Getting Started with

SAS® 9.1.3 Open Metadata Interface

Second Edition

THE POWER TO KNOW®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *Getting Started with SAS® 9.1.3 Open Metadata Interface, Second Edition*. Cary, NC: SAS Institute Inc.

Getting Started with SAS® 9.1.3 Open Metadata Interface, Second Edition

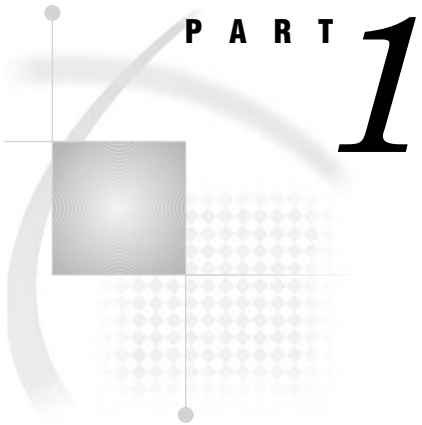
ISBN-13: 978–159994–304–6
ISBN-10: 1–59994–304–2

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

1st printing, November 2007

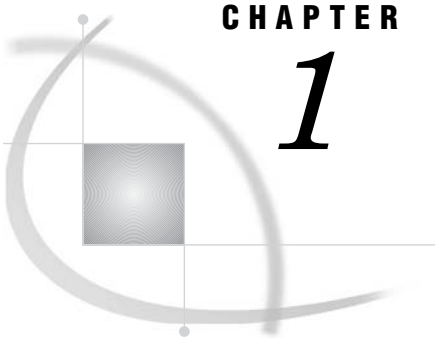
SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents



PART 1

Concepts



CHAPTER

1

Using This Guide

Purpose

This guide is designed to introduce metadata programmers to the SAS Open Metadata Architecture and the SAS Open Metadata Interface. It provides a brief introduction to metadata, metadata management, and the benefits of using the SAS Open Metadata Architecture to store technical details about an application. It also walks readers through the steps necessary to set up a personal SAS Metadata Server and a SAS Metadata Repository for a sample application on Windows XP.

Readers will learn how to identify what information they need to represent in metadata and how to select the metadata types that are best suited to represent the information. The guide also provides sample SAS Open Metadata Interface method calls that create, update, query, and delete metadata.

This guide uses a sample scenario to guide you through the steps of using the SAS Open Metadata Interface. The scenario is not intended to represent the only way that the interface can be used to create metadata. Rather, it is provided as a very simple example of *one* way that it can be used.

To get the most out of this guide, acquaint yourself with the metadata concepts in Chapter 2, "Introduction to Metadata Concepts," on page 5 before reading "Example Scenario" on page 4 and Chapter 3, "Planning a Repository," on page 9. Then read Chapter 5, "Setting Up a SAS Metadata Server," on page 23, Chapter 4, "Using the SAS Open Metadata Interface," on page 15 and Chapter 6, "Adding Metadata Objects to the Repository," on page 41 for information to set up a SAS Metadata Server and write a SAS Open Metadata Interface client to create your own sample repository.

Note: This guide does not describe how to setup and use a SAS Metadata Server in an enterprise environment. For information about how to setup and use a SAS Metadata Server in an enterprise environment, see the administrative documentation for the *SAS Intelligence Platform*. △

Before You Begin

Software Requirements

This guide explains how to set up a personal SAS Metadata Server and use a SAS Open Metadata Interface client in a Windows XP operating environment. To follow along, you need

- SAS 9.1.3 software
- SAS Integration Technologies software (shipped with SAS software)
- SAS Management Console software (SAS Management Console software must be installed from the *SAS Client-Side Components* CD-ROM that is shipped along with SAS software.)
- the appropriate software for the intended programming environment. The SAS Open Metadata Interface supports Java, Visual Basic, C++, and SAS 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.

Example Scenario

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.

Tasks Overview

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

1. Decide what information should be stored.
2. Select the appropriate metadata types.
3. Set up a SAS Metadata Server and a SAS Metadata Repository.
4. Add metadata objects.
5. Query metadata objects.
6. Update and delete metadata objects in the repository.

CHAPTER

2

Introduction to Metadata Concepts

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?

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.

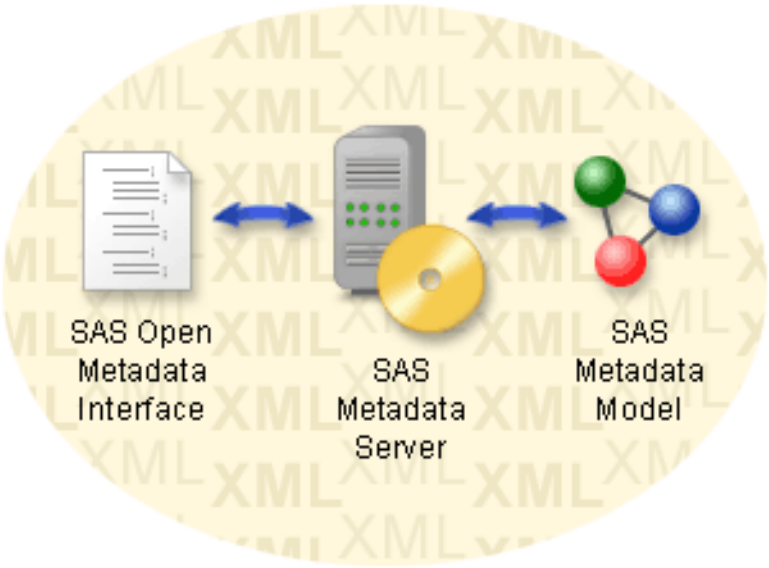
As electronic data transfers and e-commerce increase, the soundness of the metadata supporting these transactions will become as important as the data itself. Support for industry metadata models and data interchange standards enables organizations to respond quickly and economically to rapidly evolving,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

