load balancing.

*Stored Processes*

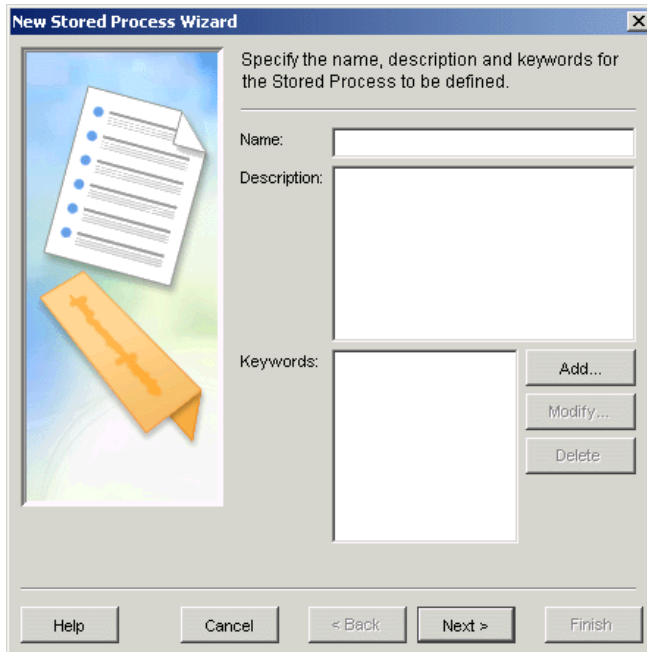
# Registering a New Stored Process

To enable a stored process to be invoked in an application, you can use the Stored Process Manager, a SAS Management Console plug-in, to register metadata that defines the name of the stored process and provides information about associated parameters.

**Note:** Before you can register a stored process, you must define the servers that the stored process will use.

To register metadata for a new stored process using the Stored Process Manager:

1. Open SAS Management Console and connect to a metadata repository.
2. From the SAS Management Console navigation tree, select the folder under Stored Process Manager in which you would like to create the new stored process. If you would like to create a new folder, you can use the New Folder Wizard by moving the mouse to the place in the navigation tree where you want to put the new folder. Select **Actions** ➔ **New Folder**. The New Folder Wizard displays.
3. After you have selected the name of the folder that is to contain the new stored process, select **Action** ➔ **New Stored Process**. The New Stored Process Wizard displays:



4. In the New Stored Process Wizard, do the following:
  - a. Enter a name for the stored process. Slashes, backslashes, and control characters cannot be used in this field.
  - b. Enter a description for the stored process. This step is optional.
  - c. Click **Add**, **Modify**, or **Delete** to specify keywords for the stored process. If you are defining a stored process to be called by a SAS BI Web Service, enter XMLA Web Service as a keyword. Otherwise, this step is optional.
  - d. Click **Next**.
  - e. Select a SAS server from the list.
  - f. Select a source repository from the list. Click **Manage** to add, modify, or delete repositories.
  - g. Enter the name of the source file for the stored process.
  - h. Select the type of output from the list. If you are defining a stored process to be called by a SAS BI Web Service, then specify Streaming as the stored process result type.

- i. Click **Details** to specify output details.
- j. Click **Next**.
- k. Click **Add Parameter**, **Add Group**, **Modify**, or **Delete** to specify the parameters and parameter groups used by the stored process. This step is optional.
- l. Click **Finish** to register the new stored process.

**Note:** After you have registered the stored process, use the Stored Process Properties window to modify the stored process.

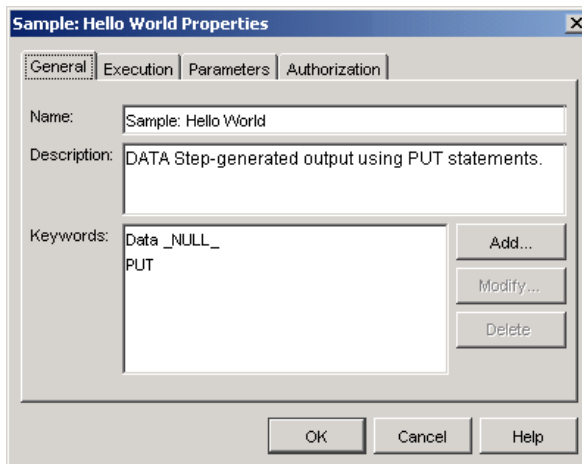
*Stored Processes*

# Modifying an Existing Stored Process

After a stored process has been registered, you might want to modify its properties. You can modify metadata for an existing stored process by using the Stored Process Properties window of the Stored Process Manager.

**To modify metadata for an existing stored process using the Stored Process Manager:**

1. Open SAS Management Console and connect to a metadata repository.
2. From the SAS Management Console navigation tree, under Stored Process Manager, select the stored process that you want to modify.
3. Select **File ► Properties**. The Stored Process Properties window displays:



4. If you want to modify the name, description, or keywords for the stored process:
  - a. Click the General tab in the Stored Process Properties window.
  - b. You can enter a different name for the stored process. Slashes, backslashes, and control characters cannot be used in this field.
  - c. You can modify the description for the stored process.
  - d. Click **Add**, **Modify**, or **Delete** to specify keywords for the stored process. If you are defining a stored process to be called by a SAS BI Web Service, enter `XMLA Web Service` as a keyword.
5. If you want to modify the execution information for the stored process:
  - a. Click the Execution tab in the Stored Process Properties window.
  - b. You can select a different SAS server from the list.
  - c. You can select a different source repository from the list. Click **Manage** to add, modify, or delete repositories.
  - d. You can enter a different name for the source file for the stored process.
  - e. You can select a different type of output from the list. If you are defining a stored process to be called by a SAS BI Web Service, then specify `Streaming` as the stored process result type.
  - f. Click **Details** to specify output details.
6. If you want to modify the parameters for the stored process:
  - a. Click the Parameters tab in the Stored Process Properties window.
  - b. Click **Add Parameter**, **Add Group**, **Modify**, or **Delete** to specify the parameters and parameter groups used by the stored process.
7. If you want to modify the authorization information for the stored process:
  - a. Click the Authorization tab in the Stored Process Properties window.
  - b. Select a user or group from the **Names** list box. If the name is not listed, click **Add** to access the Add Users and/or Groups dialog box.
  - c. Click **Delete** to delete a user or group from an access control definition.

## SAS® 9.1.2 Integration Technologies: Administrator's Guide

- d. Click **Access Control Templates** to assign permissions to the current resource by using an ACT.
  - e. In the permissions list, select check boxes to grant or deny permissions to the selected user or group; or deselect a check box to delete an explicit grant or deny permission.
8. Click **OK** to apply your changes to the stored process.

*Publishing Framework*

# Publishing Framework

The Publishing Framework provides a complete publishing environment for information delivery. The Publishing Framework enables both users and applications to publish SAS files (including data sets, catalogs, and database views), other digital content, and system-generated events to a variety of destinations, including the following:

- e-mail accounts
- message queues
- publication channels and subscribers
- WebDAV-compliant servers
- archive locations.

The Publishing Framework also provides tools that enable both users and applications to receive and process published information. For example, users can receive packages with content, such as charts and graphs, that is ready for viewing; and SAS programs can receive packages with SAS data sets that might in turn trigger additional analyses of that data.

The Publishing Framework plug-in to SAS Management Console provides an interface with which to administer the Publishing Framework. With the Publishing Framework plug-in, you can manage subscriber definitions and manage channel definitions.

For information about implementing the Publishing Framework capabilities in your applications, see Publishing Framework in the *SAS Integration Technologies Developer's Guide*.

**Note:** To publish to a subscriber who is defined with a WebDAV delivery transport on a secured WebDAV server, or to persist content on a secured WebDAV server or to an archive path on a secured HTTP or FTP server, the publisher must have credentials on that server. See Publishing to Secure Servers for details.

