SAS Publishing

SAS® 9.1 Open Metadata Interface

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *Getting Started with the SAS® 9.1 Open Metadata Interface*. Cary, NC: SAS Institute Inc.

Getting Started with the SAS® 9.1 Open Metadata Interface

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

1st printing, January 2004

Table of Contents

Table of Contents

Getting Started with the SAS® 9 Open Metadata Architecture

This guide provides a brief description of metadata, metadata management, and the benefits of using the SAS Open Metadata Architecture (OMA). It shows you *how* to use the OMA by walking you through the steps of identifying what information you need to capture, selecting the metadata types best suited for your information, and setting up an Open Metadata Server and a metadata repository on Windows NT.

This guide does not describe how to use the OMA Configuration Utility to set up an Open Metadata Server, or how to set up the server to control access to metadata. See the **SAS 9.1 Open Metadata Server: Setup Guide** for this information.

Before You Begin

Software Requirements

The topics in "Working with Metadata" explain in detail how to set up an Open Metadata Server and use an OMI client in a Windows NT operating environment. To follow along, you need

- SAS Software, Version 9
- SAS Integration Technologies Software, Version 9 (shipped with Version 9 of SAS Software)
- SAS Management Console Software, Version 9 (SAS Management Console software must be installed from the *SAS® Client−Side Components* CD−ROM that is shipped along with Version 9 SAS Software.)
- the appropriate software for the intended programming environment.

The OMA supports Java, Visual Basic, C++, and SAS OMI clients.

Additional Reading

This guide introduces basic XML and XSL concepts. If you are not familiar with these languages, we recommend the following information sources:

- XML Web page: http://www.w3.org/TR/1998/REC−xml−19980210
- **Java and XML (O'Reilly Java Tools)**, by Brett McLaughlin and Mike Loukides (Cambridge, Mass: O'Reilly, 2000).
- **XSLT: Programmer's Reference**, by Michael Kay.

Topics Overview

The topics in this guide use a sample scenario to guide you through the steps of using the Open Metadata Architecture. The scenario is not intended to represent the *only* way the OMA is used. Rather, it is provided as a very simple example of how it *can* be used.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface
The correct bibliographic citation for this manual is as follows: SAS Institute Inc., Getting Started with the SAS® 9.1 Open Metadata Architecture Cary, NC: SAS Institute Inc., 2003.

Getting Started with the SAS® 9.1 Open Metadata Architecture

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

April 2003

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

What Is Metadata?

Metadata is information about the data resources in an organization, or in simpler terms, data about data. Typically in the IT industry, we talk about "inventory data," "personnel data," "budget data," and "payroll data." The first word, a modifier, describes the data and classifies it as belonging to a certain business function. This is metadata. It tells us what the data *is*.

Metadata is also information about how the data is used. To understand this definition, consider the following example: "$100.00" is a piece of data. It could be payroll data, personnel data, inventory data, or budget data. Under the expanded definition,

- metadata is information that provides meaning and context to the piece of data. It tells us that "$100.00" is a monetary amount in U.S. dollars, expressed in terms of dollars and cents.
- metadata also tells us how to understand the way the data is expressed or represented. Metadata helps us to *understand* the data.

In any organization, there are two types of metadata:

Technical metadata

describes the physical nature of the data, how the data was created, and how it is managed. This type of metadata is often machine–readable. Borrowing from the previous example, the fact that $100.00 is a monetary value and how it is expressed is physical data. Other examples of physical metadata might answer questions such as

◊ What is the origin of the data? Does it come from an external source, or is it generated internally?
◊ Where does the data reside? Is it in a SAS table or some other structure?
◊ On which server is the structure stored?

Informational metadata

describes business rules and definitions on which the data is based. This type of metadata is often intended for people rather than machines. It is informational metadata that tells us whether the $100.00 value is payroll data, personnel data, inventory data, or budget data. Informational metadata would also answer questions such as

◊ Who is responsible for the accuracy of the data? How can I contact him or her?
◊ What business process produced this data? How do I execute the business process?
◊ Which applications should (and do) have access to this data?

Previous Page | Next Page | Top of Page

What Is Metadata Management?

If metadata helps us to understand data, metadata management enables us to *use* the metadata. Metadata creation is time–consuming and expensive. To be truly useful, once stored, metadata must be centrally available and easy to maintain.

The primary goals of metadata management are

- to promote metadata conformity to enable sharing of metadata by an organization's applications. Metadata that is defined for one application can be copied and easily adapted for use by another application.
- to provide a common, centralized method of searching and managing distinct collections of metadata.

Both goals lower the costs of metadata development and maintenance by promoting standardization and reducing redundancy. Furthermore, when these goals are achieved, metadata can provide meaningful and valuable information, for example,

- impact analysis of technical changes within an organization
- comprehensive technical reporting about the organization's application systems.

Impact analysis gauges the effect of a single technical change on all of the applications in an organization. For example, if an organization stores metadata about its computer systems, it can use that metadata to easily determine which applications will be affected by taking a specific server offline. Or, if all applications store client address information in an address object and a change is needed in the way this information is stored –– for example, to surface street address, city/state, and country as three separate fields instead of one –– the change is easy to identify, make, and propagate.

As electronic data transfers and e–commerce increase, the soundness of the metadata supporting these transactions will become as important as the data itself. Support for industry metadata models and data interchange standards enables organizations to respond quickly and economically to rapidly evolving, external reporting obligations.

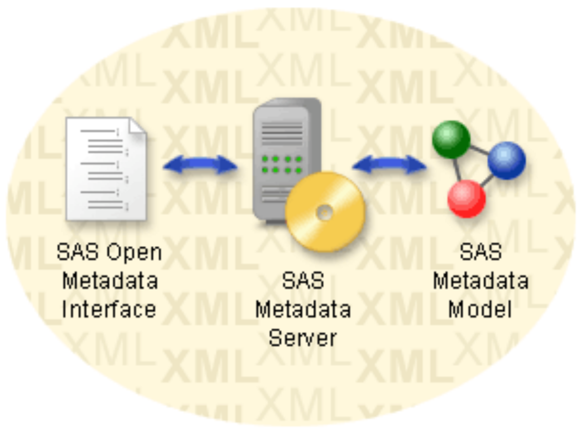
Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

What Is the SAS Open Metadata Architecture?

The SAS Open Metadata Architecture is a general–purpose metadata management facility that provides common metadata services to SAS applications. The metadata architecture provides

- **the SAS Metadata Server,** a central, shared location for storing metadata
- **the SAS Open Metadata Interface,** an application programming interface (API) that provides access to the server from a variety of programming environments, including Java, COM/DCOM, and SAS
- **the SAS Metadata Model,** a set of metadata types that are used for saving metadata on the server
- **an XML transport format and XML representation of metadata,** which makes it easy to transform the metadata to HTML and other standard XML representations, like the Object Management Group's Common Warehouse Metamodel (CWM).



Benefits of the SAS Open Metadata Architecture

- The SAS Metadata Model defines metadata types for the most commonly used objects and provides a mechanism for extending the metadata types with application–specific attributes and associations. This enables SAS applications to use a common model for most metadata objects while retaining the custom features that make them unique.
- Metadata objects are stored in application–specific repositories, which are managed by a repository manager. This tiered management approach enables metadata to be maintained separately yet accessed centrally through the repository manager, and guards the integrity of application–specific metadata while enabling global searching.
- A single tool set can be used to create, access, and manage metadata for all of your SAS applications.
- Support for the XML transport format and industry standard metadata models increases the likelihood of compatibility between SAS applications and other software applications.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Planning a Repository

As a statistician for a major pharmaceuticals company, Dr. Joe E. Doe maintains hundreds of SAS tables that contain the results of clinical trials for a new cancer inhibitor. He must report the results of these trials to the Food and Drug Administration in order to gain acceptance of the drug, and he wants to use the SAS Open Metadata Architecture to track his data. The SAS Open Metadata Architecture will provide

- a common model for storing the study metadata
- centralized access to this metadata
- the ability to reuse the metadata for other clinical trials projects
- ease of metadata transformation.

Task Overview

These are the steps for using the SAS Open Metadata Architecture:

1. Deciding What Information Should Be Stored
2. Selecting the Appropriate Metadata Types
3. Overview of Setting up a Server
4. Overview of Adding Metadata Objects
5. Overview of Querying the Repository.
6. Update and delete metadata objects in the repository.

Previous Page | Next Page | Top of Page

Deciding What Information Should Be Stored

The first step in deciding what information to store is determining what information you want to save. The following questions can help you to identify the metadata that you need to track:

- What is the data that you want to describe?
- What is the origin of the data, for example: what business process(es) produced the data? What technical process(es) produced the data?
- Who is responsible for the data?
- How is the data stored?

At the most basic level, Dr. Joe E. Doe's clinical trials consist of the following items:

- data describing the participants in the study
- data describing their visits to a doctor
- SAS programs that analyze the input and return a result that can be used to gauge the drug's efficacy
- documentation that describes the study variables and the hypotheses being tested
- study notes and other documentation.

This gives us a general idea of what information we need store. For more specific input, we need to look at the contents of an individual study, which we will call Study 1:

- Patient data is stored in a SAS table named Patient Information. This table contains the columns patient ID, initials, sex, date of birth, sponsor patient ID, weight in pounds, and weight in kilograms. The weight in kilograms is calculated by dividing the weight in pounds by 2.2.
- The data describing doctor visits is stored in a table called PatientVisits. This table contains the columns patient ID, systolic blood pressure, diastolic blood pressure, visit number, and occurrence number.
- The SAS program used in this study is a DATA step that merges the tables to create a third table called Study1Output. Study1Output contains three columns: patient ID, visit number, and a calculated column called SBPW_Coefficient. SBPW_Coefficient is created by multiplying the values of two of the columns in the input tables (Systolic_Blood_Pressure and Weight_In_Lb) and dividing by 100.
- The main study documentation is stored in an HTML file and the study notes are stored as text.
- Dr. Joe E. Doe is the person responsible for the clinical study, and SAS is the program that analyzed the data.

All of the information can be described by a set of metadata objects. From this project inventory, we suspect a need for the following types of objects:

- table objects
- column objects
- some kind of person or responsible party object
- an object describing the DATA step
- an object that describes SAS
- documentation objects.

We know that we will also need a way to relate this study to other clinical studies owned by Dr. Joe E. Doe.

It is now time to learn about the SAS Metadata Model.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Selecting the Appropriate Metadata Types

The SAS Open Metadata Architecture provides a set of metadata types for representing metadata objects. Each type has a set of attributes, for example, the name of the object and its description, and a set of associations that describe the object's relationship to other objects. The type definitions are structured in a hierarchy, and each type inherits the attributes and associations of its supertype. This hierarchy is referred to as the SAS Metadata Model.

The SAS Metadata Model defines approximately 150 metadata types. To help you locate the correct metadata type to represent a particular item, the model is broken into submodels, each of which consists of a set of related types. In alphabetical order, these submodels are

Analysis
: contains the metadata types used to describe statistical transformations, multidimensional data sources, and OLAP information.

Authorization
: includes types that are used to define access controls. Metadata objects based on these types can be associated with metadata objects describing people, repositories, and application elements to control access both to the metadata and the data that the metadata describes.

Business Information
: contains the metadata types used to describe people, their responsibilities, and information about how to contact them, as well as business documentation and other descriptive information.

Foundation
: contains the basic metadata types of the model, from which all other types are derived, and some utility metadata types.

Grouping
: contains metadata types used to group metadata objects together in a particular context, as well as to construct a hierarchy of metadata objects.

Mining
: includes types that are used to store analytic information associated with data mining.

Property
: contains types used to describe prototypes of metadata objects, parameters for processes, and properties or options for SAS libraries, data sets, connections to servers, or software commands.

Relational
: contains the metadata types used to describe relational tables and other objects used in a relational database system, such as indexes, columns, keys, and schemas.

Resource
: contains the metadata types used to describe data resources such as files, directories, SAS libraries, SAS catalogs and catalog entries.

Software Deployment
: contains the metadata types used to describe software, servers, and connection information.

Transform
: contains the metadata types used to describe a transformation of data. This can be a logical to physical mapping, or a set of steps that transforms input data to a final result.

XML
: contains types that are used to describe XML constructs such as SXLE map definitions and XPath location paths.

The individual metadata types are described in the SAS Namespace Types section of the **SAS 9.1 Open Metadata Interface: Reference**.

Selecting the appropriate metadata types is a matter of comparing the types in each category to see which metadata types best meet your needs. The following paragraphs walk you through the thought processes for selecting the metadata types for our clinical studies project. We begin by selecting objects to represent the study tables.

Choosing Objects to Represent Tables

A table is a form of data set. Metadata types that describe data sets (which are a form of relational table) are included in the Relational submodel. The Relational submodel also includes a metadata type for columns, which are defined separately from data sets.

Several metadata types are used to define data sets, including PhysicalTable, QueryTable, RelationalTable, and WorkTable. To determine which type best suits your needs, check the descriptions and compare the attributes and associations of each type in the SAS Namespace Types documentation. In this example, we use the PhysicalTable metadata type to represent the three Study 1 tables. The PhysicalTable metadata type is described as a "materialized" data set that is stored in a database or a file system. The other types describe tables that result from a query or are otherwise transitory in nature. For this example, we use this icon to represent a Physical Table:

We use the Column metadata type to create our column objects, and represent these with this icon:

Choosing Objects to Represent Programs

Programs such as the DATA step program in our example are described by metadata types that are part of the Transform submodel. A simple SAS program that does not have a requirement on any previously run program or a requirement for any postprocessing is represented as a Transformation object. A Transformation object has associations to other objects that describe the input to the transformation and the output from the transformation. In Study 1, the data describing the patients and the doctor visits are the input for the transformation, and a new data set is the output of the transformation. We use the following icon to represent a Transformation object:

A Transformation object also has an association to another object that contains the source code for the program. The metadata types that describe textual information stored on the metadata server or text stored in a file are part of the Resource submodel. The Resource submodel also contains metadata types for file system directories, SAS catalogs, and SAS catalog entries. The actual text for the source code can be stored in a metadata object, or you can use a metadata object to identify the location of the source code. In this example, we use a TextStore object and save a copy of the source code (DATA step) on the SAS Metadata Server. The following icon represents a TextStore object:

Choosing Metadata Types to Represent Software

Metadata types that describe software and where the software is installed are part of the Software Deployment submodel. Installed and runnable software like SAS or a Java Virtual Machine are represented by the metadata type DeployedComponent and its subtypes. The computer where the software is installed is represented by the metadata type Machine. The software is a SAS server so we will use the ServerComponent subtype of the DeployedComponent metadata type to represent it. We use the following icons to represent the ServerComponent and Machine metadata types, respectively:

Choosing Metadata Types to Represent Documentation and People

The remaining information, concerning documentation and the responsible party, is represented by metadata types that are part of the Business Information submodel. The type Document contains information about the location of a document as a URI (Universal Resource Identifier). In this example, we want to store the URL for a document on a Web server. We use the following icon to represent the Document metadata type:

Frequently, the information that you want to save will not be represented by a single metadata type, but by a set of types. This is true for the person responsible for the analysis. The Business Information submodel contains types for identifying roles for a person, such as owner, user, or administrator. The submodel also contains types for contact information, such as e–mail address, phone number, and address. The types that we will use in this example are ResponsibleParty, Person, and Email, which we represent as follows:

Choosing Objects to Represent Groups

The last submodel we look at is the Grouping submodel. We use a Tree metadata type to impose a hierarchy on the objects comprising Study 1. We also use a Tree object named "Clinical Studies" under which we can group all of the clinical studies, for example, Study 1, Study 2, and so on. We use a Tree icon to represent the Tree metadata type:

Now that we know what kind of information to save and which metadata types to represent the objects, we need to get some administrative tasks out of the way.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Overview of Setting up a Server

This topic leads you through the steps for setting up a SAS Metadata Server under Windows NT. Refer to the **SAS 9.1 Metadata Server: Setup Guide** for instructions about setting up a SAS Metadata Server under a UNIX, Windows 2000, Windows XP or OS/390 operating environment. These instructions assume that the software described in the Before You Begin section is installed.