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

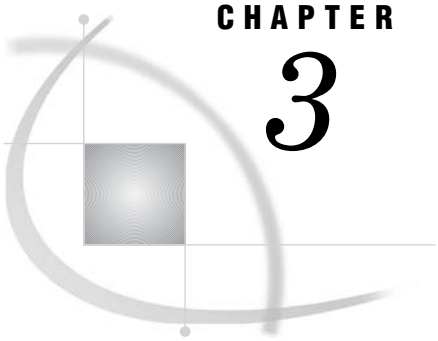
- *the SAS Metadata Server,* a central, shared location for storing metadata
- *the SAS Open Metadata Interface,* an application programming interface (API) that provides access to the server from a variety of programming environments, including Java, COM/DCOM, and SAS
- *the SAS Metadata Model,* a set 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 technical entities 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.



CHAPTER

3

Planning a Repository

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?

Focusing on our usage scenario, 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.

Let us assume that we have a study, which we will call Study 1, that has the following contents:

- Patient data is stored in a SAS table named "Patient Information." This table contains columns that store the 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 columns that store the 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.

Introduction to the SAS Metadata Model

The SAS Metadata Model defines a set of metadata types for representing application entities. Each metadata type has a set of attributes, for example, the name of the entity, its description, and defining characteristics, and a set of associations that describe the entity's relationship to other entities. The metadata type definitions are structured in a hierarchy, and each metadata type inherits the attributes and associations of its supertype.

The SAS Metadata Model defines metadata types that describe approximately 150 entities. To help you determine the correct metadata type to use to represent a particular application entity, the model is broken into submodels, each of which consists of a set of related metadata 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 metadata types that are used to define access controls. Metadata objects based on these metadata 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 that are used to group metadata objects together in a particular context, as well as to construct a hierarchy of metadata objects.

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

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

To see a listing of the metadata types that are included in each submodel, see "SAS Namespace Submodels" in the *SAS Open Metadata Interface: Reference*. To see a complete list of the metadata types that are defined in the SAS Metadata Model, refer to the "Alphabetical Listing of SAS Namespace Metadata Types" in the reference. This listing is available in online versions of the guide only. Look for it in *SAS Help and Documentation* or *SAS OnlineDoc*.

Selecting Metadata Types for Our Study

Selecting an appropriate metadata type is a matter of comparing the metadata types that are defined in each category to see which one best meets 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 a metadata type to represent the study tables.

Choosing a Metadata Type to Represent Data 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 metadata type best suits your needs, check the descriptions and compare the attributes and associations of each metadata type in the "Alphabetical Listing of SAS Namespace Types". 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 metadata types in the Relational submodel describe tables that result from a query or are otherwise transitory in nature. For this example, we use this icon to represent metadata objects of the PhysicalTable metadata type:

A metadata object is an instance of a metadata type. We use the Column metadata type to create our column objects, and represent these with this icon:

Choosing a Metadata Type to Represent a Program

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 by the Transformation metadata type. The Transformation metadata type has associations to other metadata types 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 metadata objects of the Transformation metadata type:

The Transformation metadata type also requires an association to a metadata type that either contains or describes 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 representing 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 the TextStore metadata type to save a copy of the source code (DATA step) on the SAS Metadata Server. We use the following icon represent metadata objects of the TextStore metadata type:

Choosing a Metadata Type 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 in our example 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 metadata objects of 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 Document metadata type is provided to represent 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 metadata objects of 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 metadata types. This is true for the person responsible for the analysis. The Business Information submodel contains metadata types for identifying the person, his role, and contact information, such as e-mail address, phone number, and physical address. The metadata types that we will use in this example are ResponsibleParty, Person, and Email, which we represent as follows:

Choosing a Metadata Type to Represent a Group

The last submodel that we will look at is the Grouping submodel. The Grouping submodel consists of two metadata types: Group and Tree. For our example, we use the Tree metadata type to impose a hierarchy on the metadata objects that we define to describe Study 1. We will define a Tree metadata 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 the following icon to represent the Tree metadata type:

Now that we know what kind of information to save and which metadata types to use to represent the entities, we need to get some administrative tasks out of the way. Follow the steps in Chapter 5, "Setting Up a SAS Metadata Server," on page 23 to set up a SAS Metadata Server, then read Chapter 4, "Using the SAS Open Metadata Interface," on page 15 and Chapter 6, "Adding Metadata Objects to the Repository," on page 41 and we will begin creating metadata objects. A *metadata object* is an instance of a metadata type that is stored in a SAS Metadata Repository.