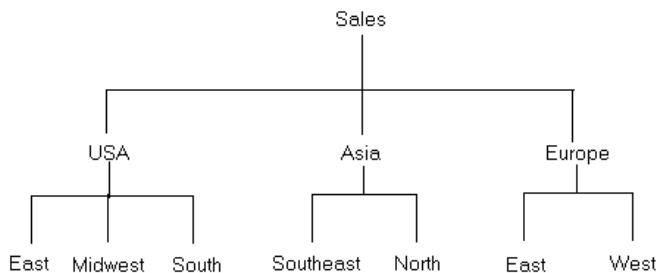
*Publishing Framework*

# Planning Your Publishing Solution

## Design Information Channels

Designing a successful publish and subscribe implementation starts with an understanding of why your organization is implementing the system. You will need to know, at a very basic level, what kind of information needs to be distributed to users and how widely that information needs to be distributed.

For example, you could start the planning process by understanding that your organization needs to disseminate sales information throughout the marketing organization and inventory data to the production organization. Starting with this base level of knowledge, you begin the process of breaking down the general categories of information into specific information channels by using a hierarchical model.



How you divide and subset the categories depends on your organization's needs, but you should work toward creating information channels as tightly focused as possible, without making them too tightly focused to be useful. Channels that are broadly defined leave users not knowing whether information delivered over the channel will be useful to them; channels that are too narrowly defined force users to subscribe to a long list of channels in order to ensure that they receive the information that they need.

To help focus the information that users receive, set up policies for name/value keywords. Name/value pairs are attributes that are specified when a package is published and that help to identify the package contents. Each subscriber definition can include a name/value filter that only allows packages that meet the subscriber's needs to be delivered.

For example, if you publish a package with a name/value attribute of `market=(Mexico)`, that package is only seen by those subscribers whose name/value filter indicates that they are interested in information about the Mexican market. Although the names and associated values can be anything that your organization finds useful, you must establish a list of acceptable keywords and values for those keywords. This list is essential for publishers to be able to provide consistent metadata that identifies published content and for subscribers to be able to filter published content in order to focus on the information they need.

When you define your information channels, you must also consider the users that will be accessing those channels as well as any restrictions that need to be placed on the channels. Although these aspects of planning are discussed separately and in more detail in the following two topics, in practice they are examined at the same time as you are defining your channels. You cannot define an information channel without first knowing who needs to see the information and how that information should be restricted.



## Identify Initial Subscriptions

When you plan an initial set of information channels, you must identify the users and groups that are initially subscribed to those channels. The information to set up these subscriptions is taken from the information you collected when you planned the channels. An understanding of your organization's need for a publish and subscribe system must include not only what information needs to be published, but also who needs to see that information.

However, you do not have to determine every piece of information that every individual needs to see. Rather, the process of planning initial subscriptions focuses on wider distributions of information, such as identifying the essential information that departments and groups of users need. How closely you follow this guideline depends on your organization's needs — there might be a few critical users who need to receive specific information, and there might be a need to subscribe a group of users to a tightly focused channel. In general, however, the initial subscriptions that you plan should be designed to distribute essential information to the largest number of users. Subscribers can request subscriptions to tightly focused channels as the need arises.

After you have determined the list of initial subscribers for each channel, you must determine how the information is to be distributed to users (whether by text- or HTML-formatted e-mail, with a WebDAV server, or through a queue) and identify their address information. The address information is essential for setting up the subscriber entries.

## Analyze Information Security Requirements

When you plan information channels you must also consider security for your publish and subscribe implementation in order to ensure that the information that is published on each planned channel is uniformly sensitive. For example, if you plan for a single channel to distribute accounting information throughout your organization, you will encounter a security problem when the accounting department needs to publish sensitive information (such as employee salaries). With only a single, unrestricted channel, you cannot publish the information to a specific set of users. In your consultations with users, you must identify information channels whose access needs to be controlled.

Your plan must address both methods that SAS Integration Technologies uses to implement security — authentication and authorization.

Authentication security involves the process of verifying that users are who they say they are. To authenticate users, servers use the host operating system's authentication provider, or they can use external LDAP or Microsoft Active Directory services. Therefore, you must implement authentication using the mechanisms provided by the host operating system authentication provider or alternative authentication provider. Authentication is a prerequisite for authorization.

Authorization security controls the information channels that users have access to. Without any security, users are able to subscribe to any information channel in your organization and access sensitive information. To prevent this situation, you must implement authorization security for each channel that you create. For more information about authorization security, see [Security](#).

## Configure Channels and Subscribers

Use the New Subscriber and New Channel wizards in the Publishing Framework plug-in to SAS Management Console to define the channels and subscribers that you identified during the planning phase. Begin by defining the subscribers; the New Channel wizard enables you to associate defined subscribers to a channel. See [Managing Channels](#) and [Managing Subscribers](#) for more information.

## Develop Applications That Deliver Content

After you set up the publish and subscribe infrastructure and implement the mechanisms that deliver content to a selected set of users, you must develop or modify applications that will be used to create the content to be published. These applications can take the form of stand-alone applications that are written in a visual programming language or SAS programs. See [Publishing Framework](#) in the *SAS Integration Technologies Developer's Guide* for information about the tools that are available to create a publishing application.

## Make Client Applications Available

After you develop or modify the applications that publish content, the initial structure of the publish and subscribe implementation is complete. Your next step is to make these applications available to users in your organization. Using the information that you gathered during initial planning, make the appropriate applications available to each user or group. Publishers must obtain or install the appropriate publishing application for their needs. For example, an individual or department that needs to publish data-intensive reports on a regular basis might use a SAS program for publishing, while a user who needs to send information to a changing number of users on an occasional basis might use the SAS Publisher application.

Subscribers must also obtain or install any appropriate software that is required to view published content. In particular, each subscriber must install the SAS Package Reader application in order to be able to view the contents of published SAS packages. For more information, see [SAS Package Reader](#) in the *SAS Integration Technologies Developer's Guide*. If the subscribers receive information through queues, they must also install the SAS Package Retriever. For more information, see [SAS Package Retriever](#) in the *SAS Integration Technologies Developer's Guide*.

## Announce Solution and Train Users

After the publishers and subscribers install the necessary applications, you can announce your implementation to your organization. You will also need to follow up the announcement with training for both publishers and subscribers, with training broken down by publishing methods, publishing needs, and subscriber applications.

*Publishing Framework*

# Managing Subscribers

## About Subscribers

A *subscriber* is a person who has a need for information that is published by the Publishing Framework. Before a user can receive information from a channel, you must define that user as a subscriber.

The Publishing Framework plug-in to SAS Management Console provides wizards that enable you to create subscribers. When you create a subscriber with a wizard, the subscriber object with the specified attributes is stored on the SAS Metadata Server.

You can create two different kinds of subscribers using the Publishing Framework: package subscribers and event subscribers.

- A *package subscriber* is a subscriber who is configured to receive packages. A package is a bundle of one or more information entities such as SAS data sets, SAS catalogs, or almost any other type of digital content.
- An *event subscriber* is a subscriber who is configured to receive events. An event is a well-formed XML document that can be published to an HTTP server, a message queue, or a channel that has event subscribers defined for it.

For each kind of subscriber, you can create individual subscribers and group subscribers. A group subscriber can contain individual subscribers or other group subscribers.

## Creating a New Subscriber

To create a new subscriber:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the Subscribers node.
4. For new package subscribers, select **Package Subscribers** and then select **Actions ▶ New Package Subscriber** or **New Subscriber Group** from the menu bar. For new event subscribers, select **Event Subscribers**, then select **Actions ▶ New Event Subscriber** or **New Subscriber Group** from the menu bar.

The appropriate wizard opens and guides you through the subscriber creation process. Click **Help** in the wizards at any time for detailed information. For examples, see [Example: Creating a Subscriber](#). The examples cover creating individual and group package subscribers; creating event subscribers is similar.