To step up a SAS Metadata Server, you must:

1. Create Directories for the Metadata Server, Repository Manager, and Repositories
2. Setting Directory and File Access Permissions
3. Setting System Access Permissions
4. Starting the SAS Metadata Server
5. Registering a Repository

Also see Stopping the SAS Metadata Server.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Creating Directories for the Metadata Server, Repository Manager, and Repositories

You will need to create directories for the SAS Metadata Server, the repository manager, and at least one repository. The default behavior of the metadata server is to look for the repository manager in the server directory in a directory named "rposmgr" and to assign it the libref RPOSMGR. If you wish to specify a different location for the repository manager or to assign it a different libref, see the steps for creating an omaconfig.xml file in the **SAS 9.1 Metadata Server: Setup Guide**. The following example creates directories respecting the default server configuration.

1. On the C: drive, select **New−>Folder** from the File menu to create a new folder.
2. Name the currently highlighted folder `omaserver`.
3. Double−click the folder to open it.
4. Within the `omaserver` folder, create two additional folders.
5. Name the first folder `rposmgr`.
6. Name the second folder `myrepos`.

The rposmgr directory is created within the server directory because it is required to be there. The repository directory is created within the server directory for convenience. The directories for a repository can be located anywhere as long as the SAS Metadata Server can access them. By creating the repository directory in the server directory, the repository can share the directory and file access permissions we set for the server.

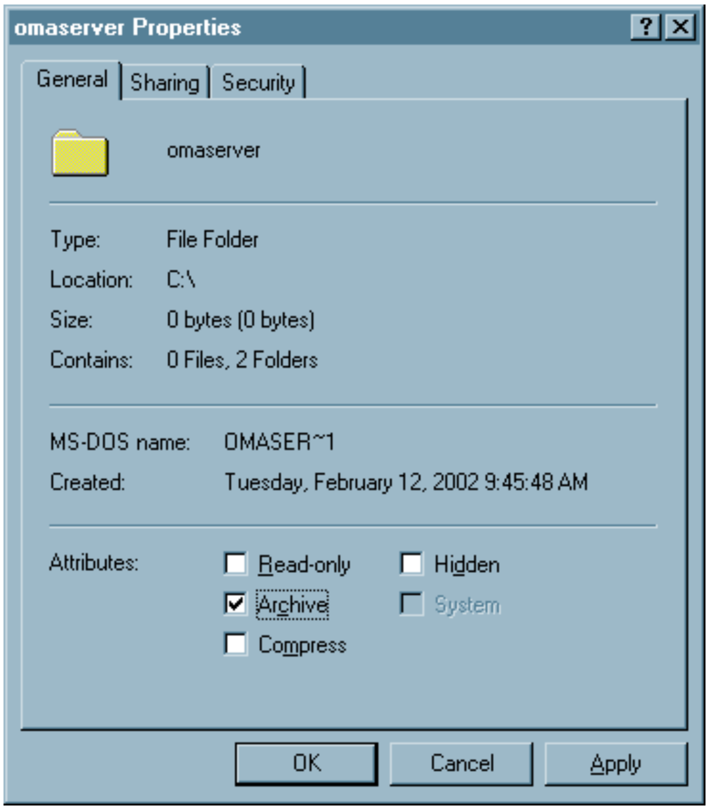
Previous Page | Next Page | Top of Page

Setting Directory and File Access Permissions

To set up or maintain a repository manager and repository, you must have full access to the repository manager and repository directories. Because we have nested the `rposmgr` and `myrepos` directories within the `omaserver` directory, we can set the necessary permissions one time for the server directory.

To set the permissions:

1. On the C: drive, select the `omaserver` folder icon. Press and hold down the right mouse button to display the File pop–up menu; then select **Properties**. The omaserver Properties window displays for you to enter the appropriate information.
2. On the General tab, select the **Archive** check box.



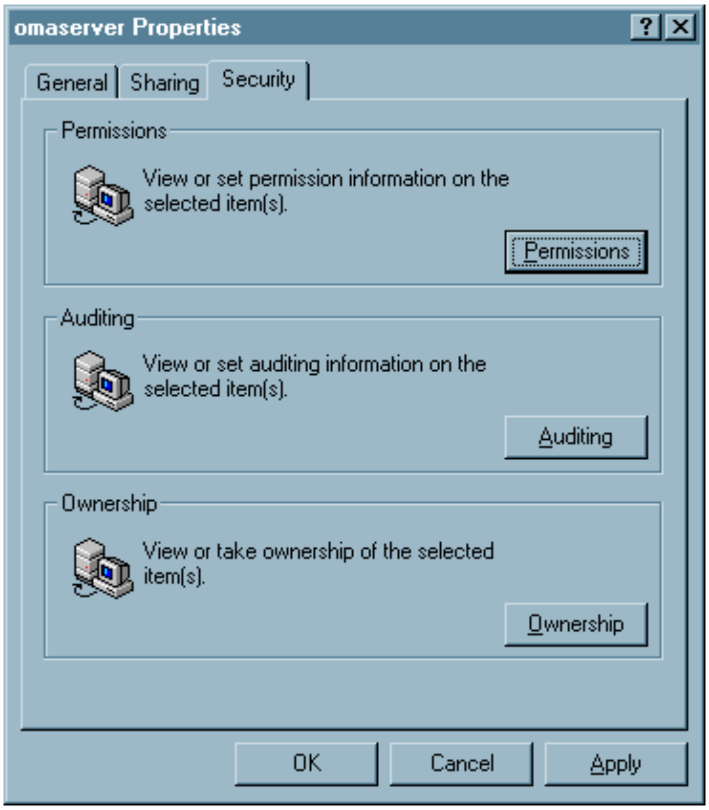
omaserver

Type: File Folder
Location: C:\
Size: 0 bytes (0 bytes)
Contains: 0 Files, 2 Folders

MS-DOS name: OMASER~1
Created: Tuesday, February 12, 2002 9:45:48 AM

Attributes: [ ] Read-only [✓ Archive] [ ] Compress [ ] Hidden [ ] System

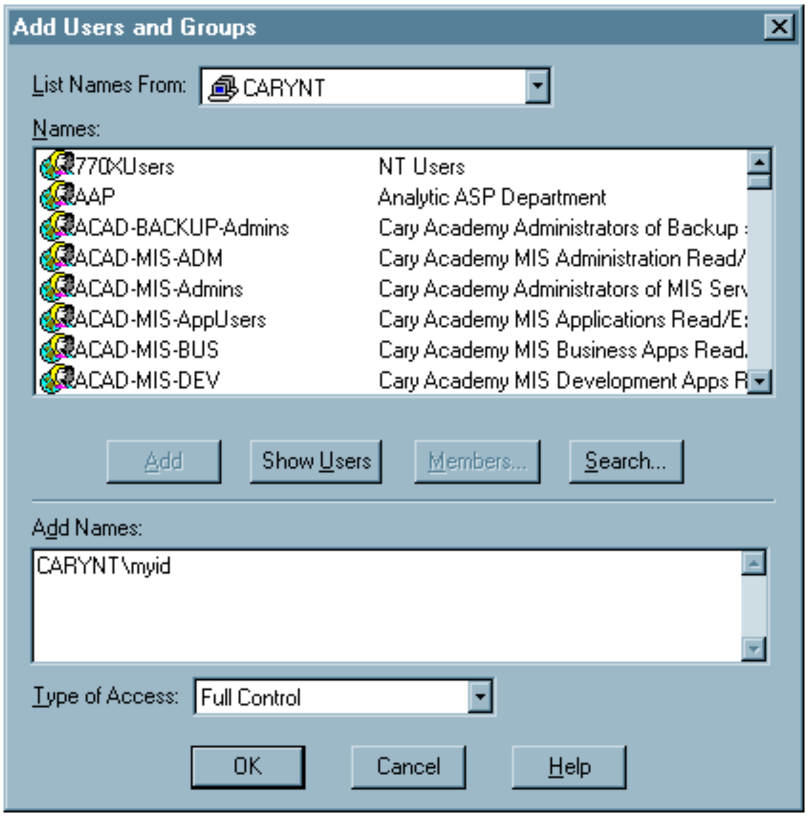
3. On the Security tab, click **Permissions**.


omaserver Properties
General
Sharing
Security
Permissions
View or set permission information on the selected item(s).
Permissions
Auditing
View or set auditing information on the selected item(s).
Auditing
Ownership
View or take ownership of the selected item(s).
Ownership
OK
Cancel
Apply

4. In the Directory Permissions window, select and remove the **Everyone** identity from the **Name** field and click the **Add** button.
5. In the Add Users and Groups window:

   a. Type your ID in the **Add Names** field in the form:

   ```
   domain\userid
   ```
   b. Specify **Full Control** as the type of access.



List Names From: CARYNT

Names:

770XUsers	NT Users
AAP	Analytic ASP Department
ACAD-BACKUP-Admins	Cary Academy Administrators of Backup :
ACAD-MIS-ADM	Cary Academy MIS Administration Read/
ACAD-MIS-Admins	Cary Academy Administrators of MIS Serv
ACAD-MIS-AppUsers	Cary Academy MIS Applications Read/E:
ACAD-MIS-BUS	Cary Academy MIS Business Apps Read.
ACAD-MIS-DEV	Cary Academy MIS Development Apps R

Add Names: CARYNT\myid

Type of Access: Full Control

c. Grant the appropriate administrator or user ID access to allow the company's backup procedures to run correctly.
d. Click **OK** to close the window and return to the Directory Permission window.

6. In the Directory Permissions window, select the **Replace permissions on subdirectories** check box, then click **OK** to close the window.
7. A confirmation dialog will ask you to verify that you want to change the permissions for the subdirectories. Click **Yes**.
8. In the omaserver Properties window, click **Apply** to save your changes, and then **OK** to close the window.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

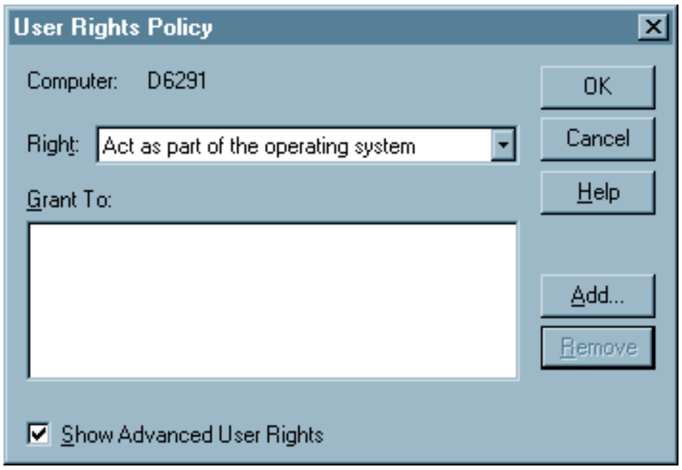
Setting System Access Permissions

As the person who invokes the metadata server, you must have specific user rights on the server host to start the SAS Metadata Server. In addition, other users and applications must be given the right to log on while the server is running.

To set these permissions:

1. From the Windows NT Start menu, select **Programs–>Administrative Tools–>User Manager**. This opens the User Manager window, from which you can make selections.
2. From the Policies menu, select **User Rights**.
3. In the User Rights Policy window:

    a. Select the **Show Advanced User Rights** check box.



Computer: D6291
Right: Act as part of the operating system
Grant To:
☑ Show Advanced User Rights

    b. From the **Right** drop–down list, select the following user rights for the server invoker:

        ◊ **Act as part of the operating system**
        ◊ **Increase quotas**
        ◊ **Replace a process level token**
        ◊ **Log on as batch job**

    Selecting a user right displays the current list of users in the **Grant To** box. Use the **Add** and **Remove** buttons to add and remove names for each right.

    c. To the **Log on as batch job** user right, also add the identity **Everyone**. This enables other users and applications to access the metadata server.

4. Click **OK** to close the User Rights Policy window and the User Manager.
5. Restart the server host so that the updates can take effect.

You are now ready to start the server.

Previous Page | Next Page | Top of Page

Starting the SAS Metadata Server

The SAS Metadata Server must have exclusive use of the repository manager. That is why the default server settings look for the rposmgr directory in the server directory. You are also required to start the metadata server in the server directory. To start the metadata server, store the following command in a startsrv.bat file in the server directory, then execute the **startsrv** command.

```
"where_your_SAS_is_installed\sas.exe" -log "C:\sasoma\logs\sasoma.log"
-logparm "write=immediate" -linesize max -pagesize max -nosplash -noterminal
-memsize max -objectserver -objectserverparms "protocol=bridge port=XXXX
classfactory=2887E7D7-4780-11D4-879F-00C04F38F0DB"
```

where

- **where_your_SAS_is_installed** is the path to the directory where SAS 9.1 has been installed (typically C:\Program Files\SAS\SAS System\9.1)
- the string XXXX in the option **port=XXXX** is an unused port number where the server will listen for requests. XXXX is a four−digit number from 0−9999.

Click here to download a startsrv.bat file you can edit.

You are now ready to register a repository.