CHAPTER

4

Using the SAS Open Metadata Interface

Overview of Using the SAS Open Metadata Interface

The SAS Open Metadata Interface can be executed from a Java, Visual Basic, C++, or SAS client. The interface contains three method classes:

- the IOMI class contains methods for reading and writing metadata.
- the IServer class contains methods for controlling SAS metadata repositories and the SAS Metadata Server
- the ISecurity class contains methods for requesting authorizations from the SAS Open Metadata Architecture authorization facility.

In this guide, we are concerned about the IOMI class. The IOMI class provides an XML-based interface to metadata; that is, programmers create, update, and query metadata on the SAS Metadata Server by submitting method calls that specify XML metadata property strings to the server. For information about how to write a metadata property string, see "Writing a Metadata Property String" on page 44.

A client issues methods from all classes by connecting to the SAS Metadata Server and instantiating objects for method parameters. The IOMI method class supports two call interfaces:

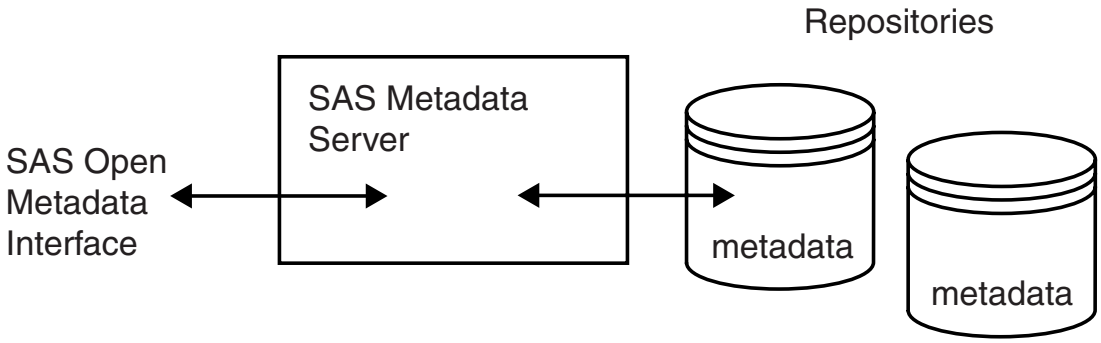
- the standard interface instantiates objects for all method parameters in the client, and parses the methods within the client.
- a messaging interface, implemented via a DoRequest method, instantiates objects for DoRequest parameters in the client and submits methods that read and write metadata to the metadata server in an XML string via the DoRequest method's inMetadata parameter. When the DoRequest method is used, the SAS Metadata Server parses the method call.

The examples in this guide are formatted in XML so that they can be submitted to the metadata server via the DoRequest method. For an example of how to issue a DoRequest method call from a Java, Visual Basic, or C++ client, see "Program-Specific Method Examples" in the *SAS Open Metadata Interface: Reference*.

Most SAS clients provide a client-specific interface to metadata and execute SAS Open Metadata Interface method calls under the covers. An exception is PROC METADATA. PROC METADATA enables users to issue SAS Open Metadata Interface method calls that are formatted for the DoRequest method from a SAS program. See "Using PROC METADATA to Issue a Method Call" on page 19 for more information.

Connecting to the SAS Metadata Server

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



The SAS Metadata Server uses the Integrated Object Model (IOM) provided by SAS Integration Technologies. IOM provides distributed object interfaces to Base SAS software features and enables you to use industry-standard languages, programming tools, and communication protocols to develop client programs that access Base SAS features on IOM servers.

To connect to a SAS Metadata Server, a Java, Visual Basic, or C++ client must invoke the appropriate IOM interface for the programming environment, supply server connection properties, and reference the type library that is appropriate to the method class it intends to use. For details about these requirements, see "Connecting to the SAS Metadata Server" in the *SAS Open Metadata Interface: Reference*.

PROC METADATA enables you to specify server connection properties by using procedure statements or system options.

The following is the minimum information that all clients must specify to connect to a running SAS Metadata Server:

1. The hostname or ip address of the computer hosting the server.
2. The port number specified in the command that was used to start the metadata server. (Multiple users connect to the same server by specifying the same hostname and port number.)
3. A valid user ID and password on the SAS Metadata Server.

Java, Visual Basic, and C++ clients must additionally specify the following information:

- The SAS Metadata Server identifier for Java clients. This property must have the value "2887e7d7-4780-11d4-879f-00c04f38f0db".
- The SAS Metadata Server identifier for Windows clients. This property must have the value "SASOMI.OMI".
- The network protocol. The valid value is "bridge".

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. These interfaces are described in "Call Interfaces" in

"Open Client Requirements" in the *SAS Open Metadata Interface: Reference*. The examples in this guide format method requests for the DoRequest method. See DoRequest in "Methods for Reading and Writing Metadata (IOMI Class)" in the reference.

The DoRequest method allows you to code metadata-related methods and all of their parameters in XML and pass the XML to the metadata server in a generic *inMetadata* parameter. 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 an IOMI class method and <Parameter*n*> represent the parameters required by a given method. Most IOMI class methods contain one or more of the following parameters:

<Metadata></Metadata>
specifies an XML metadata property string that defines the metadata object type and properties to be added, deleted, retrieved, or updated.

<NS></NS>
specifies the namespace to use as the context for the request. Method calls that create, modify, or query application metadata are issued in the SAS namespace. Method calls that query repositories are issued in the REPOS namespace.