## Duplicating an Existing Subscriber

To create a new subscriber with substantially the same properties as an existing subscriber:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the Subscribers node.
4. Select either the Package Subscribers or Event Subscribers folder.
5. Select the existing subscriber and select **Actions ▶ Duplicate Package Subscriber** or **Duplicate Event Subscriber** from the menu bar to open the appropriate New Subscriber wizard. All of the wizard's fields are

filled in with values from the existing subscriber.

6. Because subscriber names must be unique, you must change the **Name** attribute. Click **Next** to change other attributes.
7. Click **Finish** to create the new subscriber.

## Modifying an Existing Subscriber

To modify the properties of an existing subscriber:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the Subscribers node.
4. Select either the Package Subscribers or Event Subscribers folder.
5. Select the subscriber whose properties you want to modify and select **File ▶ Properties** from the menu bar.  
The Properties window for that subscriber displays.
6. Use the tabs in the Properties window to modify the various properties of the subscriber. Some properties, such as the **User**, cannot be modified.
7. When you are finished modifying properties, click **OK**.

## Deleting a Subscriber

To delete a subscriber:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the **Subscribers** node.
4. Select either the Package Subscribers or Event Subscribers folder.
5. Select the subscriber that you want to delete and select **Edit ▶ Delete** from the menu bar.

To delete all package subscribers, select the Package Subscribers folder and select **Edit ▶ Delete** from the menu bar.

To delete all event subscribers, select the Event Subscribers folder and select **Edit ▶ Delete** from the menu bar.

*Publishing Framework*

# Delivery Transports

The *delivery transport* is a property of the subscriber that indicates how to deliver content to that subscriber. This section describes each of the available delivery transports and its attributes.

**Note:** For the WebDAV delivery transport, if the specified server is secured, then the publisher(s) must have credentials on that server. See [Publishing to Secure Servers](#) for details.

You can choose the following delivery transport options:

- None
- E-mail
- WebDAV
- Queue
- HTTP.

## None

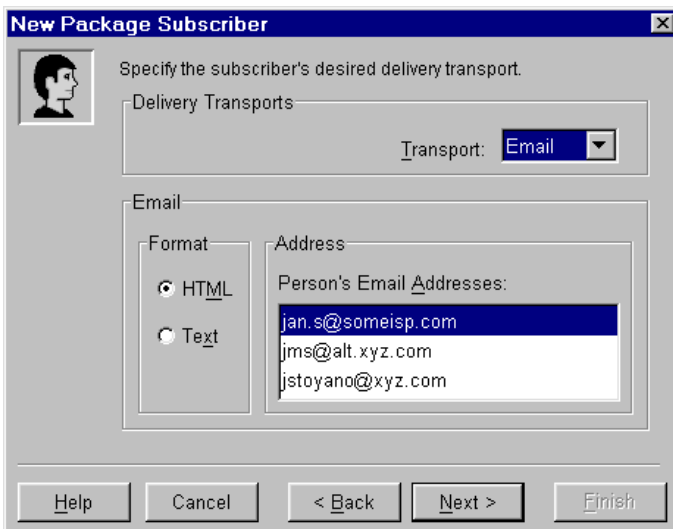
**Valid for:** Package and Event subscribers

If no delivery transport is specified for a subscriber, then that subscriber receives no published content.

## E-mail

**Valid for:** Package subscribers

With the e-mail delivery transport, the content package is delivered to the subscriber as an e-mail attachment. The attributes of the E-mail delivery transport are e-mail address and format (HTML or plain text).



## WebDAV

**Valid for:** Package subscribers

With the WebDAV delivery transport, the content package is published as a WebDAV collection to a location on a WebDAV-enabled server. To specify a WebDAV delivery transport, you must specify a WebDAV-enabled server and a base path. The relative path is optional. The base path and relative path are combined to form the URL that the subscriber uses to download the package. You must also specify a URL type (Parent or Collection). With a Parent URL type, the URL is the location *under which* the WebDAV collection is published. With a Collection URL type, the URL is the location of the WebDAV collection itself.

## Queue

**Valid for:** Package and Event subscribers

With the queue delivery transport, the content is published to an MQSeries or MSMQ message queue. The only attribute is the queue name.

For an MQSeries queue, the name of the queue is as follows:

```
MQSERIES://queueManager:queueName
```

*queueManager*

identifies the target queue manager.

*queueName*

identifies the name of the queue.

For an MSMQ queue, the name of the queue is as follows:

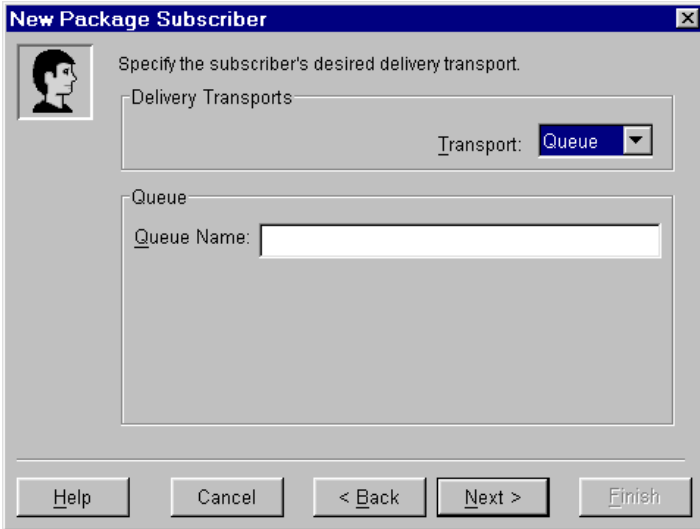
```
MSMQ://queueHostMachineName\queueName
```

*queueHostMachineName*

identifies the queue's machine name.

*queueName*

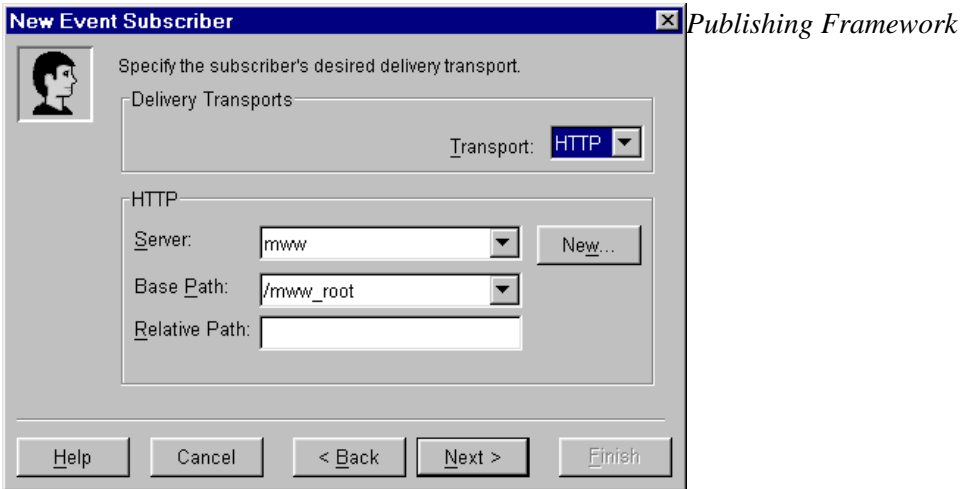
identifies the name of the queue



## HTTP

**Valid for:** Event subscribers

With the HTTP delivery transport, content is published to a location on an HTTP server. To specify an HTTP delivery transport, you must specify an HTTP server and a base path. The relative path is optional. The base path and relative path are combined to form the URL that is used to publish the event content.



# Filters

## What are Filters?

A *filter* is a property of a subscriber that enables that subscriber to receive only that content that meets certain criteria. Filters can be used to exclude content that the subscriber is not interested in, or that the subscriber's computing resources cannot handle. Filters can be defined based on the entry type, MIME type, or one or more name/value pairs that are defined for the content. A filter can be an *include* filter, which means that the subscriber receives all content that meets the filter criteria, or an *exclude* filter, which means that the subscriber receives all content that does not meet the filter criteria.