Previous Page | Next Page | Top of Page

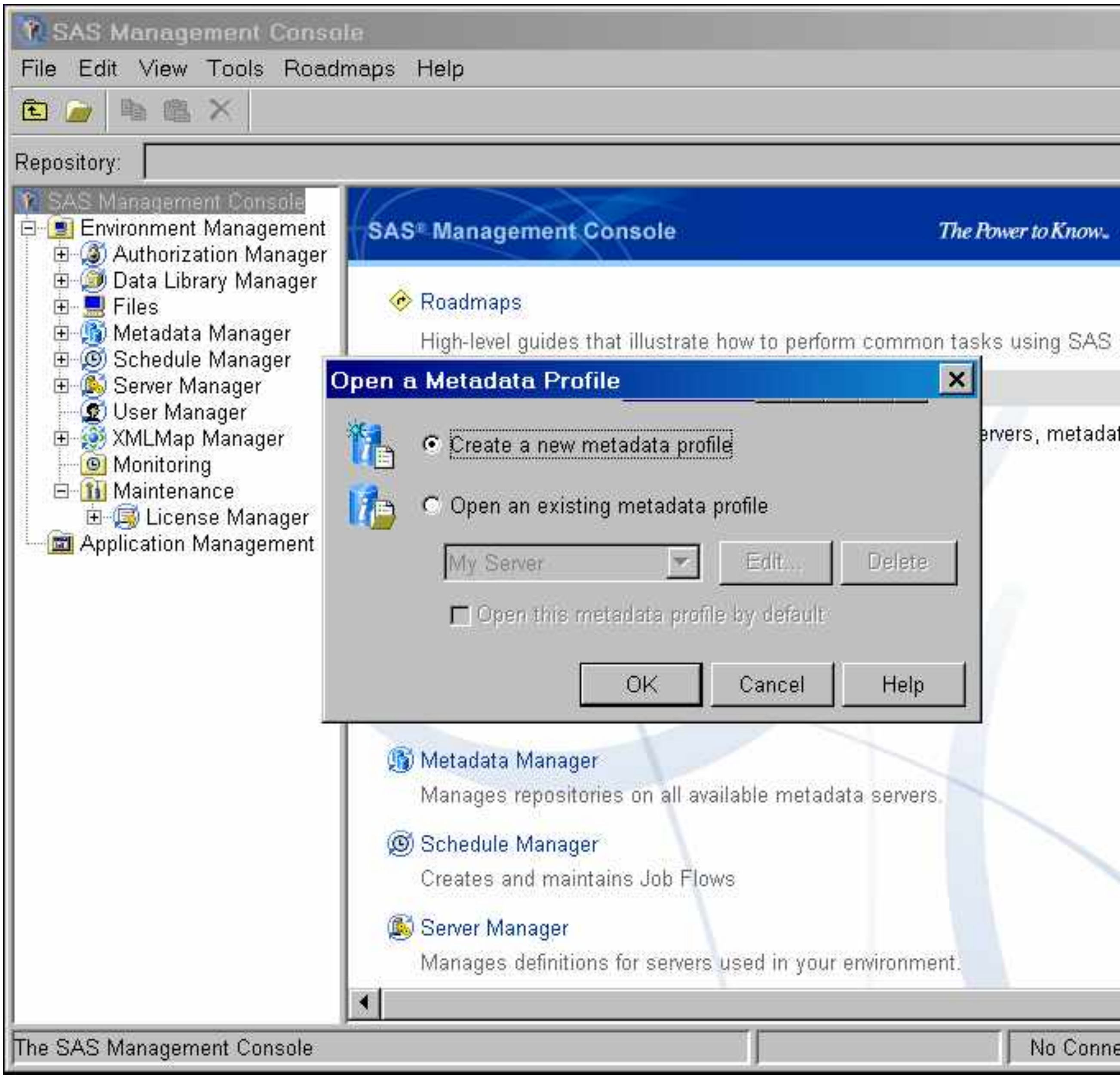
Getting Started with the SAS 9.1 Open Metadata Interface

Registering a Repository

Before you can manage a repository, it must be identified to the repository manager in a registration process. The recommended method for registering a repository is using SAS Management Console.

To register a repository:

1. Select **Programs–>SAS–>SAS Management Console** from the Windows NT Start menu. The software displays a window similar to the following:



High-level guides that illustrate how to perform common tasks using SAS

[✓ Create a new metadata profile]

Open an existing metadata profile

My Server

Open this metadata profile by default

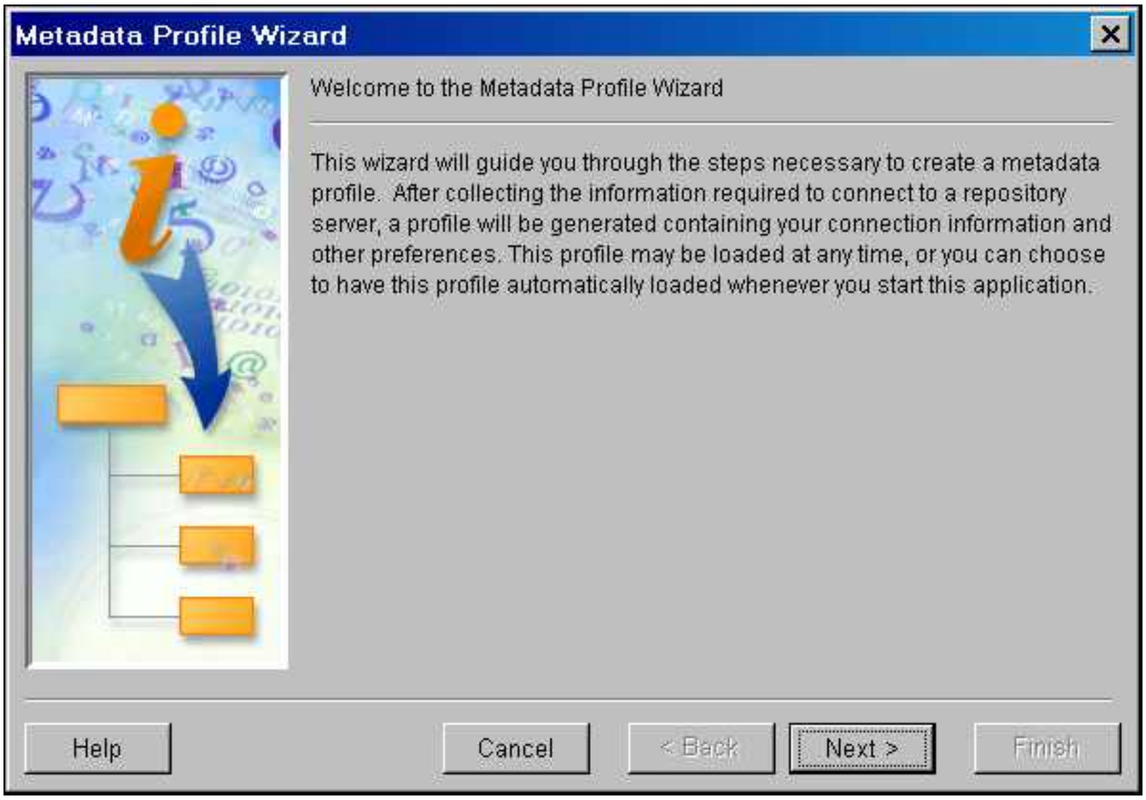
ervers, metadat

Manages repositories on all available metadata servers.

Creates and maintains Job Flows

Manages definitions for servers used in your environment.

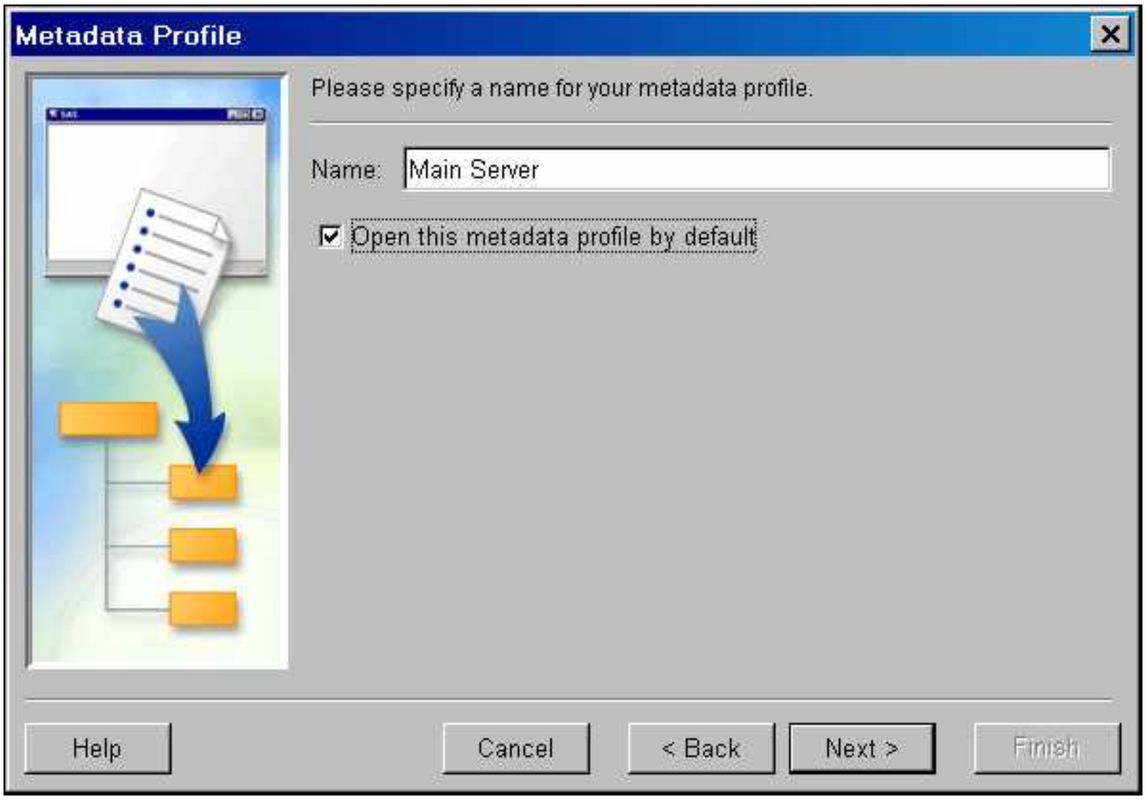
2. Before you can use SAS Management Console, you must connect to a running metadata server. In the Open a Metadata Profile window, select **Create a new metadata profile**, and click **OK**. The software opens the Metadata Profile Wizard to guide you through the steps of entering server connection properties.


Metadata Profile Wizard

Welcome to the Metadata Profile Wizard

This wizard will guide you through the steps necessary to create a metadata profile. After collecting the information required to connect to a repository server, a profile will be generated containing your connection information and other preferences. This profile may be loaded at any time, or you can choose to have this profile automatically loaded whenever you start this application.

3. Click **Next** to begin the wizard.
4. The Metadata Profile window of the wizard prompts you to enter a name to identify the server. This can be any name up to 200 characters.


Metadata Profile

Please specify a name for your metadata profile.

Name: Main Server

☑ Open this metadata profile by default

Note: If you do not select the **Open this metadata profile by default** check box, you will be prompted for server connection information each time you open SAS Management Console. Click **Next** to continue.

5. The Connection Information window collects server connection properties. In the appropriate field, enter

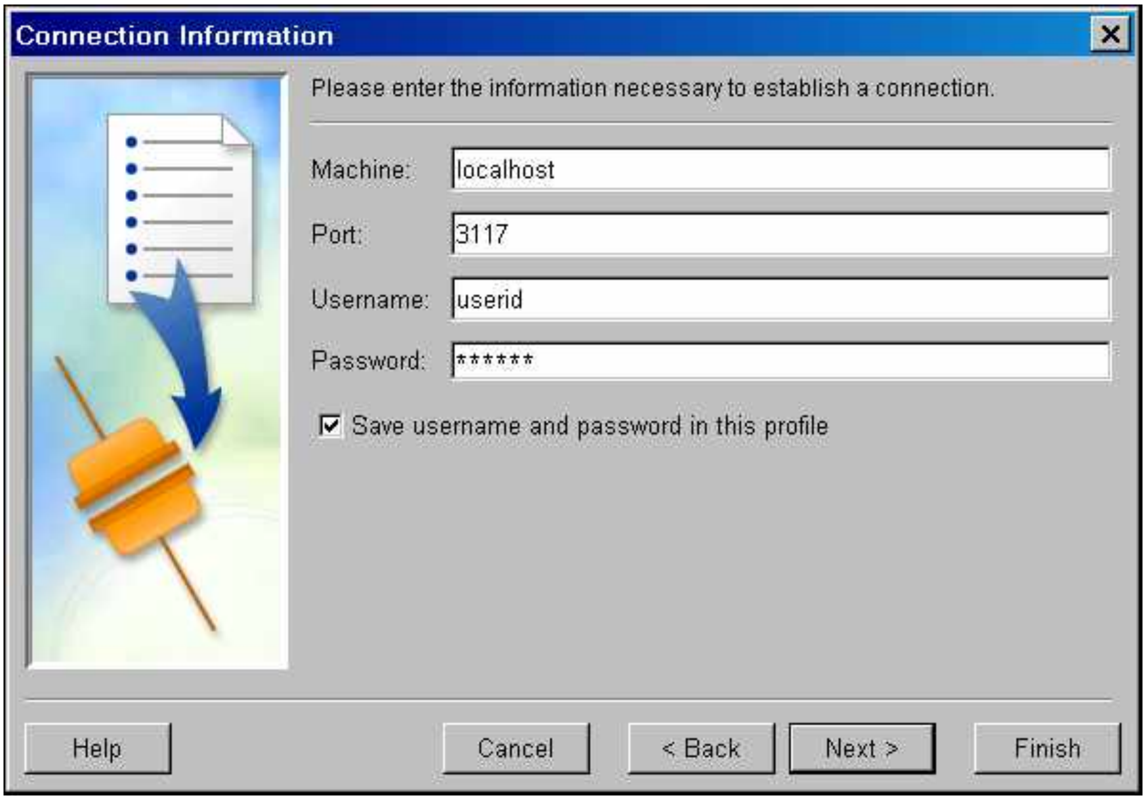
   Machine
   : is the host name or Internet Protocol address of the computer on which you started the metadata server.

   Port
   : is the port number passed in the start server command, to which the server will listen for requests.

   Username
   : specifies a valid user ID on the computer hosting the metadata server.

   Password
   : specifies the password associated with the user ID.

   The following graphic shows a typical set of connection properties.



Connection Information

Please enter the information necessary to establish a connection.