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

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

<Reposid></Reposid>
specifies a repository identifier.

<Type></Type>
specifies a metadata type.

<Supertype></Supertype>
specifies an abstract metadata type that has subtypes defined in the SAS Metadata Model.

See the method descriptions in "Methods for Reading and Writing Metadata (IOMI Class)" in the *SAS Open Metadata Interface: Reference* for information about the parameters required by specific methods.

An example of an AddMetadata method call that is formatted for the DoRequest method 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>
```

In the example, <AddMetadata> is the method name, and <Metadata>, <Reposid>, <NS>, <Flags> and <Options> are AddMetadata parameters. The XML elements that are nested within the <Metadata></Metadata> tags comprise a metadata property string that defines the metadata object to be added.

You can submit two or more method calls in one server request by enclosing the XML method calls 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>
 <NS>SAS</NS>
 <Flags>0</Flags>
 <Options/>
</GetMetadata>
</Multiple_Requests>
```

In the preceding example, the first XML method request issues an AddMetadata method call to add a column to PhysicalTable A2345678.A2000001 and the second request — a GetMetadata method call— gets the columns defined for the table to verify the column's addition.

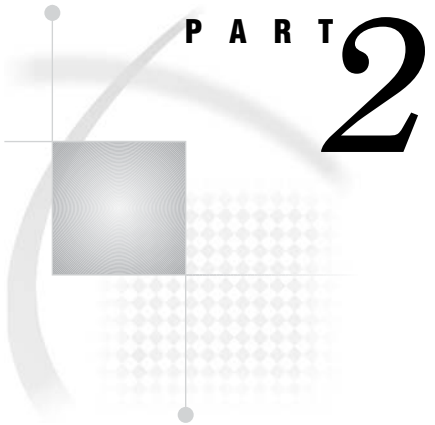
Using PROC METADATA to Issue a Method Call