**Note:** For each type of filter (entry type, MIME type, or name/value pair), you can define either inclusion or exclusion filters (but not both). If you have previously defined exclusion name/value filters, for example, and then specify an inclusion filter, then all of the previously defined exclusion filters are deleted from the repository.

## Entry Filters

Each published package contains one or more entries. Each entry is one of several possible types. You can create a filter to include or exclude one or more entry types. Valid entry types include the following:

binary    catalog    dataset  
fdb        html        mddb  
reference   sqlview   nested\_package  
text        viewer

## MIME Type Filters

MIME types provide details about the information that is being published. For example, specifying the MIME type `audio/basic` indicates that the file is an audio file and requires software that can interpret such content.

You can define a filter that determines the type of information the subscriber receives. For example, a subscriber who is connecting with a modem might not want to receive some data types that may be large or unwieldy, such as movies or audio. By excluding those MIME types, the subscriber never encounters those types of information.

Some common MIME types include the following:

application/msword    application/octet-stream  
application/pdf        application/postscript  
application/zip        audio/basic  
image/jpeg            image/gif  
image/tiff             model/vrml  
text/html              text/plain  
text/richtext         video/quicktime  
video/mpeg



## Name/Value Pair Filters

Publishers can specify name/value pairs that describe the package that is being published. Knowledge of name/value pairs enables you to define filters for a subscriber that determine the packages that are received. If an inclusion name/value filter is defined for a subscriber, then the subscriber will receive only those packages that match the name/value filter.

A name/value pair is expressed as either a name or a relationship between a name and a value in the form

```
name < operator value >
```

- *name* is a variable to which a value can be assigned. *name* is not case-sensitive.
- *operator* relates the variable to the value. Commonly used operators are as follows:

Comparison Operators	Logical Operators
= (equals)	& (AND)
!= (not equal)	(OR)
? (contains)	

- *value* is a character string or numeric value. *value* is case-sensitive.

### Examples:

The following is an example of a package description using name/value pairs that a publisher has assigned to a published package:

```
market=(Mexico, US) type=report Quarter4 sales _priority_=low
```

Knowing the conventions that a publisher uses to describe packages helps subscribers to write meaningful filters. The following examples illustrate filter strings that determine whether the preceding example entity would be selected by the filter. If the package meets the filter conditions, then the package is delivered to the subscriber.

```
market=(US, Asia, Europe)
```

**No match.** Because the equals comparison operator (=) is used, the subscriber values and the publisher values that are assigned to the variable name MARKET must match exactly. In this example, the subscriber filters for US, Asia, and Europe, whereas the publisher assigns a value of Mexico and US. The conditions for selection are not met. Therefore, the package is not delivered to the subscriber.

```
market=(mexico, us)
```

**No match.** Because the equals comparison operator (=) is used, the subscriber values and the publisher values that are assigned to the variable name MARKET must match exactly. In this example, the subscriber values do not match the publisher values because of case differences.

```
market=US | market=Asia | market=Mexico
```

**No match.** Because the equals comparison operator (=) is used, the subscriber values and the publisher values that are assigned to the variable name MARKET must match exactly. In this example, although the OR operator (|) might seem to cause a matching condition, the equals operator (=) requires that each name/value pair that is separated by an OR operator (|) match the publisher name/value pair entirely. A match would result if the subscriber values were written as follows:

```
market=Mexico, US | market=Asia | market=Mexico
```

The first name/value pair in the series would match.

*market=(Mexico, US)*

**Match.** Because the equals comparison operator (=) is used, the subscriber values and the publisher values that are assigned to the variable name MARKET must match exactly. In this example, the value set does match.

*market=(US, Mexico)*

**Match.** Because the equals comparison operator (=) is used, the subscriber values and the publisher values that are assigned to the variable name MARKET must match exactly. In this example, the value set matches, regardless of the order of values within the value set.

*market?US & market?Asia & market?Mexico*

**No match.** The conditions that are specified in the subscriber name/value pair read: Variable name MARKET must contain the values US and Asia and Mexico. The contains comparison operator (?) identifies the eligible values for consideration. In this example, although the publisher variable MARKET contains US and Mexico, it does not also contain Asia. Because the logical AND operator (&) is used, its condition is not satisfied.

*market?US | market?Asia | market?Mexico*

**Match.** The conditions that are specified in the subscriber name/value pair read: Variable name MARKET must contain the values US or Asia or Mexico. The contains comparison operator (?) identifies the eligible values for consideration. In this example, the publisher variable MARKET contains US, and the logical OR operator (|) condition is satisfied.

*Quarter4=sales*

**No match.** Because the equals comparison operator (=) is used, the subscriber values and the publisher values that are assigned to the variable name QUARTER4 must match exactly. In this example, because the publisher variable name QUARTER4 does not contain a value and the subscriber variable name QUARTER4 does contain a value of sales, the value sets do not match.

*Quarter4*

**Match** Variable names are not required to have values. In this example, because the publisher variable name QUARTER4 does not have an assigned value and the subscriber variable name QUARTER4 does not have an assigned value, the value sets match.

*type=report & forecast*

**No match.** Two conditions must be met. The equals comparison operator (=) requires that the subscriber values and the publisher values that are assigned to variable name TYPE match. In this example, the first condition is met because both the publisher and the subscriber assign the value report to variable TYPE. However, the AND logical operator (&) requires that the variable name TYPE also be assigned the value forecast. Because the publisher variable name TYPE is not assigned a value of forecast, the final condition is not met.

*type=report & sales*

**Match.** Two conditions must be met. The equals comparison operator (=) requires that the subscriber value and the publisher value that are assigned to variable name TYPE match. In this example, the values match. Both assign the value report to the variable name TYPE. The AND logical operator (&) also requires that the variable name SALES match. Because both the publisher and the subscriber identify a variable name sales with no assigned value, the final condition is also met.

### *Publishing Framework*

# Managing Channels

## About Channels

A *channel* is a topic or identifier that acts as a conduit for related information. The channel carries the information from the publishers who created it to the subscribers who want it.

A channel has a name, a description, a subject, keywords, and a persistent store associated with it. A channel also has individual and group subscribers associated with it. Subscribers can be event subscribers or package subscribers.

The Publishing Framework plug-in to SAS Management Console provides a New Channel Wizard, which enables you to define all the properties of a channel, including what subscribers are associated with it. Each association of a subscriber to a channel is a subscription. A subscription enables the information that is published to a channel to be delivered to the interested (subscribed) users.

You should create a channel for each distinct topic or audience. For instance, users of a particular application might want a channel for discussion and data exchange, while the programmers of that application might want another channel to discuss technical problems and future enhancements. Although the topic is the same application, the discussion and data exchanged will be very different, so two separate channels would probably best serve the needs of the two groups of users.

## Create a Channel Folders

If you anticipate creating a large number of channels, then consider grouping related channels into channel folders. You can create subfolders within folders, thereby creating a folder hierarchy to which access controls can be applied.

To create channel folders:

1. From the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. If you are creating a top-level folder, then select Channels. If you are creating a subfolder, then navigate to and select the desired parent folder.
4. From the menu bar, select **Actions** ➔ **New Folder**. The New Channel Folder wizard displays.

The New Channel Folder wizard guides you through the process of creating a channel folder. Click **Help** in the wizard at any time for more information about the current window.

You can create a subfolder by selecting the desired parent folder and selecting **Actions** ➔ **New Folder** from the menu bar.

**Note:** Currently it is not possible to move an existing channel into a folder or from one folder to another. Plan ahead to avoid having to delete and recreate channels.

## Create a New Channel

To create a new channel:

1. From the SAS Management Console navigation tree, expand the Publishing Framework node.

2. Select the desired metadata repository node.
3. If you are creating a channel within a folder, select the Channels node and navigate to the desired folder.
4. Select the Channels item or the desired folder and select **Actions ▶ New Channel** from the menu bar to open the New Channel wizard.

The New Channel wizard guides you through the process of creating a new channel. Click **Help** in the wizard at any time for detailed information. For an example, see [Example: Creating a Channel](#).