Machine: localhost

Port: 3117

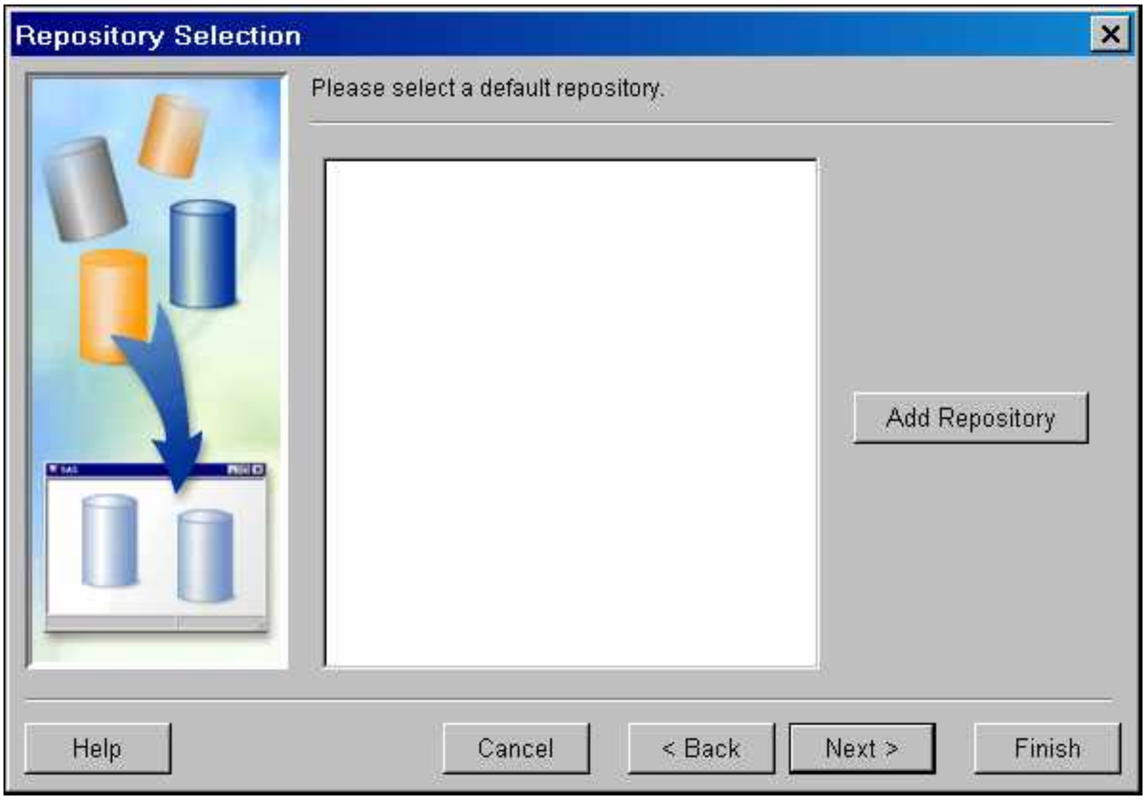
Username: userid

Password: ******

☑ Save username and password in this profile

   Click **Next** to continue.

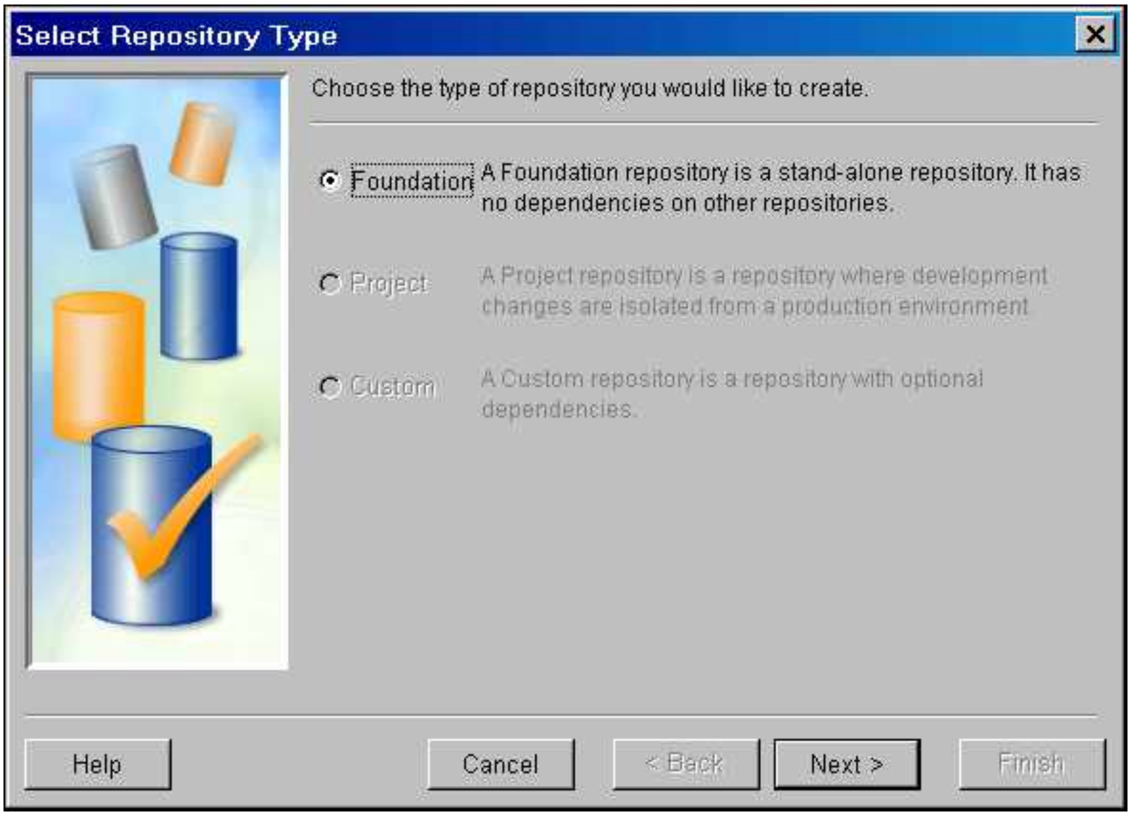
6. The Repository Selection window allows you to enter a default repository. Select **Add Repository** to define a repository.



7. The software opens the New Repository Wizard, which enables you to create repositories of three types:

   - Foundation is a "base" or "master" repository on which other repositories rely for shared information such as metadata identities and default authorization settings.
   - Project is a repository that is used to isolate changes from a production environment. A metadata developer uses a project repository as a playpen for making changes.
   - Custom is a repository that is dependent on a foundation repository or another custom repository.

   SAS Management Console requires that you create a foundation repository as the first repository on a metadata server.


Select Repository Type

Choose the type of repository you would like to create.

[✓ Foundation] A Foundation repository is a stand-alone repository. It has no dependencies on other repositories.

Project A Project repository is a repository where development changes are isolated from a production environment.

Custom A Custom repository is a repository with optional dependencies.

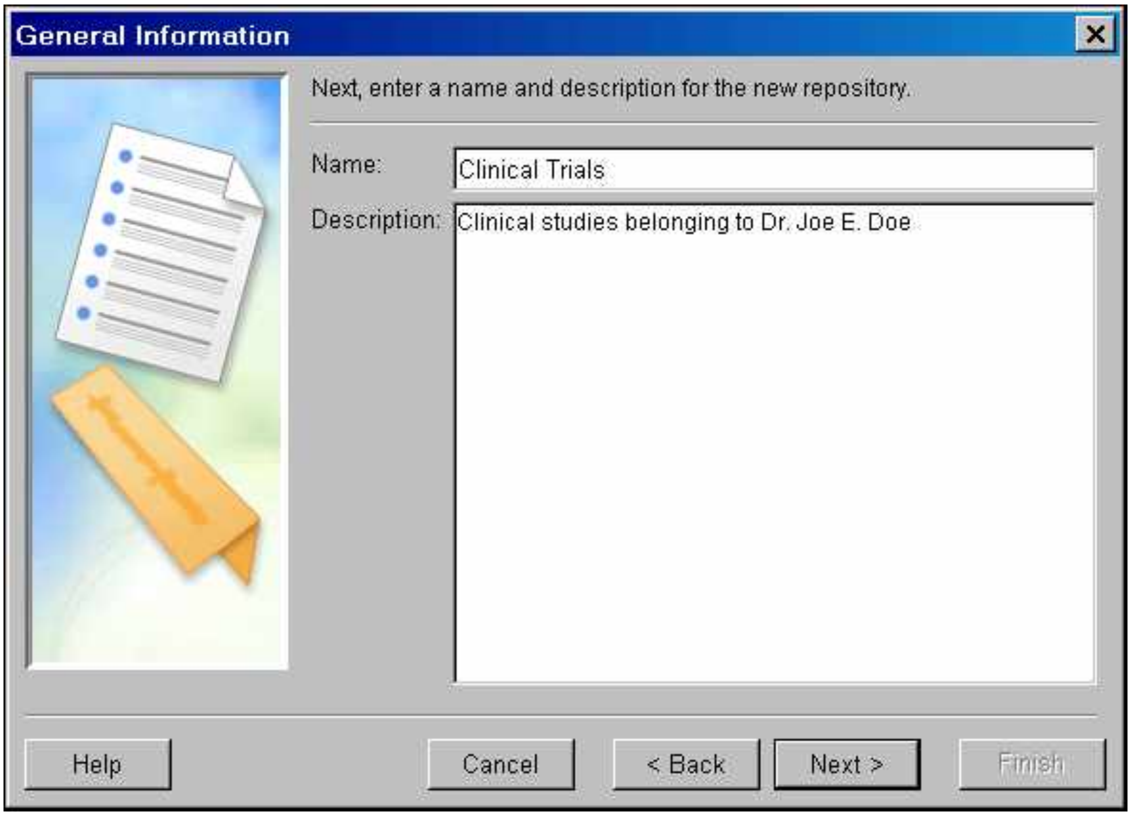
Click **Next** to continue.

8. In the General Information window, enter a name and a description for the repository:

Name
is a unique name for the repository.

Description
is an optional descriptive phrase for the repository.

The following graphic shows values we might enter for our clinical trials repository.



Next, enter a name and description for the new repository.

Name: Clinical Trials

Description: Clinical studies belonging to Dr. Joe E. Doe

Click **Next** to continue.

9. In the Definition of Data Source window, enter the data source information for the repository:

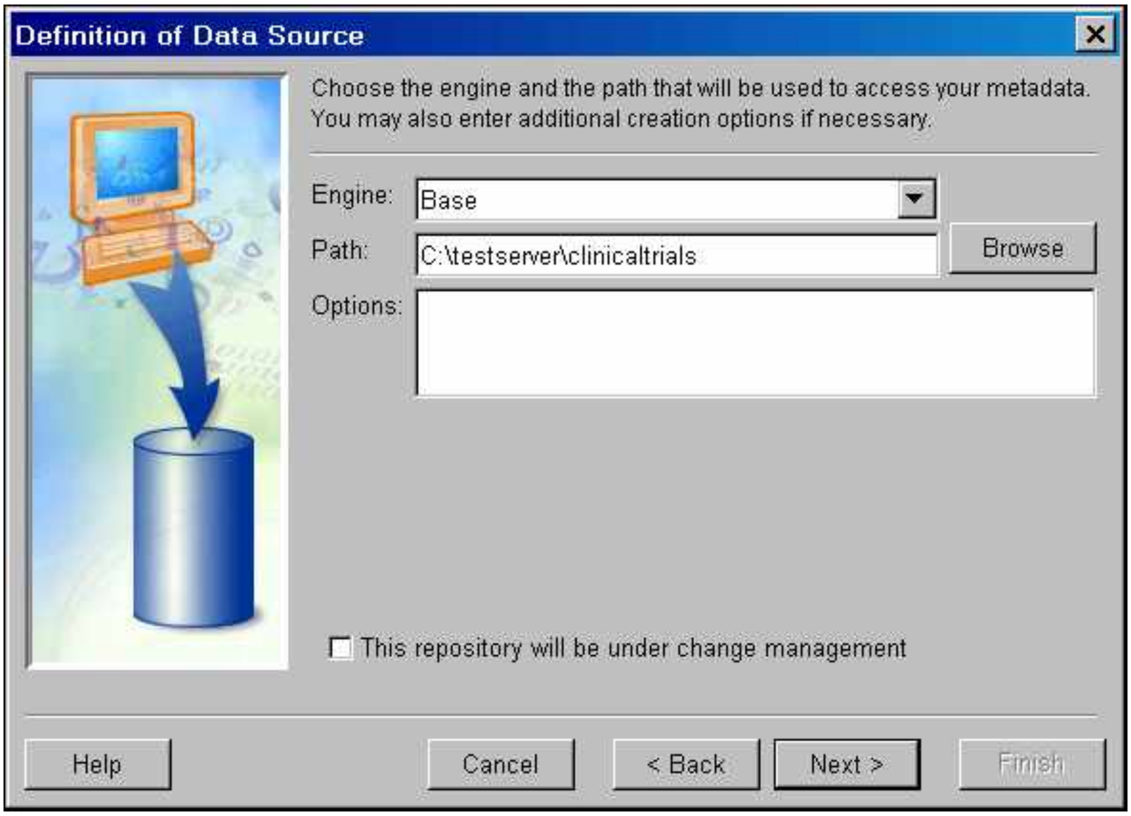
Engine
: is the engine used to access this repository. Valid values are Base, DB2, and Oracle. **Base**, representing the SAS base engine, is the default engine if one is not specified.

Path
: is the physical path to the repository directory. This is a required parameter when the Base engine is selected. When creating a repository that uses the DB2 or Oracle engines, leave this field blank. The DB2 and Oracle engines get their path information from the Options field.

Options
: specifies LIBNAME and engine–connection options required to create a repository on an external DBMS. This exercise describes how to create a repository using the SAS Base engine. See the SAS Management Console help for information about the options required to create a repository on DB2 or Oracle.

This repository will be under change management
: When checked, specifies that metadata in the repository will be checked–out to and updated in a project repository. When left unchecked, indicates that metadata will be updated in place. For this exercise, we will leave this option unchecked, so we can directly update metadata in the repository.

The following graphic shows the values for our clinical trials repository.


Definition of Data Source

Choose the engine and the path that will be used to access your metadata. You may also enter additional creation options if necessary.