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;
```

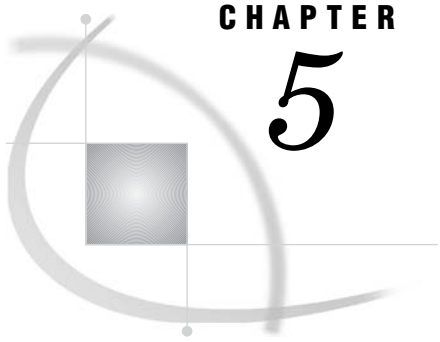
The first five statements supply server connection properties. The IN statement passes the XML-formatted method request to the server.

For more information about PROC METADATA, see the “METADATA Procedure” in the *SAS Open Metadata Interface: Reference*.



PART 2

Preparation



CHAPTER

5

Setting Up a SAS Metadata Server

Overview of Setting up a SAS Metadata Server

This topic leads you through the steps for setting up a personal SAS Metadata Server under Windows XP. These instructions assume that the software described in "Before You Begin" on page 4 is installed.

Note: Do not use these instructions to set up an enterprise SAS Metadata Server. The recommended method for setting up an enterprise SAS Metadata Server is using the SAS Configuration Wizard as directed in the *SAS Intelligence Platform: Installation Guide*. The *SAS Intelligence Platform: Installation Guide* is the intended starting point for all production SAS 9 server deployments. These instructions are provided solely to enable you to set up a personal server for this tutorial. △

To set up a personal SAS Metadata Server on Windows XP:

1. Create directories for the SAS Metadata Server, repository manager, and a repository.
2. Set directory and file access permissions.
3. Set system access permissions.
4. Start the metadata server.
5. Register a repository.

Each step is described in the sections that follow.

Creating Directories for the SAS Metadata Server, Repository Manager, and a Repository

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 a subdirectory of the server directory that is 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, you must change the SAS Metadata Server's default configuration. The server's configuration is managed by creating an omaconfig.xml file in the server directory. For more information about the omaconfig.xml file, see the *SAS Intelligence Platform: System Administration Guide*. To keep things simple, we'll respect the default configuration in this example.