## Duplicate an Existing Channel

To create a channel with substantially the same properties as an existing channel:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the Channels node and, if applicable, navigate to the folder where the existing channel is stored.
4. Select the existing channel and select **Actions ▶ Duplicate Channel** from the menu bar to open the New Channel wizard. All of the wizard's fields are filled in with values from the existing channel.
5. Because channel names must be unique, you must change the Name attribute. Click **Next** to change other attributes.
6. Click **Finish** to create the new channel.

## Modify an Existing Channel

To modify the properties of an existing channel:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the Channels node.
4. If applicable, open the appropriate folder(s) to navigate to the desired channel.
5. Select the channel that you want to modify and select **File ▶ Properties** from the menu bar. The Properties window for that channel displays.
6. Use the tabs in the Properties window to modify the various properties.
7. When you are finished modifying properties, click **OK**.

## Delete a Channel

To delete a channel:

1. In the SAS Management Console navigation tree, expand the Publishing Framework node.
2. Select the desired metadata repository node.
3. Select the Channels node.
4. If applicable, open the appropriate folders to navigate to the desired channel.
5. Select the channel that you want to delete and select **Edit ▶ Delete** from the menu bar.

To delete a channel folder (including any subfolders and channels under it), right-click the folder and click **Delete**.

To delete all channels, right-click the **Channels** item and click **Delete**.



# Persistent Stores

A channel can be defined to have a persistent store. A *persistent store* is a location where published content is permanently stored (or *persisted*). The Publishing Framework publishes the content to the persistent store location, and then publishes the content to the subscribers.

This section describes each of the available persistent store options and their attributes.

**Note:** To persist content on a secured WebDAV server, or to an archive path that is defined as a secured HTTP or FTP server, the publishers might need credentials on that server. See [Publishing to Secure Servers](#) for details.

You can choose the following persistent store options:

- None
- Archive (including File, FTP, and HTTP)
- WebDAV.

## None

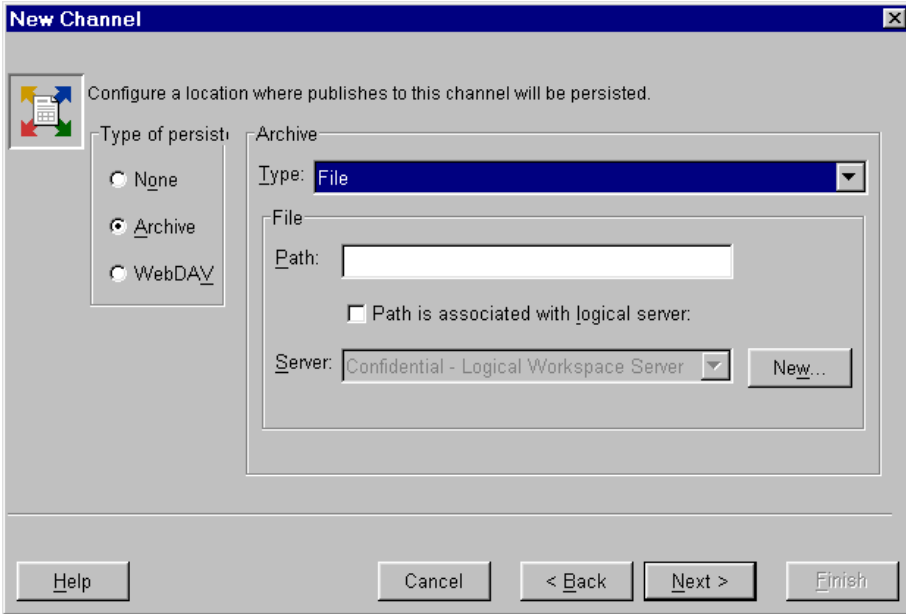
If you do not specify a persistent store for a channel, then all content that is published on that channel is published directly to the subscribers and is not persisted.

## Archive Persistent Stores

With an archive persistent store, the Publishing Framework publishes the content as an archive (binary `.spk`) file to the persistent store location. An archive persistent store can be defined as a physical file location, an FTP server, or an HTTP server.

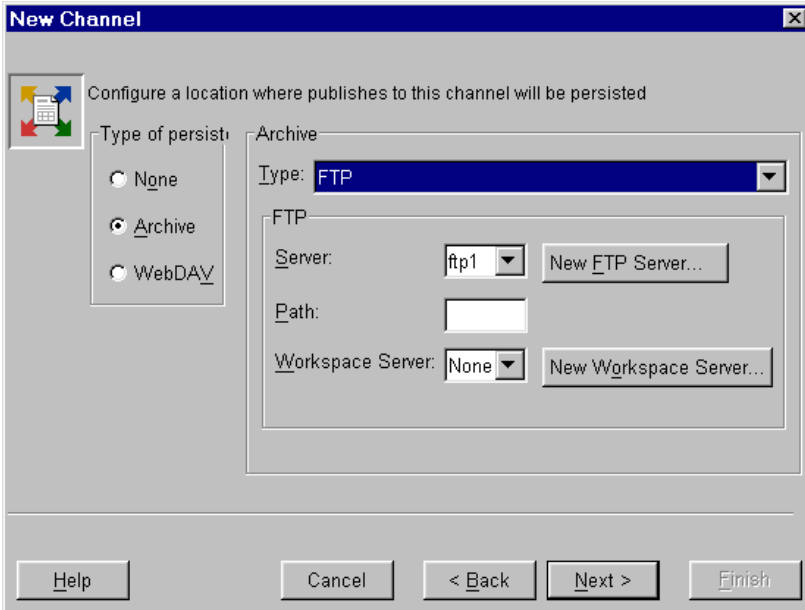
### File

For an archive persistent store that is defined as a physical file location, you must specify a file path. You can optionally associate the file path with a logical server if you want to be able to retrieve the archive file from a remote host.



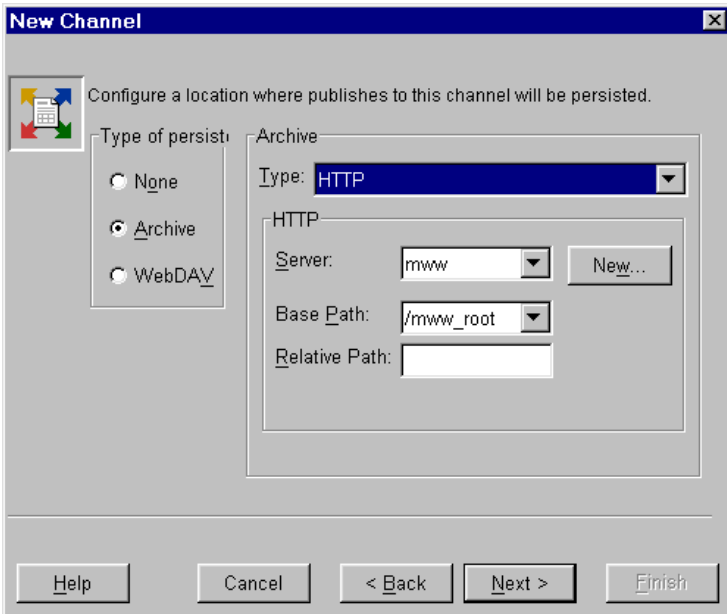
## FTP Server

For an archive persistent store that is defined as an FTP server, you must specify the name of the FTP server. You can optionally specify a path within the FTP server and a logical workspace server. The Publish Service component of the SAS Foundation Services needs an IOM Workspace server defined in order to publish an archive to an FTP location or to delete a file from an FTP location.