Engine: Base

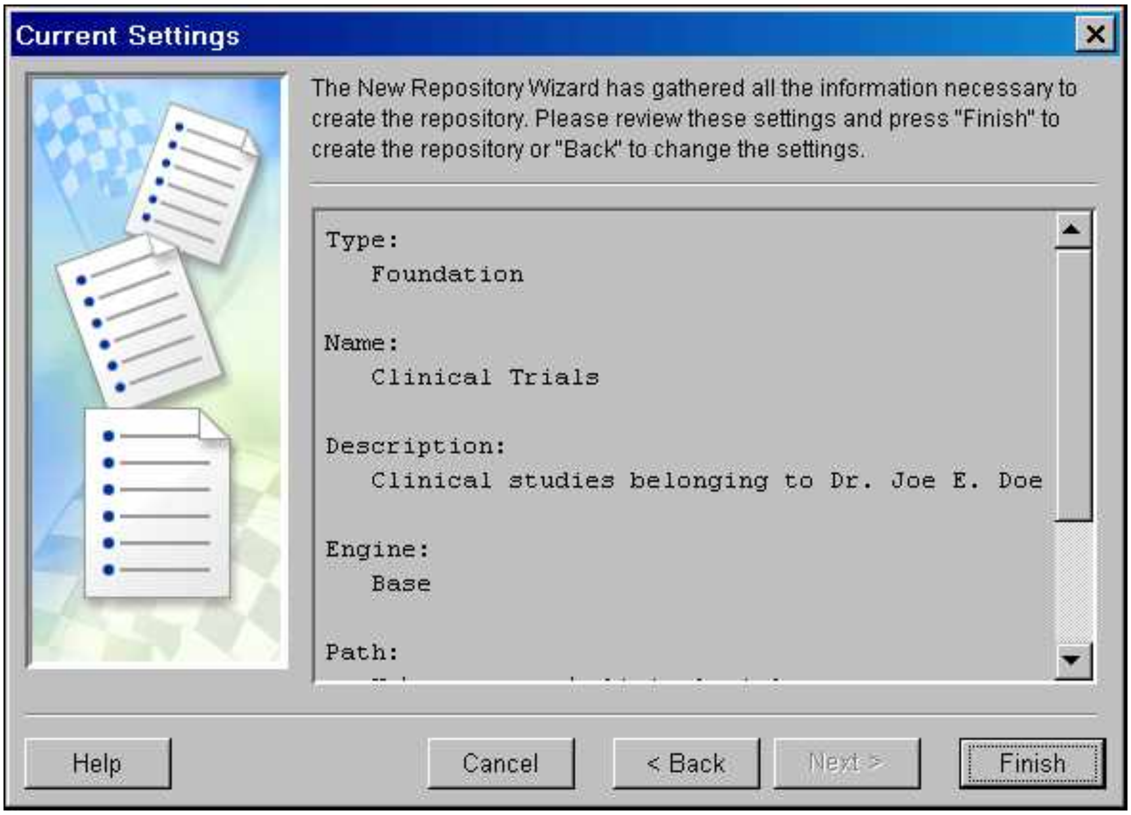
Path: C:\testserver\clinicaltrials

Options:

☐ This repository will be under change management

Click **Next** to continue.

10. The wizard displays a Current Settings window for you to review the repository properties.

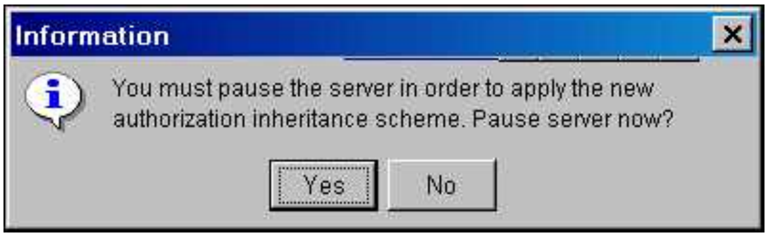

Current Settings

The New Repository Wizard has gathered all the information necessary to create the repository. Please review these settings and press "Finish" to create the repository or "Back" to change the settings.

```
Type:
   Foundation

Name:
   Clinical Trials

Description:
   Clinical studies belonging to Dr. Joe E. Doe

Engine:
   Base

Path:
```

Click **Finish** to save the settings and create the repository.

11. The software displays an informational message as it initializes the repository. Foundation repositories have a set of default resource templates and authorization metadata created in them. When the initialization is complete, it displays a window to inform you the initialization was successful. Click **OK** to continue.
12. The software will display a window informing you that you must pause the server in order to apply the authorization settings.



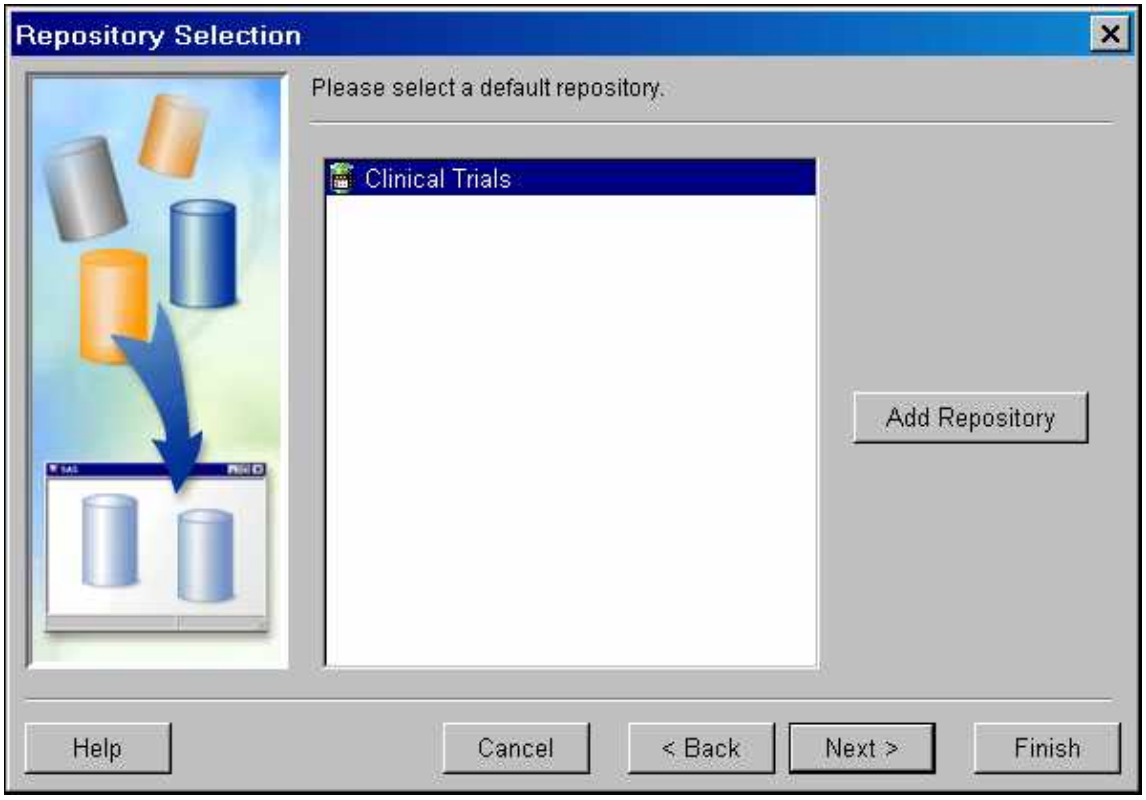
Information

You must pause the server in order to apply the new authorization inheritance scheme. Pause server now?

Click **Yes**.

Note: If you click **No**, the directory and system access controls you defined in the previous steps will be the only security enforced on the repository.

13. The software will return you to the Repository Selection window. Select your new repository and click **Finish** to store the new metadata profile and repository information.



Repository Selection

Please select a default repository.

Clinical Trials

You are now ready to begin storing metadata.

| Next Page |

Previous Page Top of Page

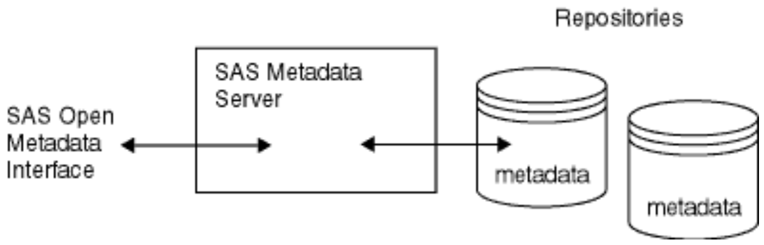
Getting Started with the SAS 9.1 Open Metadata Interface

Overview of Writing Clients

You use the SAS Metadata Server by writing SAS Open Metadata Interface clients in Java, Visual Basic, or C++, or by using SAS−provided SAS Open Metadata Interface clients such as PROC METADATA, PROC METAOPERATE, and SAS Management Console. A SAS Open Metadata Interface client is a program that connects to the SAS Metadata Server and issues SAS Open Metadata Interface method calls to create, query, or update metadata in SAS metadata repositories or to administer SAS metadata repositories or the metadata server.

- PROC METADATA enables you to issue XML−formatted SAS Open Metadata Interface method calls to create and query all types of metadata objects from the SAS Program Editor.
- PROC METAOPERATE enables you to pause, resume, and refresh repositories to temporarily change them to another state or to recover memory for the server from the Program Editor. PROC METAOPERATE also allows you to stop and get the status of the metadata server from the Program Editor.
- SAS Management Console provides a graphical interface for registering SAS metadata repositories and for creating metadata describing global resources, such as servers, users, library connections, and metadata and data access controls.

All SAS Open Metadata Interface clients must connect to a running metadata server before they can issue method calls.



This guide describes the method calls necessary to read and write metadata. The XML method calls provided as examples can be submitted to the metadata server via PROC METADATA or they can be included in a Java, Visual Basic, or C++ client. For more information about PROC METADATA, see the METADATA Procedure in the **SAS 9.1 Open Metadata Interface: Reference**. For more information about writing and using SAS Open Metadata Interface clients, including server connection requirements, see "Client Requirements" in the **SAS 9.1 Open Metadata Interface: Reference**.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Issuing a Method Call

Each metadata–related method takes a set of parameters that are used to drive the behavior of the method. SAS Open Metadata Interface clients must define object variables for these parameters. The SAS Open Metadata Interface supports two interfaces for issuing metadata–related method calls. The examples in this guide format method requests for the *inMetadata* parameter of the DoRequest method. For more information, see the DoRequest method in the **SAS 9.1 Open Metadata Interface: Reference**.

The DoRequest method allows you to code metadata–related methods and all of their parameters in XML and pass the XML method request to the metadata server as a generic input string. The server parses the XML string, issues the method call, and returns output in a generic *outMetadata* parameter. The syntax of the DoRequest method is as follows:

```
rc = DoRequest(inMetadata,outMetadata);
```

The format of the XML input string accepted in the *inMetadata* parameter is:

```
<MethodName>
  <Parameter1>Value</Parameter1>
  <Parameter2>Value</Parameter2>
  <Parametern>Value</Parametern>
  ...
</MethodName>
```

where <MethodName> is the name of a metadata–related method and <Parameter*n*> represent the parameters required by a given method. Most metadata–related methods require one or more of the following parameters:

<Metadata></Metadata>
: passes a metadata property string defining the metadata properties to be added, deleted, retrieved, or updated.

<NS></NS>
: specifies the namespace to use as the context for the request. Metadata–related calls are issued in the SAS namespace.

<Flags></Flags>
: specifies additional commands for processing the call, specified in numeric form. Multiple flags are specified by adding their numbers together.

<Options></Options>
: specifies additional parameters for processing the call.

<Reposid></Reposid>
: passes a repository identifier.

<Type></Type>
: passes a metadata type.

<Supertype></Supertype>
: passes a metadata type for which you wish to list subtypes, if they exist.

Metadata–related methods are part of the SAS Open Metadata Interface IOMI class. See the method descriptions in IOMI Class in the **SAS 9.1 Open Metadata Interface: Reference** for information about the parameters used by specific methods.

An example of an AddMetadata method call that is formatted in XML is shown below:

```
<AddMetadata>
  <Metadata>
  <Column Id="" Name="New Column">
    <Table>
     <PhysicalTable Objref="A2345678.A2000001" Name="Test Table"/>
    </Table>
  </Column>
  </Metadata>
  <Reposid>A0000001.A2345678</Reposid>
  <NS>SAS</NS>
  <!--OMI_TRUSTED_CLIENT flag-->
  <Flags>268435456</Flags>
  <Options/>
</AddMetadata>
```

You can pass two or more method requests in an input XML string by enclosing their elements between <Multiple_Requests> elements as follows:

```
<Multiple_Requests>
<AddMetadata>
  <Metadata>
  <Column Id="" Name="New Column">
    <Table>
     <PhysicalTable Objref="A2345678.A2000001" Name="Test Table"/>
    </Table>
  </Column>
  </Metadata>
  <Reposid>A0000001.A2345678</Reposid>
  <NS>SAS</NS>
  <!--OMI_TRUSTED_CLIENT flag-->
  <Flags>268435456</Flags>
  <Options/>
</AddMetadata>

<GetMetadata>
   <Metadata>
     <PhysicalTable Id="A2345678.A2000001" Name="TestTable">
        <Columns/>
     </PhysicalTable>
   </Metadata>
   <MS>SAS</NS>
   <Flags>0</Flags>
   <Options/>
</GetMetadata>
</Multiple_Requests>
```

In the preceding example, the first XML method request adds a column to PhysicalTable A2345678.A2000001 and the second gets the columns defined for the table to verify the column's addition.

Issuing a Method Call via PROC METADATA

PROC METADATA accepts method calls that are formatted for the *inMetadata* parameter of the DoRequest method. To submit the preceding AddMetadata method call to the metadata server using PROC METADATA, issue the following statements in the Program Editor:

```
PROC METADATA

    SERVER="host_name_of_computer_running_the_server"
    PORT=port_number
    USERID="userid"
    PASSWORD="password"
    PROTOCOL=BRIDGE

    IN="<AddMetadata>
    <Metadata>
    <Column Id="" Name="New Column">
      <Table>
        <PhysicalTable Objref="A2345678.A2000001" Name="Test Table"/>
      </Table>
    </Column>
    </Metadata>
    <Reposid>A0000001.A2345678</Reposid>
    <NS>SAS</NS>
    <!--OMI_TRUSTED_CLIENT flag-->
    <Flags>268435456</Flags>
    <Options/>
  </AddMetadata>";

RUN;
```

The first five statements supply server connection properties. The IN statement passes the XML–formatted method request to the server.

For more information, see the METADATA Procedure in the **SAS 9.1 Open Metadata Interface: Reference**.