As long as they are not nested within the rposmgr directory, directories for repositories can be created in any location that the SAS Metadata Server can access. For convenience, we'll create the repository directory locally as a subdirectory of the metadata server directory, at the same level as the rposmgr directory. To create the directories:

1. Create a folder and name it **omaserver**.
2. Open the **omaserver**folder.
3. Within the **omaserver** folder, create two subfolders.
4. Name the first folder **rposmgr**.
5. Name the second folder **myrepos**.

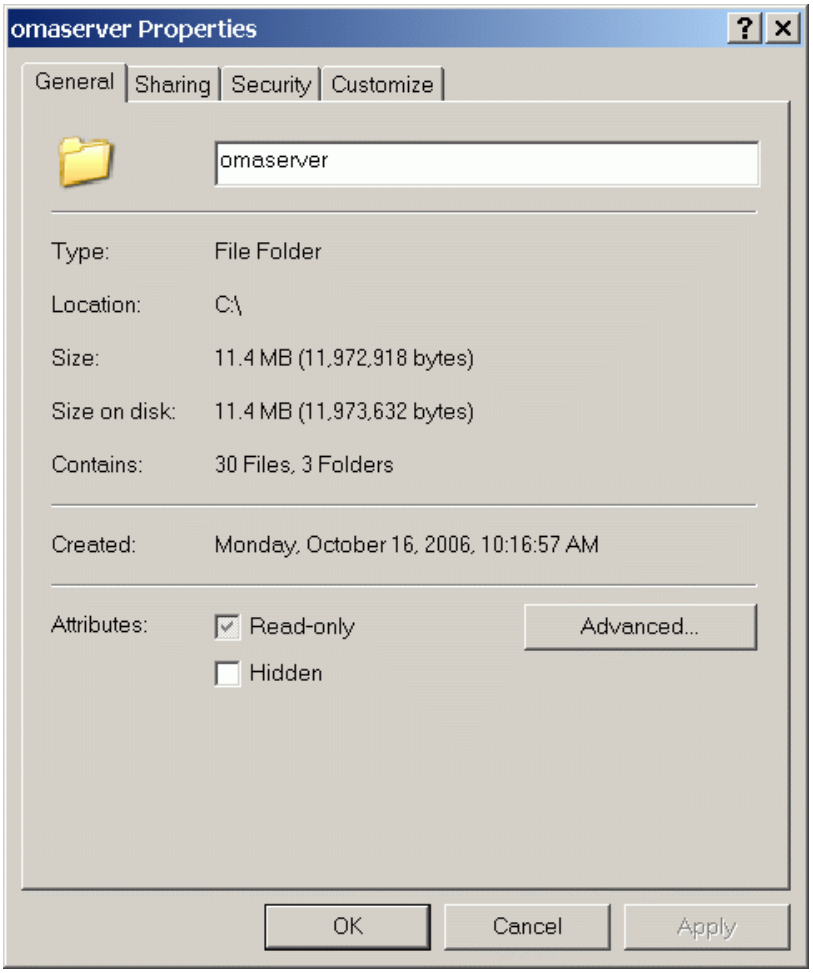
By creating the **rposmgr** and **myrepos** directories as subdirectories of the **omaserver** directory, the directories can inherit the directory and file access permissions that we set for the metadata server directory. Creating the subdirectories as parallel directories prevents contention between them.

Setting Directory and File Access Permissions

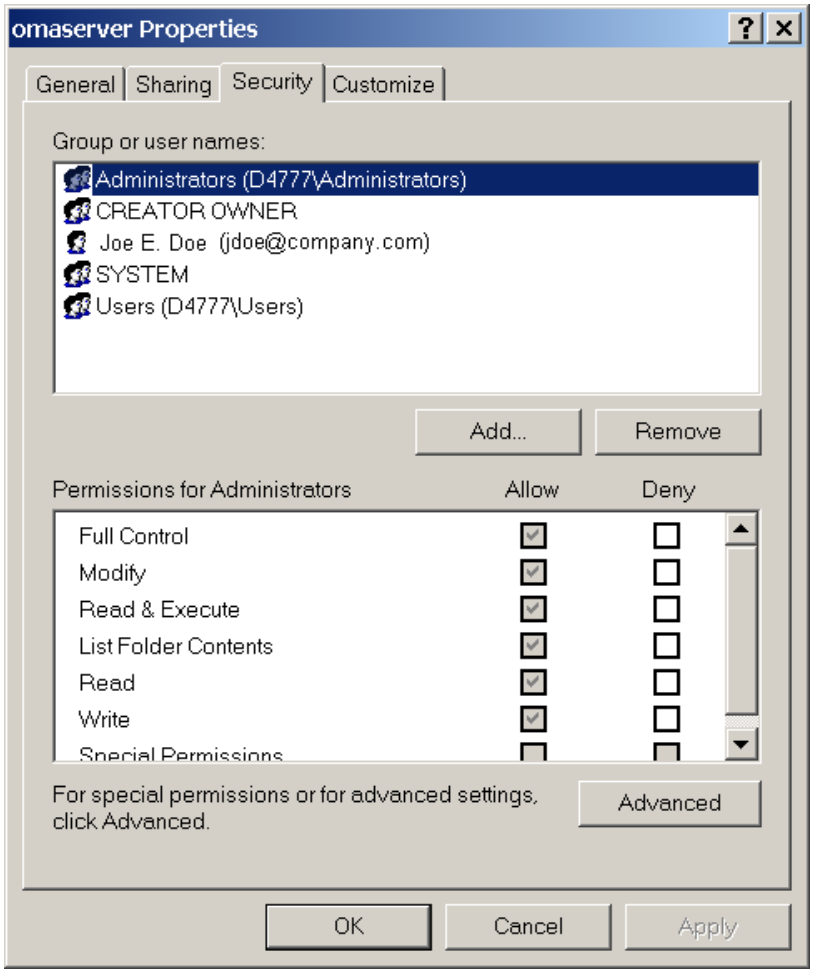
To set up or maintain a repository manager and repository, you must have full access to the repository manager and repository directories. Because the **rposmgr** and **myrepos** directories are subdirectories of the **omaserver** directory, we can set the necessary permissions one time for the **omaserver** directory and the subdirectories will inherit the permissions.

To set the permissions:

1. Highlight the **omaserver** folder icon and select to view its properties. The omaserver Properties window displays as follows:



2 Switch to the Security tab. The *Group or user names:* field lists the identities that have access to this directory and the permissions that are assigned to them.



3 Grant or verify that the following permissions are set:
   - □ Your identity has **`Full Control`** to the directory.
   - □ The administrator identity has appropriate permissions to allow the company's backup procedures to run correctly.
   - □ Click the Advanced button.

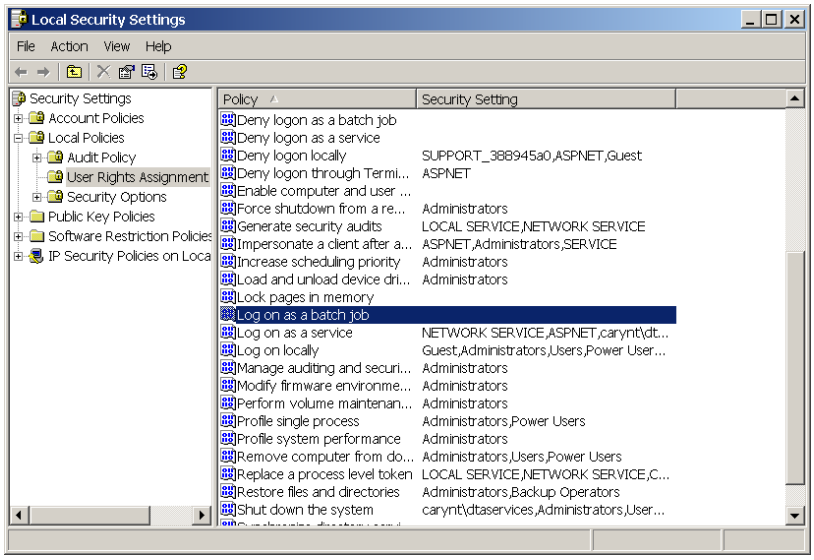
4 In the Advanced Security Settings for omaserver window:
   - **a** Select the *Replace permission entries on all child objects with entries shown here that apply to child objects* checkbox, then click *Apply*.