## HTTP Server

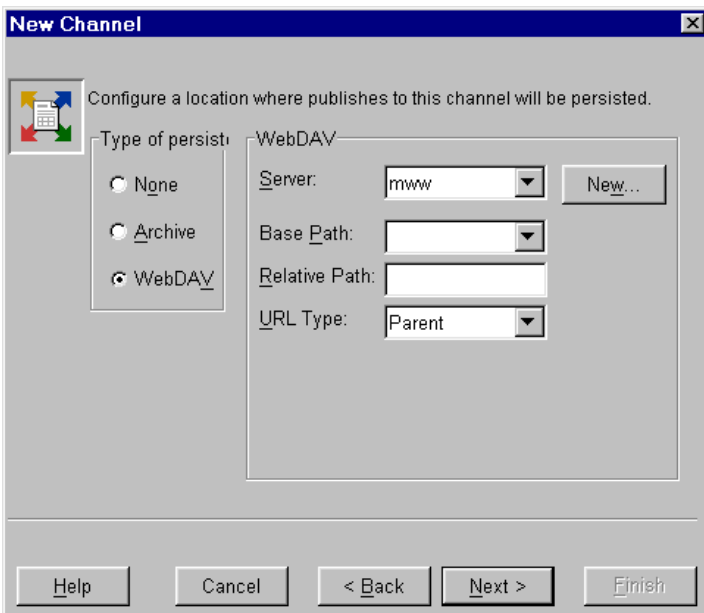
For an archive persistent store that is defined as an HTTP server, you must specify an HTTP server and base path. You can optionally specify a relative path. The base path and relative path are combined to form the URL of the location where the archive is persisted.



## WebDAV Persistent Store

For a WebDAV persistent store, you must specify a WebDAV-enabled HTTP server and base path. You can optionally specify a relative path. The base path and the relative path are combined to form the URL where the WebDAV collection is persisted. You also must specify a URL type of either Collection or Parent:

- Collection: WebDAV collection is persisted **in** the specified URL
- Parent: WebDAV collection is persisted **under** the specified URL



## See Also

[Administering HTTP Servers and WebDAV](#) in the *SAS Integration Technologies Server Administrator's Guide*.





# Publishing to Secure Servers

Under certain circumstances, when publishing to a channel, the user who is publishing the content (the publisher) will need credentials in order to connect to a server. The following example scenarios all require the publisher to have server credentials:

- publishing to a subscriber with a delivery transport that is defined as a secured WebDAV server
- publishing to a channel's persistent store that is defined as a secured WebDAV server
- publishing to a channel's persistent store that is defined as an archive path that is a secured HTTP server
- publishing to a channel's persistent store that is defined as an archive path that is a secured FTP server

In all of the above scenarios, the publisher needs access to the credentials in order to connect to that server. Because various users can also be publishers, the login information should not be defined in the individual publisher definitions. Instead, the logins should be defined in a group that can be accessed by all publishers. There are two major steps to creating this group:

1. Define the subscribers and persistent stores.
2. Add credentials to the group.

## Define the Subscribers and Persistent Stores

To define the subscribers and persistent stores, do the following:

1. Identify the package subscribers whose delivery transport will be defined as WebDAV and the event subscribers whose delivery transport will be defined as HTTP. From each of these users for which the HTTP server is secured, obtain a login that will be available for the publisher to access the HTTP server. Using the Publishing Framework plug-in to SAS Management Console, define these package and event subscribers.
2. Using the Publishing Framework plug-in to SAS Management Console, define the channel(s) whose persistent store will be defined as a secured WebDAV server, and the channels whose persistent store will be an archive path that is defined on a secured HTTP or FTP server. The appropriate server logins should be available for the publisher to access these servers.

If you use the Xythos WFS WebDAV server, see [Implementing Authentication and Authorization for the Xythos WFS WebDAV Server](#) in the *SAS Integration Technologies Server Administrator's Guide* for more information about security.

## Add the HTTP Credentials, FTP Credentials, and Publishers to a Group

To add the appropriate credentials and publishers to the group, do the following:

1. Using the User Manager plug-in to SAS Management Console, define a group that will contain all logins that are needed to access WebDAV, HTTP, and FTP servers.
2. Add to this group the logins that are needed to access the secured WebDAV servers that are defined as persistent stores, and the secured HTTP and FTP servers that are defined as persistent store archive paths. These logins are needed when the persistent store for a channel is defined as a secured WebDAV, HTTP, or FTP server.
3. Add to this group the logins for all package subscribers whose delivery transport is defined as a secured WebDAV server, and all event subscribers whose delivery transport is defined as a secured HTTP server.

## SAS® 9.1.2 Integration Technologies: Administrator's Guide

Obtain the authentication domain, user name, and password for each of these subscribers.

4. Add any users who will be publishing content.

**Note:** Logins can be added either as group logins or user logins.

**Tip:** Each login within this group should have a unique authentication domain so that each domain has a specific login to use. If more than one login has the same domain, then the Publishing Framework tries each login until it finds one that works. Because you cannot specify the order in which the Publishing Framework tries logins for a given authentication domain, there is no guarantee that the first successful login will be the desired login.

## See Also

For more information, see the following topics in the *SAS Integration Technologies Server Administrator's Guide*:

- [Administering HTTP Servers and WebDAV](#)
- [Implementing Authentication and Authorization for the Xythos WFS WebDAV Server](#)
- [Defining SAS Users, Groups, and Login Definitions](#)

*Publishing Framework*

# Example: Creating a Subscriber

## Creating an Individual Subscriber

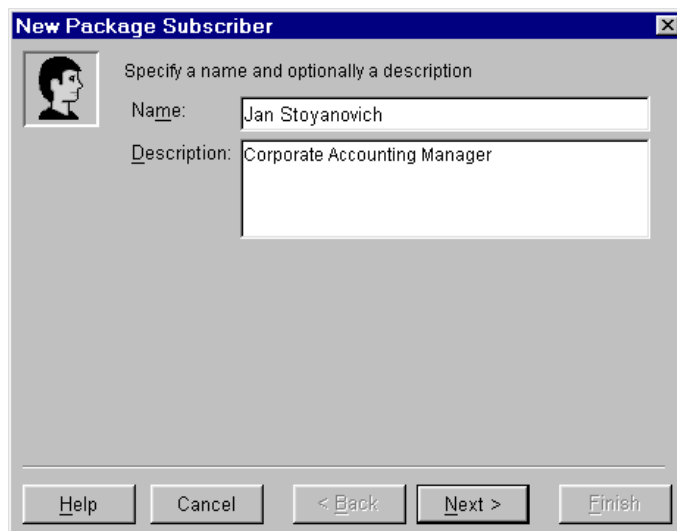
The New Package Subscriber wizard and New Event Subscriber wizard in SAS Management Console guide you through the process of creating, respectively, a new package subscriber and a new event subscriber. (See [Managing Subscribers](#) for information about opening the New Package Subscriber wizard and the New Event Subscriber wizard.) In this example, an individual package subscriber is created using the New Package Subscriber wizard.

**Note:** The process of creating an individual event subscriber is similar, except for the following:

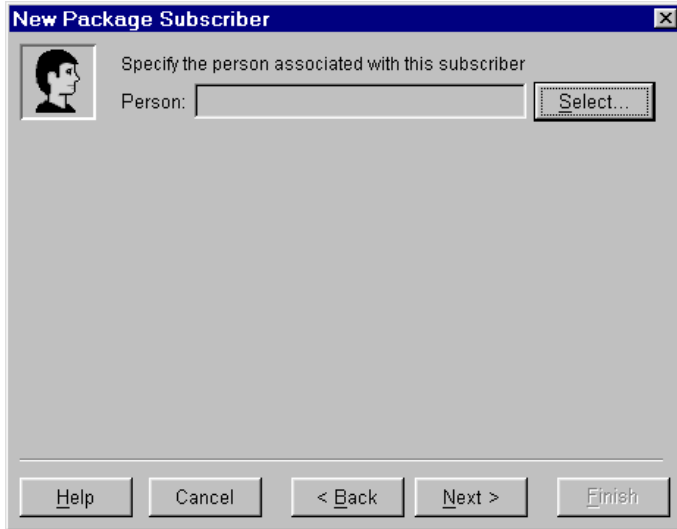
- You cannot specify filters for an individual event subscriber.
- The available delivery transports for individual event subscribers are HTTP and Queue.