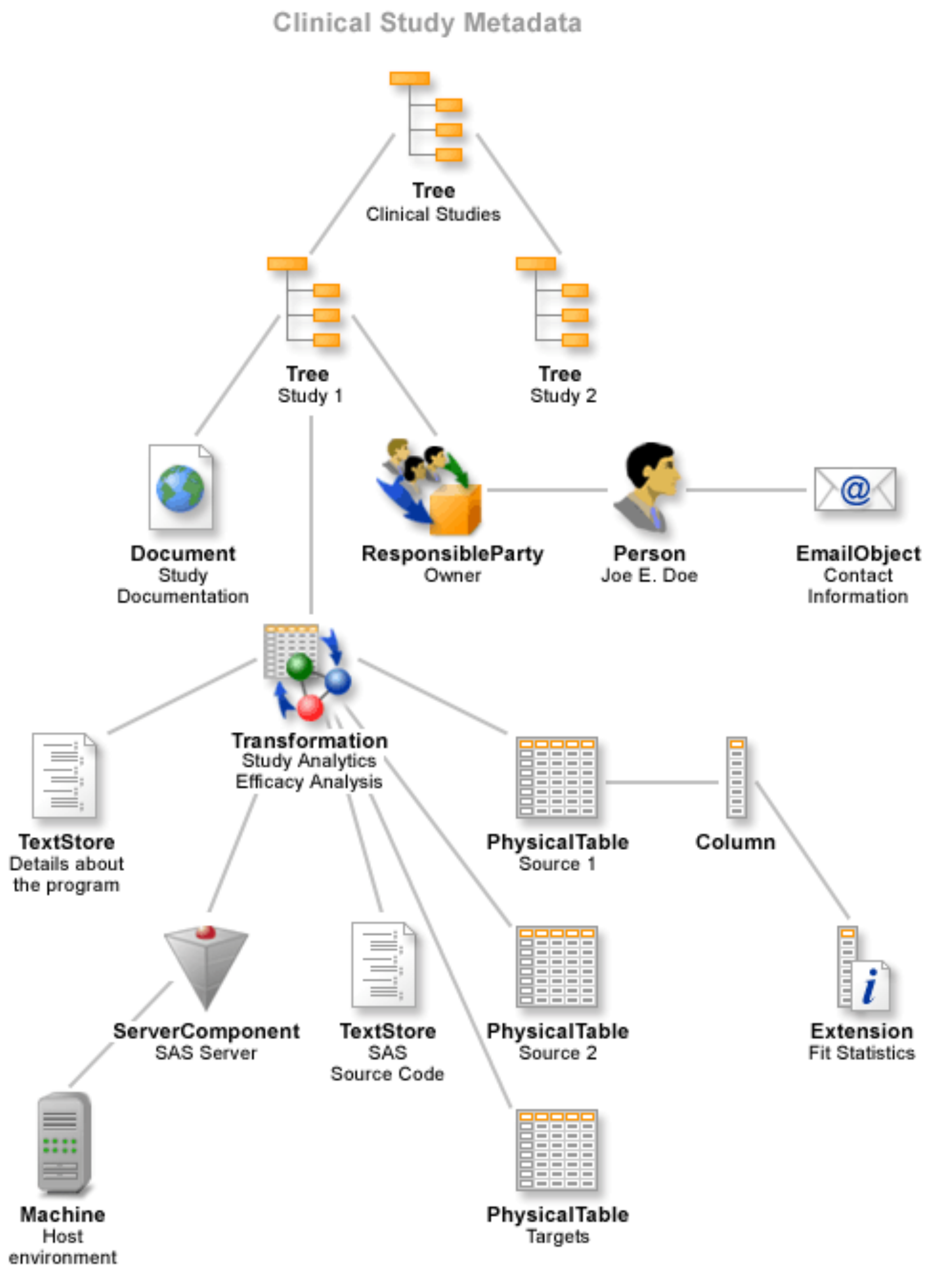
Previous Page | Next Page | Top of Page

Overview of Adding Metadata Objects

This section describes the SAS Open Metadata Interface method calls necessary to add objects representing the metadata types identified in Selecting the Appropriate Metadata Types to the myrepos repository created in Overview of Setting Up a Server. As a refresher, we identified the need to define

- PhysicalTable objects for the Study 1 input tables and output table
- Column objects describing the columns in each table
- Person, ResponsibleParty, and Email objects describing the study owner (Dr. Joe E. Doe)
- Transformation and text objects to represent and store the DATA step program that processed the input and output tables and notes about the transformation
- a Documentation object
- ServerComponent and Machine objects describing SAS and its host environment
- Tree objects for grouping the Study 1 objects into one project, and the Study 1 project together with other projects owned by Dr. Joe E. Doe.

These metadata types will enable us to build a repository containing the following objects and relationships:



Concepts

At this point, it is important to introduce some SAS Metadata Model concepts and terms:

- The relationship between two metadata types (represented as a branch in this diagram) is referred to as an **association**.
- The associations supported for each metadata type are predefined by the SAS Metadata Model and they have a name (not shown in this diagram), which is referred to as the **association name.**
- Every association in the SAS Metadata Model is a two−way association. That is, the association has a different association name depending on which of the metadata types is used to refer to the association. For example, when defining an association between the Tree metadata type and the Document metadata type, the association will have a Documents association name (indicating that the

document is one of several possible documents that is associated with a particular Tree). When defining an association between the Document metadata type and a Tree, the association will have a Trees association name (Tree is one of several possible Trees to which the document has an association). The partner metadata type in the relationship is referred to as the **association subelement**.
- A **cardinality** is assigned to each association name. The cardinality indicates the number of objects that are supported in the association and whether the association is required or optional. The cardinality of each association is described in the metadata type documentation.

Tasks

Metadata objects are added to SAS metadata repositories by using the AddMetadata method. We will create the objects and associations for the following metadata types:

1. Creating a Clinical Studies Tree
2. Creating a Study 2 Tree
3. Creating a Study 1 Tree and Table−related Objects
4. Creating TransformationActivity, TextStore, and ServerComponent Objects

The examples represent *one* way the metadata can be added. This is by no means the only way it can be done.

Before you can add any metadata objects, however, you must know the ID of the repository where you are adding the objects. You must also know how to write a metadata property string.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Determining the Repository ID

Every object in a SAS metadata repository has a unique identifier. A **repository identifier** is assigned to the repository when it is registered in the SAS Metadata Server repository manager. The SAS Metadata Server assigns an **object instance identifier** of the form REPOSID.INSTANCEID to each metadata object upon the successful completion of an AddMetadata call.

When you issue an AddMetadata call, you must specify a repository identifier for the repository where you write the object. Therefore, you must first determine the repository identifier of the target repository. This is done with the SAS Open Metadata Interface GetRepositories method. An XML input string containing a sample GetRepositories call follows:

```
<GetRepositories>
  <Repositories/>
  <Flags>0</Flags>
  <Options/>
</GetRepositories>
```

The output received from the server is as follows:

```
<GetRepositories>
  <Repositories>
  <Repository Name="testrepos" Desc="clinical trials metadata"
   DefaultNS="SAS" Id="A0000001.A57A0KGS"/>
  </Repositories>
  <Flags>0</Flags>
  <Options/>
</GetRepositories>
```

It shows that one repository (A57A0KGS) has been defined in the current repository manager (A0000001). A0000001.A57A0KGS is the value we will use to identify the Clinical Studies repository in our AddMetadata calls.

Now we can learn how to write a metadata property string.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Writing a Metadata Property String

Methods that read or write a metadata object must pass a string of properties that describe that object to the SAS Metadata Server. This property string is passed to the server in the <Metadata> element (inMetadata parameter) of the method call.

A metadata object is described by

- its metadata type
- attributes that are specific to the metadata object, such as its ID, name, description, and other characteristics
- its associations with other metadata objects.

The SAS Open Metadata Interface supports the following XML elements for defining a metadata property string:

Metadata type

identifies the metadata type that you want to read or write, within angle brackets. This example shows the XML element representing the PhysicalTable metadata type.

```
<PhysicalTable></PhysicalTable>
```

A shorthand method of specifying this tag set is:

```
<PhysicalTable/>
```

Metadata type attributes

specify attributes of the metadata type as XML attributes (within the angle brackets of the metadata type). This example specifies the PhysicalTable metadata object that has NE Sales in the Name attribute.

```
<PhysicalTable Name="NE Sales"/>
```

Metadata type association name and association subelement elements

describe the relationship between the metadata type named in the main XML element and another metadata type as nested XML elements, as follows:

```
<PhysicalTable Name="NE Sales"/>
   <Columns>
      <Column/>
   </Columns>
</PhysicalTable>
```

The first nested element, Columns, is the **association name**. The association name is a label that describes the relationship between the main element and the subelement. The second nested element, Column, is the **association subelement**. The association subelement identifies the partner metadata object in the relationship. The nested elements in the example specify that the main metadata object, PhysicalTable, has a Columns association to an object of metadata type Column.

Note: In order to meet XML parsing rules, the metadata type, attribute, and association element and subelement names that you specify in the metadata property string must exactly match those published in the metadata type documentation.

We can now begin adding metadata objects to our example repository.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Creating the Clinical Studies Tree

The following XML string creates the Tree object to represent the top–level node of the clinical study information. It is an example of a simple AddMetadata call: the call adds a single object containing the specified attributes to the repository identified in the *Reposid* parameter.

```
<AddMetadata>
  <Metadata>
    <Tree
      Desc="Top level node for clinical study information."
      Name="Clinical Studies"
      TreeType="Clinical Study">
    </Tree>
   </Metadata>
   <Reposid>A0000001.A57A0KGS</Reposid>
   <Ns>SAS</Ns>
   <Flags>268435456</Flags>
   <Options/>
</AddMetadata>
```

This is the output received from the SAS Metadata Server:

```
<AddMetadata>
  <Metadata>
    <Tree Desc="Top level node for clinical study information."
     Name="Clinical Studies" TreeType="Clinical Study" Id="A57A0KGS.A1000006"/>
   </Metadata>
   <Reposid>A0000001.A57A0KGS</Reposid>
   <Ns>SAS</Ns>
   <Flags>268435456</Flags>
   <Options/>
</AddMetadata>
```

The output mirrors the input, except the attribute Id="A57A0KGS.A1000006" is appended to the metadata property string. We will save this value and use it to create references from other objects.

References to objects that have been defined, but for which an identifier has not yet been assigned, can be made by using a **symbolic name**. These concepts are described in greater detail in the sections that follow.

Previous Page | Next Page | Top of Page

Creating the Study 2 Tree

The following XML string creates a Tree object to represent Study 2 as well as an association between the Clinical Studies Tree and the Study 2 Tree. In addition to the metadata type and metadata type attributes, the XML string specifies a nested association name and association subelement (shown in **bold** text) to create the association.

Note in the association subelement that an ObjRef= attribute is used to identify the Clinical Studies Tree as the partner type in the association. An Id= attribute with a real value is never specified in an AddMetadata call. The absence of an Id= attribute, or a blank Id= attribute in an AddMetadata call, indicate to the server that a new object is to be created. The ObjRef= attribute creates a relative reference to an existing object.

```
<AddMetadata>
  <Metadata>
    <Tree
      Desc="All information dealing with Study 2."
      Name="Study 2"
      TreeType="Clinical Study">
      <ParentTree>
         <Tree ObjRef="A57A0KGS.A1000001"/>
      </ParentTree>
    </Tree>
   </Metadata>
   <Reposid>A0000001.A57A0KGS</Reposid>
   <Ns>SAS</Ns>
   <Flags>268435456</Flags>
   <Options/>
</AddMetadata>
```

The output received from the server is as follows:

```
<AddMetadata>
  <Metadata>
    <Tree Desc="All information dealing with Study 2." Name="Study 2"
      TreeType="Clinical Study" Id="A57A0KGS.A1000008">
      <ParentTree>
         <Tree ObjRef="A57A0KGS.A1000001"/>
      </ParentTree>
    </Tree>
   </Metadata>
   <Reposid>A0000001.A57A0KGS</Reposid>
   <Ns>SAS</Ns>
   <Flags>268435456</Flags>
   <Options/>
</AddMetadata>
```

The Study 2 Tree has been assigned the ID value A57A0KGS.A1000008. The association name and subelement are stored as properties of the metadata object.

Previous Page | Next Page | Top of Page

Creating the Study 1 Tree and Table−related Objects

This topic describes the input XML string necessary to add objects for all of the Study 1 metadata types identified in Overview of Adding Metadata Objects except for TransformationActivity, TextStore, and ServerComponent. The XML string is an example of a **stacked** AddMetadata request. In a stacked request, multiple object definitions are provided, one after another, in a single method call. Often, symbolic names are used to serve as identifiers until real identifiers can be assigned.

Explanatory text is interspersed within the sample code.

The first part of the input XML string defines the Study 1 Tree:

```
<AddMetadata>
  <Metadata>
    <Tree
      Desc="All information dealing with Study 1."
      Id="$Study1"
      Name="Study 1"
      TreeType="Clinical Study">
      <ParentTree>
         <Tree ObjRef="A57A0KGS.A1000001"/>
      </ParentTree>
    </Tree>
```

Desc, Name, and TreeType are metadata type attributes. The Id= attribute specifies the symbolic name $Study1. A symbolic name is simply an alias that is preceded by a dollar sign ($). The alias enables you to refer back to the object that is being created before the server assigns it an identifier. The request also creates an association between the Study 1 and Clinical Studies trees.

The following request defines the Document object:

```
    <Document
      Desc="Documentation for clinical study."
      Name="Specification For Clinical Study 1"
      URI="http://webserver.xyz.com/doc/Study.html"
      URIType="URL">
      <Trees>
         <Tree ObjRef="$Study1"/>
      </Trees>
    </Document>
```

Desc, Name, URI, and URIType are metadata type attributes. In the SAS Open Metadata Architecture, a Document is a Web page that contains documentation pertinent to the object to which this document is related. The URI attribute of the Document object specifies the URL of the Web document. The URIType attribute identifies the type of location.

The ObjRef= attribute in the Tree subelement specifies the symbolic name $Study1 to create an association to the Study 1 object. The symbolic name is replaced with a real Id= value at the successful completion of the AddMetadata request.

The following requests define the ResponsibleParty, Person, and Email objects.

```
    <ResponsibleParty
      Desc="Owner of clinical studies."
      Id="$ResponsibleParty1"
      Name="Owner of Study 1"
      Role="OWNER">
      <Objects>
         <Tree ObjRef="$Study1"/>
      </Objects>
      <Persons>
         <Person ObjRef="$Person1"/>
      </Persons>
    </ResponsibleParty>
    <Person
      Desc="Manager of clinical studies."
      Id="$Person1"
      Name="Joe E. Doe"
      Title="Manager of Clinical Studies">
    </Person>
    <Email
      Address="J.E.Doe@xyz.com"
      Desc="Primary e-mail address."
      Name="e-mail address for Joe E. Doe">
      <Persons>
         <Person ObjRef="$Person1"/>
      </Persons>
    </Email>
```

Symbolic names are defined and used to create the following associations:

- The symbolic name $Study1 creates an Objects association between the ResponsibleParty object and the Study 1 Tree object defined earlier in the input XML string.
- The symbolic name $Person1 creates a Persons association between the ResponsibleParty object and a Person object, which has not yet been defined.
- The symbolic name $Person1 creates a Persons association between the Email object and the Person object.
- The symbolic name $ResponsibleParty1 is defined but is not referenced.