   - **b** A confirmation dialog will ask you to verify that you want to change the permissions for the child objects. Click *Yes*.
   - **c** The software will return you to the omaserver Properties window. Click OK to close the window.

Setting System Access Permissions

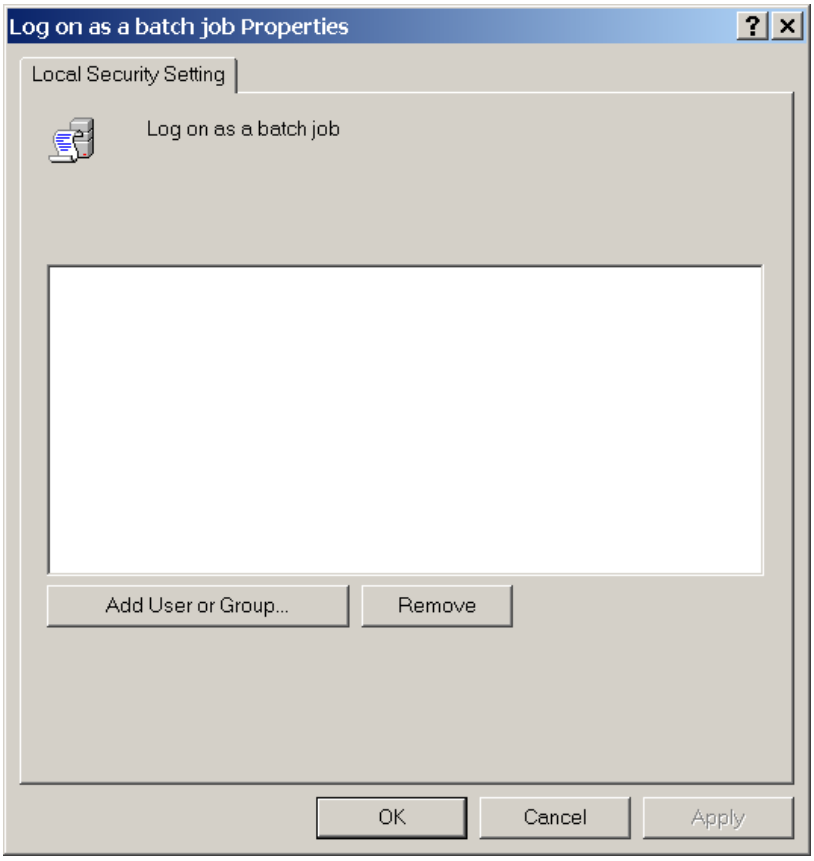
On Windows XP, the operating system account that is used to invoke the SAS Metadata Server is not required to have any special user rights. However, all server accessors (everyone who will use the server) must have the *Log on as a batch job* user right set.

For an enterprise SAS Metadata Server, we recommend that customers create a group, add all server users to the group, and then assign the right to the group. To assign the *Log on as a batch job* user right:

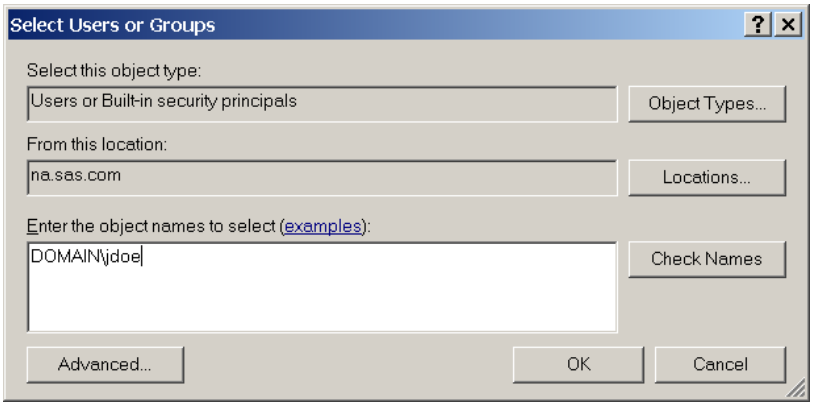
1. From the Windows XP Start menu, select **`Start->Settings->Control Panel`**.
2. In the Control Panel, open **`Administrative Tools`**.
3. In Administrative Tools, open **`Local Security Policy`**.
4. In the Local Security Settings window, select **`User Rights Assignment`**.
5. Scroll the list of policies until you find **`Logon as a batch job`**.



6. Double-click the policy name. The software opens the Logon as a batch job Properties window.



7 In the Logon as a batch job Properties window, click **`Add User or Group`**. The software opens the Select Users or Groups window.



8 Add your user name or the name of a group in which you are a member as shown in the examples provided by the user interface.

9 Click OK to close the Select Users or Groups window.

10 Close the other windows.

11 Restart the computer that is hosting the server so that the updates can take effect.

You are now ready to start the SAS Metadata Server.

Starting the SAS Metadata Server

You are required to start the SAS Metadata Server from the directory that is defined for the SAS Metadata Server. To start the metadata server, create an executable file in the **omaserver** directory that contains the following comand, and then execute the file. For this example, name the executable file "startsrv.bat".