To create an individual package subscriber, do the following:

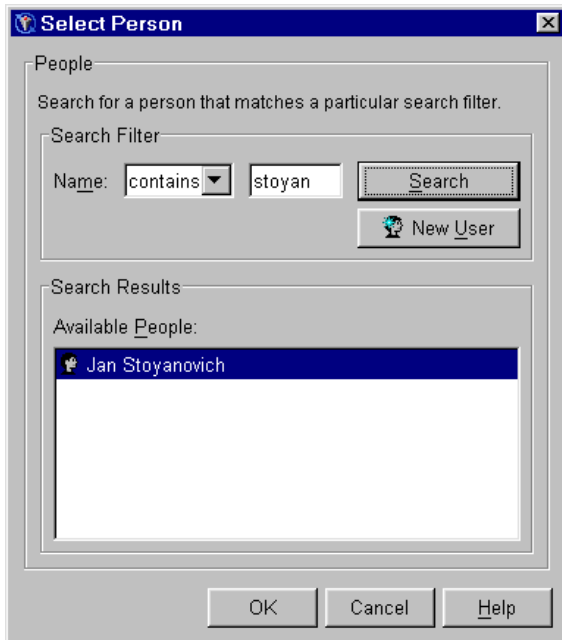
1. Specify a name and a description for this subscriber. The name must be unique within its parent folder. The description is optional.



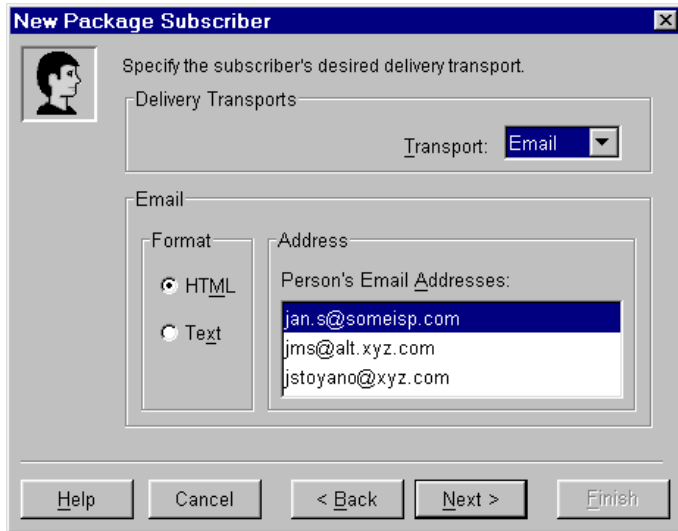
2. Click **Next**.
3. Click **Select** to associate a person with this subscriber.



- The Search Filter enables you to search the repository for users whose names either contain or are equal to a string that you specify. Enter the string in the text field, select either *contains* or *equals* from the drop-down list, and click **Search**. A list of users whose names meet your search criteria appears in the *Available People* list.

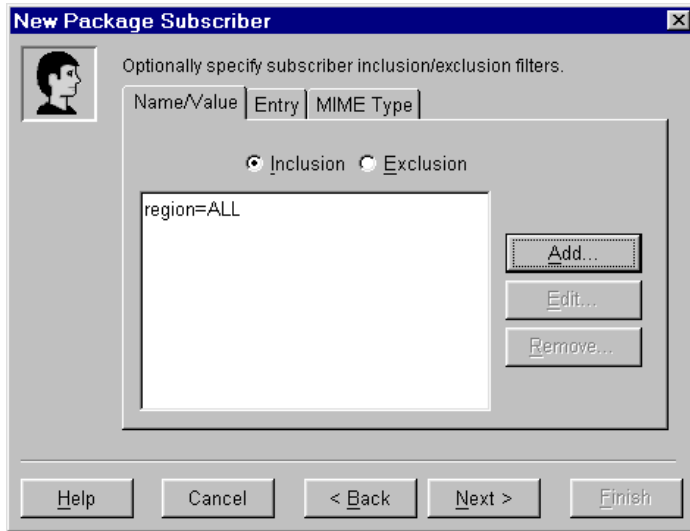


- If the desired user does not exist in the repository, then click **New User** to define that user.
- Then, select the desired user from the *Available People* list and click **OK**.
- Click **Next**.
- Select the subscriber's delivery transport. For this example, *Email* is selected from the *Transport* drop-down list. Other options are *WebDAV*, *Queue*, and *None*. For more information about delivery transports, see [Delivery Transports](#).
- Specify the attributes for the selected delivery transport. For this example, the e-mail format is selected to be HTML, and one of the user's e-mail addresses is selected.



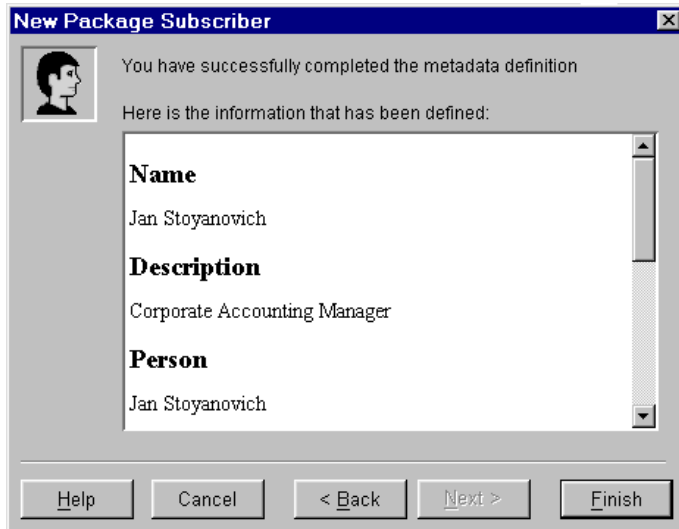
10. Click **Next**.

11. Specify one or more filters to eliminate content that the subscriber does not want to receive. To add a filter, select the tab that corresponds to the type of filter (Name/Value, Entry, or MIME Type). Select **Inclusion** or **Exclusion** and then click **Add** to specify the filter criteria. See [Filters](#) for more information.



12. Click **Next**.

13. Review the subscriber specifications. Click **Back** to make any corrections. Click **Finish** when you are satisfied with your selections.

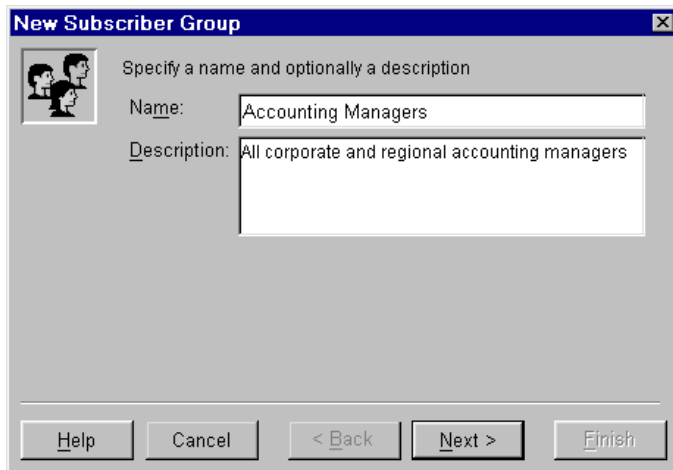


## Creating a Group Subscriber

The New Subscriber Group wizard guides you through the process of creating a subscriber group. (See [Managing Subscribers](#) for information about opening the New Subscriber Group wizard.) In this example, a group package subscriber is created. The process of creating a group event subscriber is identical.