The following object requests define the properties for Patient Information table and its columns:

```
  <PhysicalTable
    Desc="Information describing an individual patient."
    Id="$PatientInformation"
    MemberType="DATA"
    Name="Patient Information"
    SASTableName="Patient_Information"
    TableName="Patient_Information">
    <Trees>
       <Tree ObjRef="$Study1"/>
    </Trees>
  </PhysicalTable>
  <Column
    Name="Patient ID"
    Desc="Patient Information"
    ColumnName="Patient_ID"
    SASColumnName="Patient_ID"
    ColumnType="12"
    SASColumnType="C"
    ColumnLength="32"
    SASColumnLength="32"
    SASFormat="$Char32."
    SASInformat="$32.">
    <Table>
       <PhysicalTable ObjRef="$PatientInformation"/>
    </Table>
  </Column>
  <Column
    Name="Initials"
    Desc="Patient Initials"
    ColumnName="Initials"
    SASColumnName="Initials"
    ColumnType="12"
    SASColumnType="C"
    ColumnLength="3"
    SASColumnLength="3"
    SASFormat="$Char3."
    SASInformat="$3.">
    <Table>
       <PhysicalTable ObjRef="$PatientInformation"/>
    </Table>
  </Column>
  <Column
    Name="Sex"
    Desc="Sex of Patient"
    ColumnName="Sex"
    SASColumnName="Sex"
    ColumnType="12"
    SASColumnType="C"
    ColumnLength="1"
    SASColumnLength="1"
    SASFormat="$Char1."
    SASInformat="$1.">
    <Table>
       <PhysicalTable ObjRef="$PatientInformation"/>
    </Table>
  </Column>
  <Column
    Name="Date Of Birth"
    Desc="Date Of Birth"
    ColumnName="Date_Of_Birth"
    SASColumnName="Date_Of_Birth"
    ColumnType="91"
    SASColumnType="N"
    ColumnLength="9"
    SASColumnLength="9"
    SASFormat="date9."
    SASInformat="date9.">
    <Table>
       <PhysicalTable ObjRef="$PatientInformation"/>
    </Table>
  </Column>
  <Column
    Name="Sponsor Patient ID"
    Desc="Sponsor Patient Information"
    ColumnName="Sponsor_Patient_ID"
    SASColumnName="Sponsor_Patient_ID"
    ColumnType="12"
    SASColumnType="C"
    ColumnLength="32"
    SASColumnLength="32"
    SASFormat="$Char32."
    SASInformat="$32.">
    <Table>
```

```
      <PhysicalTable ObjRef="$PatientInformation"/>
   </Table>
</Column>
<Column
   Name="Weight In Pounds"
   Desc="Patient Weight In Pounds"
   ColumnName="Weight_In_Lb"
   SASColumnName="Weight_In_Lb"
   ColumnType="6"
   SASColumnType="N"
   ColumnLength="6"
   SASColumnLength="6"
   SASFormat="6.2"
   SASInformat="6.2">
   <Table>
      <PhysicalTable ObjRef="$PatientInformation"/>
   </Table>
</Column>
<Column
   Id="$WeightInKgColumn"
   Name="Weight In Kilograms"
   Desc="Patient Weight In Kilograms"
   ColumnName="Weight_In_Kg"
   SASColumnName="Weight_In_Kg"
   ColumnType="6"
   SASColumnType="N"
   ColumnLength="6"
   SASColumnLength="6"
   SASFormat="6.2"
   SASInformat="6.2">
   <Table>
      <PhysicalTable ObjRef="$PatientInformation>"/>
   </Table>
</Column>
```

In the requests:

- The symbolic name $Study1 creates a Trees association between the Patient Information table and the Study 1 Tree objects.
- The symbolic name $PatientInformation is in turn assigned to the Patient Information table object and creates a Columns association between the table and each of its columns.
- Note that the column definitions specify attributes both as they are defined in a DBMS and as they are defined by SAS software.
- The symbolic name $WeightInKgColumn is assigned to the Weight in Kilograms column and is not yet referenced.

The following request defines an Extension object for the Weight In Kilograms column object.

```
<Extension
   Desc="Algorithm for column."
   Name="Algorithm"
   Value="Weight_In_Lb\2.2">
   <OwningObject>
      <Column ObjRef="$WeightInKgColumn"/>
   </OwningObject>
</Extension>
```

The extension contains an algorithm for converting pounds to kilograms. The symbolic name $WeightInKgColumn is used to define an OwningObject association between the Extension and the Weight In Kilograms column objects.

The following object requests define the properties for the Visit Information table and its columns:

```
<PhysicalTable
   Desc="Information describing a patient visit."
   Id="$PatientVisit"
   MemberType="DATA"
   Name="Visit Information"
   SASTableName="Patient_Visit"
   TableName="Patient_Visit">
   <Trees>
      <Tree ObjRef="$Study1"/>
   </Trees>
 </PhysicalTable>
 <Column
   Name="Visit Name"
   Desc="Visit Name"
   ColumnName="Visit_Name"
   SASColumnName="Visit_Name"
   ColumnType="12"
   SASColumnType="C"
   ColumnLength="32"
   SASColumnLength="32"
   SASFormat="$Char32."
   SASInformat="$32.">
   <Table>
      <PhysicalTable ObjRef="$PatientVisit"/>
   </Table>
 </Column>
 <Column
   Name="Sponsor Patient ID"
   Desc="Sponsor Patient ID"
   ColumnName="Sponsor_Patient_ID"
   SASColumnName="Sponsor_Patient_ID"
   ColumnType="12"
   SASColumnType="C"
   ColumnLength="32"
   SASColumnLength="32"
   SASFormat="$Char32."
   SASInformat="$32.">
   <Table>
      <PhysicalTable ObjRef="$PatientVisit"/>
   </Table>
 </Column>
 <Column
   Name="Patient ID"
   Desc="Patient ID"
   ColumnName="Patient_ID"
   SASColumnName="Patient_ID"
   ColumnType="12"
   SASColumnType="C"
   ColumnLength="32"
   SASColumnLength="32"
   SASFormat="$Char32."
   SASInformat="$32.">
   <Table>
      <PhysicalTable ObjRef="$PatientVisit"/>
```

```
  </Table>
</Column>
<Column
  Name="Systolic Blood Pressure"
  Desc="Systolic Blood Pressure"
  ColumnName="Systolic_Blood_Pressure"
  SASColumnName="Systolic_Blood_Pressure"
  ColumnType="4"
  SASColumnType="N"
  ColumnLength="4"
  SASColumnLength="4"
  SASFormat="4.0"
  SASInformat="4.0">
  <Table>
     <PhysicalTable ObjRef="$PatientVisit"/>
  </Table>
</Column>
<Column
  Name="Diastolic Blood Pressure"
  Desc="Diastolic Blood Pressure"
  ColumnName="Diastolic_Blood_Pressure"
  SASColumnName="Diastolic_Blood_Pressure"
  ColumnType="4"
  SASColumnType="N"
  ColumnLength="4"
  SASColumnLength="4"
  SASFormat="4.0"
  SASInformat="4.0">
  <Table>
     <PhysicalTable ObjRef="$PatientVisit"/>
  </Table>
</Column>
<Column
  Name="Visit Number"
  Desc="Visit Number"
  ColumnName="Visit_Number"
  SASColumnName="Visit_Number"
  ColumnType="4"
  SASColumnType="N"
  ColumnLength="4"
  SASColumnLength="4"
  SASFormat="4.0"
  SASInformat="4.0">
  <Table>
     <PhysicalTable ObjRef="$PatientVisit"/>
  </Table>
</Column>
<Column
  Name="Occurrence Number"
  Desc="Occurrence Number"
  ColumnName="Occurrence_Number"
  SASColumnName="Occurrence_Number"
  ColumnType="4"
  SASColumnType="N"
  ColumnLength="4"
  SASColumnLength="4"
  SASFormat="4.0"
  SASInformat="4.0">
  <Table>
     <PhysicalTable ObjRef="$PatientVisit"/>
  </Table>
```

```
    </Column>
```

In the request:

- The symbolic name $Study1 creates a Trees association between the Visit Information table and the Study 1 Tree objects.
- The symbolic name $PatientVisit is in turn assigned to the Visit Information table object and creates a Columns association between the table and each of its columns.

Finally, the following object requests define the Study 1 output table and its columns and complete the AddMetadata call:

```
   <PhysicalTable
      Desc="Information output from Study 1."
      Id="$Study1Output"
      MemberType="DATA"
      Name="Study 1 Output"
      SASTableName="Study_1_Output"
      TableName="Study_1_Output">
      <Trees>
         <Tree ObjRef="$Study1"/>
      </Trees>
    </PhysicalTable>
    <Column
      Name="Patient ID"
      Desc="Patient Information"
      ColumnName="Patient_ID"
      SASColumnName="Patient_ID"
      ColumnType="12"
      SASColumnType="C"
      ColumnLength="32"
      SASColumnLength="32"
      SASFormat="$Char32."
      SASInformat="$32.">
      <Table>
         <PhysicalTable ObjRef="$Study1Output"/>
      </Table>
    </Column>
    <Column
      Name="Visit Name"
      Desc="Visit Name"
      ColumnName="Visit_Name"
      SASColumnName="Visit_Name"
      ColumnType="12"
      SASColumnType="C"
      ColumnLength="32"
      SASColumnLength="32"
      SASFormat="$Char32."
      SASInformat="$32.">
      <Table>
         <PhysicalTable ObjRef="$Study1Output"/>
      </Table>
    </Column>
    <Column
      Name="Normalized SBPW Coefficient"
      Desc="Normalized SBPW Coefficient"
      ColumnName="SBPW_Coefficient"
      SASColumnName="SBPW_Coefficient"
      ColumnType="6"
      SASColumnType="N"
      ColumnLength="8"
      SASColumnLength="8"
      SASFormat="8.2"
      SASInformat="8.2">
      <Table>
         <PhysicalTable ObjRef="$Study1Output"/>
      </Table>
    </Column>
   </Metadata>
   <Reposid>A0000001.A57A0KGS</Reposid>
   <Ns>SAS</Ns>
   <Flags>268435456</Flags>
   <Options/>
</AddMetadata>
```

In the requests:

- The symbolic name $Study1 creates a Trees association between the Study 1 output table and the Study 1 Tree objects.
- The symbolic name $Study1Output is assigned to the Study 1 output table object and creates a Columns association between the table and each of its columns.

You might have noticed that the metadata property strings that define table columns specify an integer in the ColumnType attribute. For example:

```
    <Column
      Name="Visit Name"
      Desc="Visit Name"
      ColumnName="Visit_Name"
      SASColumnName="Visit_Name"
      ColumnType="12"
      SASColumnType="C"
      ColumnLength="32"
      SASColumnLength="32"
      SASFormat="$Char32."
      SASInformat="$32.">
      <Table>
         <PhysicalTable ObjRef="$Study1Output"/>
      </Table>
    </Column>
```

The ColumnType attribute describes the SQL type of a DBMS column as an integer value. The integer "12" corresponds to the VARCHAR SQL type. In other column definitions in this example, the integer "6" corresponds to the FLOAT SQL type and the integer "4" corresponds to the INTEGER SQL type. See the description of the ColumnType attribute in the documentation for the Column metadata type in the **SAS 9.1 Open Metadata Interface: Reference** for a complete list of supported integer and SQL type values.

Previous Page | Next Page | Top of Page

Creating the TransformationActivity, TextStore, and ServerComponent Objects

The following input XML string defines the metadata properties for the TransformationActivity, TextStore, and ServerComponent objects described in Overview of Adding Metadata Objects. This is also a stacked AddMetadata request. Explanatory text is been interspersed throughout the sample code.

The first part of the input XML string defines the TransformationActivity object:

```
<AddMetadata>
  <Metadata>
    <TransformationActivity
      Desc="Program for calculating drug efficacy."
      Id="$TransformationActivity1"
      Name="Program For Calculating Drug Efficacy">
      <TransformationSources>
         <PhysicalTable ObjRef="A57A0KGS.A7000001"/>
         <PhysicalTable ObjRef="A57A0KGS.A7000002"/>
      </TransformationSources>
      <TransformationTargets>
         <PhysicalTable ObjRef="A57A0KGS.A7000003"/>
      </TransformationTargets>
      <Trees>
         <Tree ObjRef="A57A0KGS.A100002T"/>
       </Trees>
    </TransformationActivity>
```

In the request:

- The symbolic name $TransformationActivity1 is assigned to enable other objects to create an association to the TransformationActivity object.
- A TransformationSources association is defined between the TransformationActivity object and the Patient Information and Patient Visit objects that were defined in the previous AddMetadata request. Real identifiers are specified in the ObjRef= attribute.
- A TransformationTargets association is defined between the TransformationActivity object and the Study 1 Output table object that was defined in the previous AddMetadata request.
- A Trees association is defined between the TransformationActivity object and the Study 1 Tree object defined in the previous AddMetadata request.

The following requests define the TextStore objects that contain the study notes and the DATA step that performs the transformation.

```
    <TextStore
      Desc="Details About The Program For Calculating Normalized SBPW Coefficient."
      Name="Details About The Program For Calculating Normalized SBPW Coefficient"
      StoredText="The Normalized SBPW Coefficient is used to look
         for correlations between Systolic Blood Pressure, Weight,
         and reactions described in the Visit_Name column."
      TextRole="NOTE"
      TextType="TEXT">
      <Objects>
         <TransformationActivity ObjRef="$TransformationActivity1"/>
      </Objects>
    </TextStore>
    <TextStore
      Desc="Source code for calculating Normalized SBPW Coefficient."
      Name="Source code for calculating Normalized SBPW Coefficient"
      StoredText=" proc sort data=Patient_Information; by Patient_ID;run;\n
                   proc sort data=Patient_Visit; by Patient_ID;run;\n
                   data Study_1_Output;\n
                      keep Patient_ID Visit_Name SBPW_Coefficient;\n
                      format SBPW_Coefficient 8.2;\n
                      merge Patient_Visit Patient_Information;\n
                      by Patient_ID;\n
                      SBPW_Coefficient = (Systolic_Blood_Pressure * Weight_In_Lb)/100; \n
                      run;"
      TextRole="SOURCE"
      TextType="DATASTEP">
      <AssociatedTransformation>
         <TransformationActivity ObjRef="$TransformationActivity1"/>
      </AssociatedTransformation>
    </TextStore>
```

The respective text stores are specified in a StoredText attribute. The TextRole and TextType attributes describe the content of the text. In addition, the symbolic name $TransformationActivity1 creates an Objects association between each TextStore object and the TransformationActivity object.

The following request defines the ServerComponent object, which describes the software that performs the transformation. In addition, it completes the AddMetadata call.

```
    <ServerComponent
      Desc="SAS Software"
      IsLicensed="1"
      Major="9"
      Minor="0"
      Name="SAS Software on olive.us.xyz.com"
      ProductName="The SAS System"
      SoftwareVersion="9.0"
      Vendor="SAS Institute">
      <ComputeTasks>
         <TransformationActivity ObjRef="$TransformationActivity1"/>
      </ComputeTasks>
    </ServerComponent>
  </Metadata>
  <Reposid>A0000001.A57A0KGS</Reposid>
  <Ns>SAS</Ns>
  <Flags>268435456</Flags>
  <Options/>
</AddMetadata>
```

In the request:

- The ProductName, SoftwareVersion, and Vendor attributes provide the software name, version, and vendor (SAS System, Version 9, from SAS Institute).
- the Major and Minor attributes identify the major and minor release numbers associated with the software, and a value of 1 in the IsLicensed attribute indicates the software is a licensed component.
- the symbolic name $TransformationActivity1 creates a ComputeTasks association between the ServerComponent object and the TransformationActivity object that was defined at the beginning of the AddMetadata request.

The repository has been populated with metadata objects.

Note: The OMI_TRUSTED_CLIENT flag (268435456) is required by methods that support write operations. This flag must be specified in all AddMetadata calls.