To create a group subscriber, do the following:

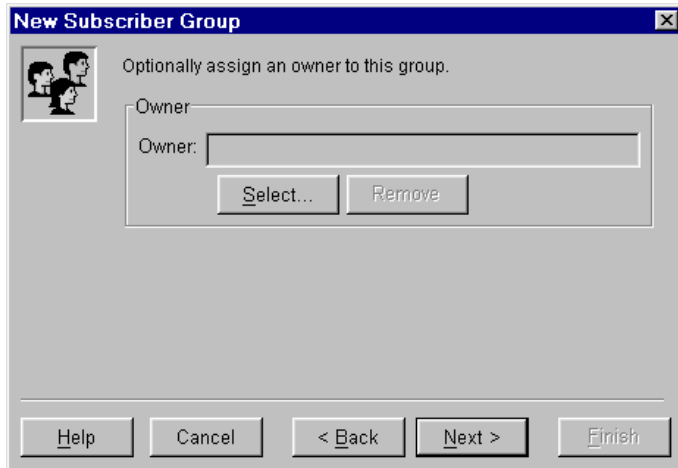
1. Specify a name and a description for this subscriber group. The name must be unique within its parent folder. The description is optional.



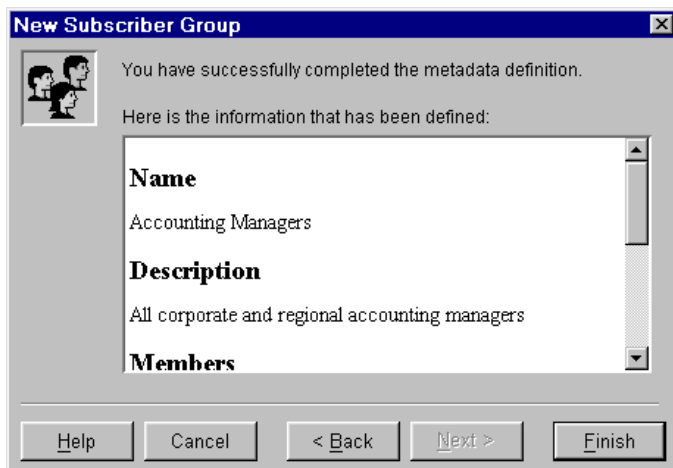
2. Click **Next**.
3. Associate members with the subscriber group. The **Available** list comprises all individual and group subscribers for this type of subscription (package or event). Select one or more subscribers from the **Available** list and click the right arrow to move them to the **Selected** list. To move all users to the **Selected** list, click the double-right arrow.



4. Click **Next**.
5. Optionally, assign an owner to this group. The **Owner** value is for information purposes only and is not used by the software. Click **Select** to open the Select Person dialog box. The owner is chosen from among all known users and does not need to be a subscriber.



6. Click **Next**.
7. Review your specifications. Click **Back** to make any corrections. Click **Finish** when you are satisfied with your selections.





*Publishing Framework*

# Example: Creating a Channel

The New Channel wizard guides you through the process of creating a new channel. (See [Managing Channels](#) for information about how to open the New Channel wizard.)

To create a channel using the New Channel wizard, do the following:

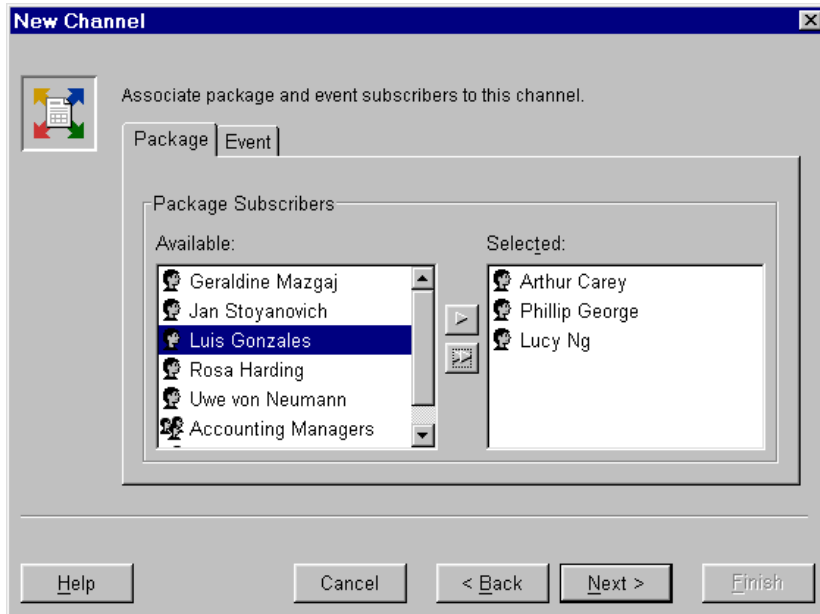
1. Specify a name for the channel. The channel name must be unique within its folder (if it is in a folder) or within the Channels node (if it is not in a folder).
2. Optionally, specify a description and a subject for the channel. The **Subject** can be used to provide a general "short description" of the channel's purpose.
3. Optionally, specify one or more keywords for the channel. **Keywords** enable you to provide more detailed description and can also be used in keyword searching. To add a keyword (which can be a single word or a phrase), click **Add** and specify the keyword in the Add Keyword dialog box. Click **OK** to add your keyword to the keyword list.

The screenshot shows the 'New Channel' wizard dialog box. The title bar reads 'New Channel'. The main area contains a list of fields for channel configuration:

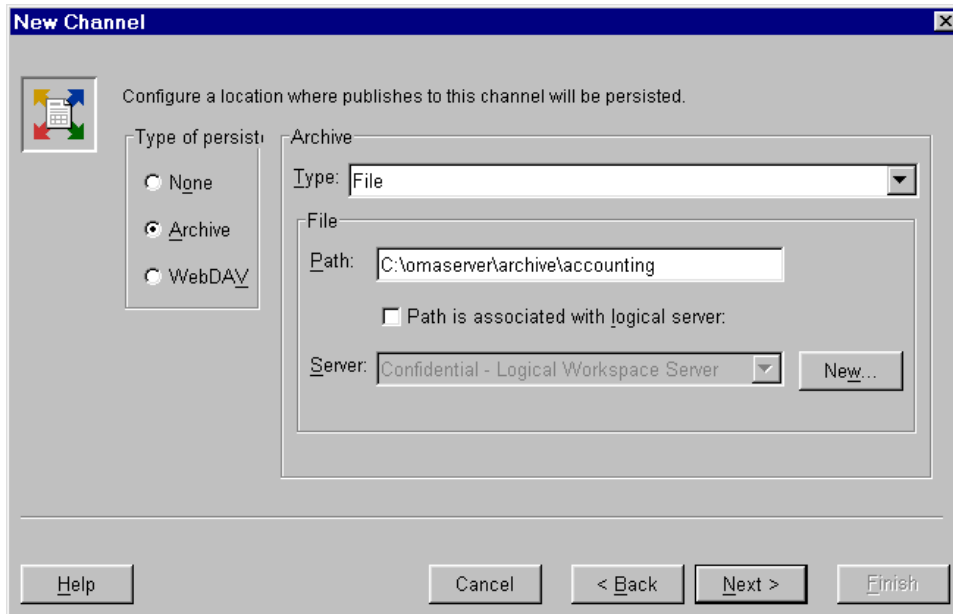
- Name:** Accounting
- ID:** A51VJOZC.\$0000012
- Description:** Accounting reports for each region and overall
- Subject:** accounting reports
- Keywords:** accounts receivable, accounts payable, balance sheets, profit and loss

Buttons for 'Add...', 'Edit...', and 'Remove' are located to the right of the keywords list. At the bottom of the dialog are buttons for 'Help', 'Cancel', '< Back', 'Next >', and 'Finish'.

4. Click **Next**.
5. Optionally, select subscribers to associate with this channel. Use the Package and Event tabs to select package subscribers and event subscribers. For each type, select zero or more subscribers in the **Available** list and click the right arrow to move them to the **Selected** list. Use the double-right arrow to move all subscribers from the **Available** list to the **Selected** list.



6. Click **Next**.
7. Select a type of persistent store and specify the attributes for that persistent store. See [Persistent Stores](#) for more information. For this example, **Archive** is selected, and a file on the local disk is specified as the persistent store location.



8. Click **Next**.
9. Review your selections. Click **Back** to make any corrections. Click **Finish** when you are satisfied with your selections.