We are now ready to query the repository.

Previous Page | Next Page | Top of Page

Overview of Querying the Repository

Now that we have objects in the myrepos repository, we can use the SAS Open Metadata Interface to list them and to query their attributes and associations. The SAS Open Metadata Interface provides the GetMetadata and GetMetadataObject methods for performing queries.

- GetMetadataObjects gets metadata objects when passed the repository and type.
- GetMetadata reads specified metadata from a repository.

This section contains XML–formatted method calls that show you how to

- list objects of a given metadata type
- list properties of specific objects
- use a search string to filter a metadata request.

These simple calls get you started using the SAS Open Metadata Interface query methods. For a detailed description of the methods and information about advanced features, see the **SAS 9.1 Open Metadata Interface: User's Guide**.

Previous Page | Next Page | Top of Page

Listing Objects of a Given Metadata Type

You list metadata objects by using the GetMetadataObjects method. The GetMetadataObjects method retrieves all objects of a given metadata type and provides options for expanding and filtering the request. The following XML input string shows how to format a GetMetadataObjects request in order to return all objects that are of metadata type Tree.

```
<GetMetadataObjects>
  <Reposid>A0000001.A57A0KGS</Reposid>
  <Type>Tree</Type>
  <Objects/>
  <NS>SAS</NS>
  <Flags>0</Flags>
  <Options/>
</GetMetadataObjects>
```

The output received from the server is as follows:

```

  <Reposid>A0000001.A57A0KGS</Reposid>
  <Type>Tree</Type>
  <Objects>
  <Tree Id="A57A0KGS.A1000001" Name="Clinical Studies"/>
  <Tree Id="A57A0KGS.A1000002" Name="Study 2"/>
  <Tree Id="A57A0KGS.A100002T" Name="Study 1"/>
  </Objects>
  <NS>SAS</NS>
  <Flags>0</Flags>
  <Options/>
</GetMetadataObjects>
```

The GetMetadataObjects method lists general, identifying information about an object (the Id= and Name= attributes). You can request additional attributes for each of the objects retrieved by GetMetadataObjects by setting the OMI_GET_METADATA flag in the method call and one or more other GetMetadata method flags. To request additional attributes and information for a specific object returned by GetMetadataObjects, you must use the GetMetadata method.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Requesting Properties for Specific Objects

To retrieve properties for a specific object, use the GetMetadata method. The GetMetadata method enables you to retrieve

- specific properties for the requested object
- a particular category of properties for the requested object (all attributes, all associations, and so forth)
- properties for associated objects
- any combination of the above.

Here are some usage suggestions:

- Identify the requested object and any specific attributes and associations that you want to retrieve in the <Metadata> element (*inMetadata* parameter).
- Use SAS Open Metadata Interface flags to request categories of properties or to indicate that special processing is needed. For information about available flags, see GetMetadata in the **SAS 9.1 Open Metadata Interface: Reference**.
- To request information about associated objects, use flags or templates. A template is an additional metadata property string that is passed to the metadata server in a <Templates> XML element in the *Options* parameter of a GetMetadata method call. The OMI_TEMPLATE (4) flag must be set so that the server knows to look for this element. The properties specified in a template augment the properties requested by other GetMetadata parameters.

The following is an example of a GetMetadata call that uses a template to request specific attributes of the objects that are associated with the Study 1 Tree. To assist you in following the example, the requested object is highlighted in **bold**, associations and associated objects are highlighted in ***bold−italic***, and template components are highlighted in **italic**.

```
<GetMetadata>
  <Metadata>
    <Tree Id="A57A0KGS.A100002T">
       <Members/>
       <ResponsibleParties/>
    </Tree>
  </Metadata>
  <NS>SAS</NS>
  <Flags>4</Flags> <!-- OMI_TEMPLATES flag-->
  <Options>
     <Templates>
       <Tree
            Desc=""
            Id=""
            Name=""
            TreeType=""/>
       <Document
            Desc=""
            Id=""
            Name=""
            URI=""
            URIType=""/>
      <PhysicalTable
            Desc=""
            Id=""
            MemberType=""
            Name=""
            SASTableName=""
            TableName="">
            <Columns/>
       </PhysicalTable>
       <Column
            Name=""
            Id=""
            Desc=""
            ColumnName=""
            SASColumnName=""
            ColumnType=""
            SASColumnType=""
            ColumnLength=""
            SASColumnLength=""
            SASFormat=""
            SASInformat="">
            <Extensions/>
        </Column>
        <Extension
            Desc=""
            Name=""
            Value=""/>
        <TransformationActivity
            Desc=""
            Id=""
            Name="">
            <ComputeLocations/>
            <Notes/>
            <SourceCode/>
            <TranformationSources/>
            <TransformationTargets/>
        </TransformationActivity>
        <ServerComponent
             Desc=""
             Id=""
             IsLicensed=""
             Major=""
             Minor=""
             Name=""
             ProductName=""
             SoftwareVersion=""
             Vendor=""/>
        <TextStore
             Desc=""
             Name=""
             StoredText=""
             TextRole=""
             TextType=""/>
        <ResponsibleParty
             Desc=""
             Id=""
             Name=""
             Role="">
             <Persons/>
        </ResponsibleParty>
        <Person
             Desc=""
             Id=""
             Name=""
             Title="">
             <EmailAddresses/>
```

```
        </Person>
        <Email
             Address=""
             Desc=""
             Id=""
             Name=""/>
      </Templates>
   </Options>
</GetMetadata>
```

In the request, note the following:

- The <Metadata> element identifies the top–level node in the object hierarchy and the associations that are queried.
- The <Flags> element contains the OMI_TEMPLATE (4) flag.
- The <Templates> element contains 11 templates, which request specific attributes of associated objects of type Tree, Document, PhysicalTable, Column, Extension, TransformationActivity, ServerComponent, TextStore, ResponsibleParty, Person, and Email.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Using a Search String to Filter a Metadata Request

You can filter the metadata objects returned by the SAS Metadata Server by including an <XMLSelect> element in the GetMetadataObjects call. The <XMLSelect> element enables you to pass a search string to the server in the *Options* parameter. The OMI_XMLSELECT (128) flag must be set so that the server knows to look for the element in the *Options* parameter.

The <XMLSelect> search syntax is described in Filtering a GetMetadataObjects Request in the **SAS 9.1 Open Metadata Interface: User's Guide**. In the following example, we will use a subset of that functionality to perform a simple attribute=value search. The following GetMetadataObjects call uses the <XMLSelect> element to return TextStore objects that have the value "SOURCE" in the TextRole= attribute. All other objects are filtered from the object request.

```
<GetMetadataObjects>
  <Reposid>A0000001.A57A0KGS</Reposid>
  <Type>TextStore</Type>
  <Objects/>
  <NS>SAS</NS>
  <!--OMI_XMLSelect flag-->
  <Flags>128</Flags>
  <Options>
    <XMLSelect search="@TextRole = 'SOURCE'"/>
  </Options>
</GetMetadataObjects>
```

In the search string, note the following:

- The "at" symbol (@) denotes that the attached string is an attribute name. We are searching for the TextRole attribute.
- The equal sign (=) invokes a matching operation.
- SOURCE is the value to match, specified within single quotation marks. Matches are case–insensitive unless the OMI_MATCH_CASE (512) flag is also set.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Updating Metadata Objects in the Repository

Suppose we encounter an error in the DATA step code that we stored in one of the TextStore objects and need to replace it with a corrected version. The SAS Open Metadata Interface provides the UpdateMetadata method for updating existing objects. The following XML input string contains a sample UpdateMetadata call. The call specifies to replace the text in the StoredText attribute of TextStore A57A0KGS.AB000002 with the text supplied in the method call.

```
<UpdateMetadata>
   <Metadata>
     <TextStore
       Id="A57A0KGS.AB000002"
       StoredText="proc sort data=Patient_Information; by Patient_ID;run;\n
                   proc sort data=Patient_Visit; by Patient_ID;run;\n
                   data Study_1_Output;\n
                     keep Patient_ID Visit_Name SBPW_Coefficient;\n
                     format SBPW_Coefficient 8.2;\n
                     merge Patient_Visit Patient_Information;\n
                     by Patient_ID;\n
                     SBPW_Coefficient = (Systolic_Blood_Pressure * Weight_In_Lb)/100;\n
                     run;">
      </TextStore>
   </Metadata>
   <NS>SAS</NS>
   <!--OMI_TRUSTED_CLIENT flag-->
   <Flags>268435456</Flags>
   <Options/>
</UpdateMetadata>
```

The UpdateMetadata method can also be used to add or modify associations between objects. For more information about updating metadata objects, see the **SAS 9.1 Open Metadata Interface: User's Guide**.

Previous Page | Next Page | Top of Page

Getting Started with the SAS 9.1 Open Metadata Interface

Deleting Metadata Objects in the Repository

You delete metadata from a repository by using the DeleteMetadata method. The following XML input string contains a sample DeleteMetadata call. The call deletes the TextStore object that contains the text describing the Study 1 transformation.

```
<DeleteMetadata>
   <Metadata>
     <TextStore Id="A57A0KGS.AB000001"/>
   </Metadata>
   <NS>SAS</NS>
   <!--OMI_TRUSTED_CLIENT + OMI_RETURN_LIST flags-->
   <Flags>268436480</Flags>
   <Options/>
</DeleteMetadata>
```

The <Metadata> element identifies the object to be deleted and two flags are set. The OMI_TRUSTED_CLIENT (268435456) flag is required when you are writing metadata. In order to verify the operation, OMI_RETURN_LIST (1024) is set to return a list of deleted object IDs as well as any cascading object IDs that may have been deleted. For more information about deleting metadata objects, see the **SAS 9.1 Open Metadata Interface: User's Guide**.

Previous Page | Next Page | Top of Page

Stopping the SAS Metadata Server

Note: A user must have Unrestricted User or Administrative User status in order to stop a SAS Metadata Server. For more information, see "Server Administrative Privileges" in the **SAS 9.1 Metadata Server: Setup Guide**.

The person who started the server, or someone who has Unrestricted User status or Administrative User status, can stop the metadata server by using the METAOPERATE procedure or by using SAS Management Console.

Stopping the Server Using PROC METAOPERATE

The following is an example of the statements required to stop a server using PROC METAOPERATE:

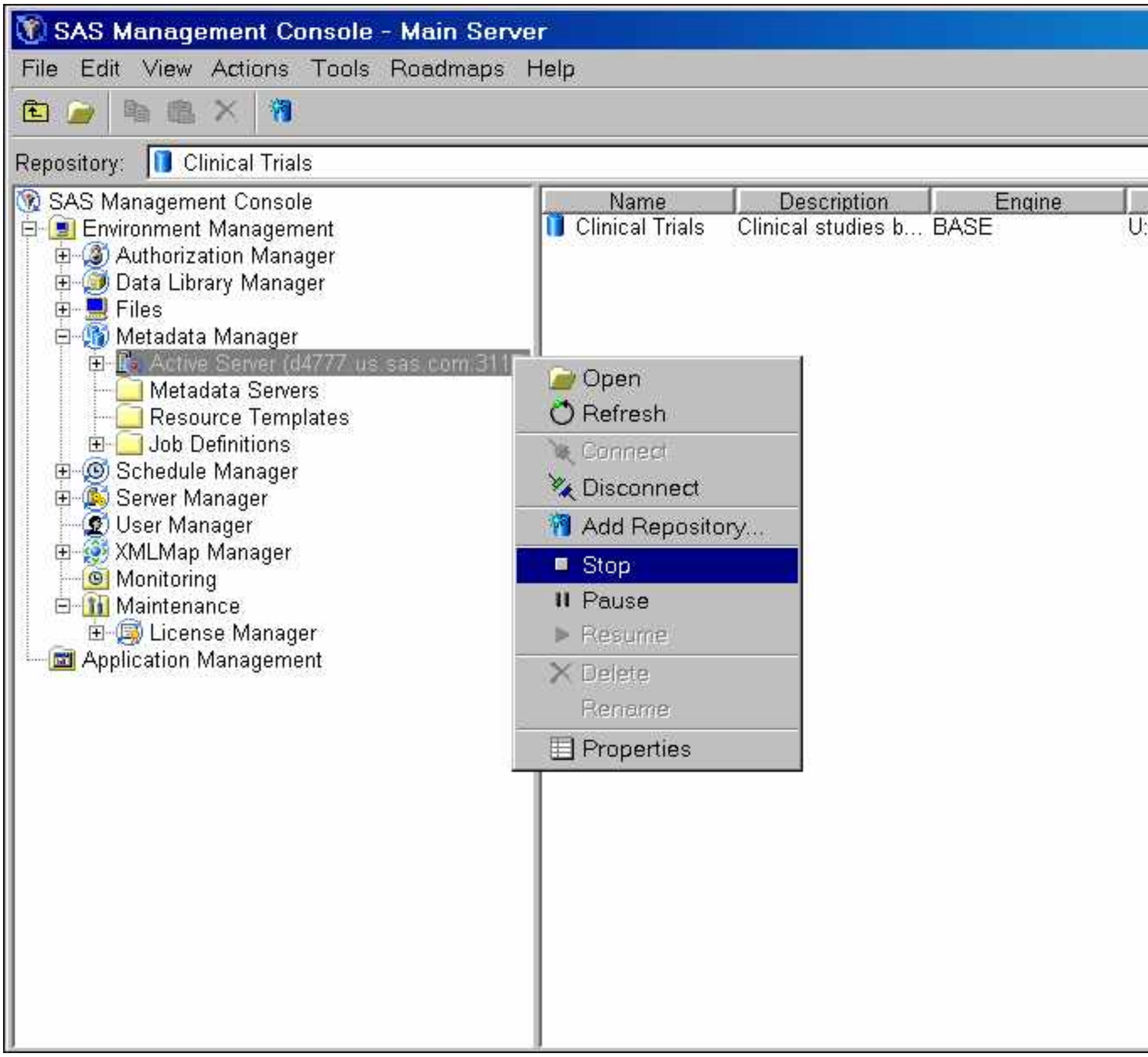
```
PROC METAOPERATE
   SERVER="host_name_of_computer_running_the_server"
   PORT=port_number
   USERID="userid"
   PASSWORD="password"
   PROTOCOL=BRIDGE

   ACTION=STOP;
RUN;
```

The first five statements are server connection parameters. The sixth statement specifies the STOP action.

Stopping the Server Using SAS Management Console

To stop the server from SAS Management Console:

1. From the SAS Management Console main window, expand the Metadata Manager node. The software will expand the tree view to display an icon representing the Active Server and folders for other metadata server definitions, resource templates, and job definitions.
2. With your mouse, right−click the Active Server to display the File pop−up menu, and select **Stop**.



Name	Description	Engine
Clinical Trials	Clinical studies b...	BASE

3. The software will open a dialog box asking you to confirm the operation. Click **OK**.

Previous Page | Next Page | Top of Page