```
"where_your_SAS_is_installed\sas.exe" -nosplash -noterminal -noautoexec
-sasuser sasusrms -rsasuser -log "C:\logs\omaserver.log" -memsize max
-logparm "rollover=auto open=replaceold write=immediate" -linesize max
-pagesize max -objectserver -objectserverparms
"protocol=bridge port=XXXX classfactory=2887E7D7-4780-11D4-879F-00C04F38F0DB"
```

In the command:

- replace *where_your_SAS_is_installed* with the path to the directory where SAS is installed on the host computer (typically C:\Program Files\SAS\SAS System\9.1)
- NOSPLASH, NOTERMINAL, NOAUTOEXEC, SASUSER, and RSASUSER are environment control parameters

  NOSPLASH
  suppresses the display of the SAS splash screen.

  NOTERMINAL
  specifies to run the server in batch mode.

  NOAUTOEXEC
  specifies not to process the SAS autoexec file, even if one exists. Processing of the file can result in unpredictable server behavior.

  SASUSER="*library-specification*"
  specifies the SAS data library to contain the server's profile catalog. This library must be dedicated to the metadata server. We recommend that you name it **sasusrms** or **sasuser**.

  RSASUSER
  limits access to the SASUSER library to read-only mode.
- LOG=, LOGPARM=, PAGESIZE=, and LINESIZE= are log output parameters

  LOG="*directory-name*"
  specify an existing directory to which to write the server log file.

  LOGPARM="WRITE=*value* ROLLOVER=*value* OPEN=*value*"
  controls when SAS log files are opened and closed. Specify the parameter "WRITE=IMMEDIATE"; otherwise, the server buffers log entries in memory. If your server runs in a 24/7 environment, you might also want to specify "ROLLOVER=AUTO OPEN=REPLACEOLD" to partition the log into daily or weekly reports.

  PAGESIZE=*value*
  specifies the number of lines that make up a page of SAS output. A setting of MAX is recommended to reduce the occurrence of SAS log page headings and carriage controls.

  LINESIZE=*value*
  specifies the line size of SAS output. A setting of MAX is recommended to reduce log entry wrap-around.
- OBJECTSERVER and OBJECTSERVERPARMS invoke an Integrated Object Model server of the specified parameters.

PROTOCOL=bridge
identifies the network protocol used to communicate with the server. The valid value is *bridge*.

PORT=*port-number*
specifies the TCP port on which the server will listen for requests and that clients will use to connect to the server. The port-number value must be a unique four-digit number between 0 and 64K. The default port number is *8561*.

CLASSFACTORY=*factory-number*
specifies the type of IOM server to instantiate. The value 2887e7d7-4780-11d4-879f-00c04f38f0db identifies a SAS Metadata Server.

This is the minimum set of parameters needed to start a SAS Metadata Server. For information about additional parameters that invoke optional functionality, see the *SAS Intelligence Platform: System Administration Guide*.

You are now ready to register a repository.

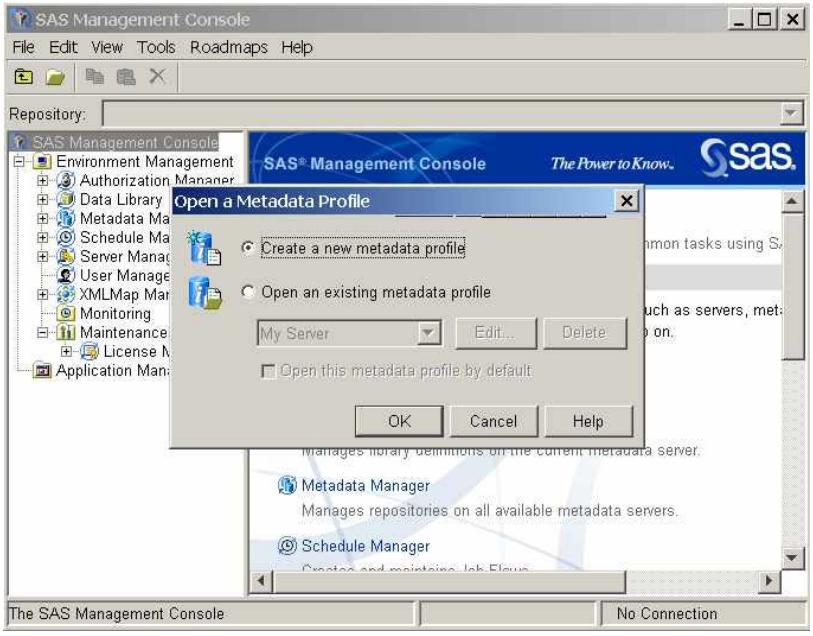
Registering a SAS Metadata Repository

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. SAS Management Console is a SAS Open Metadata Interface client that connects to the SAS Metadata Server and provides a user interface for managing SAS Metadata Repositories.

Note: You must be an *administrative user* of the server in order to be able to register a repository. You are an administrative user if you connect to the metadata server using the user ID that was used to start the server, or if your user ID is listed in an adminUsers.txt file in the metadata server directory. For more information about the adminUsers.txt file, see the *SAS Intelligence Platform: System Administration Guide*. △

To register a repository:

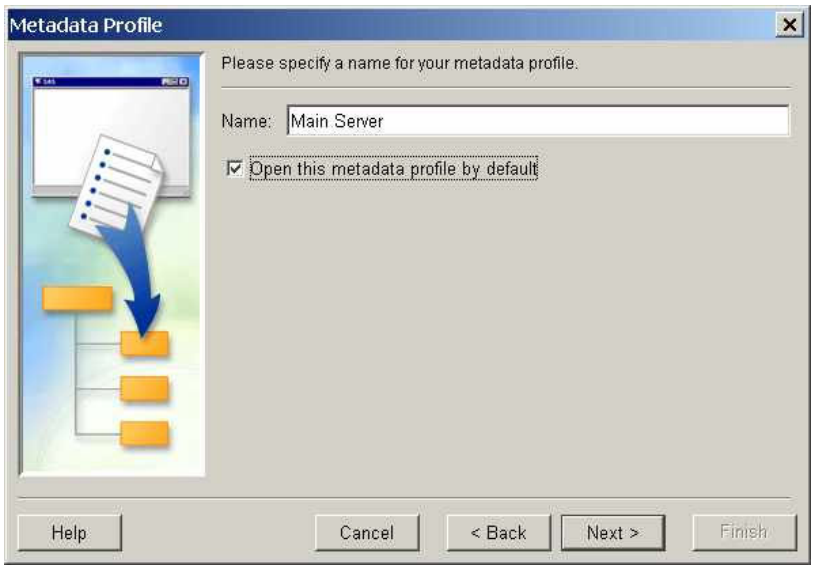
1 Select *Programs->SAS->SAS Management Console* from the Windows XP Start menu. The software displays a window similar to the following:



2 Before you can use SAS Management Console, you must connect to a running SAS 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.

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.

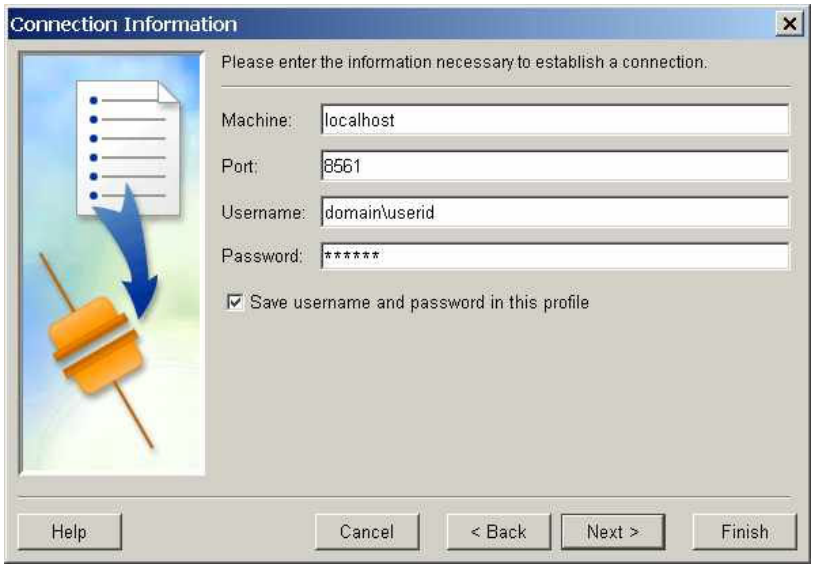


Note: If you do not select the **`Open this metadata profile by default`** check box, you will be prompted for server connection information each time you open SAS Management Console. Click Next to continue. △

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

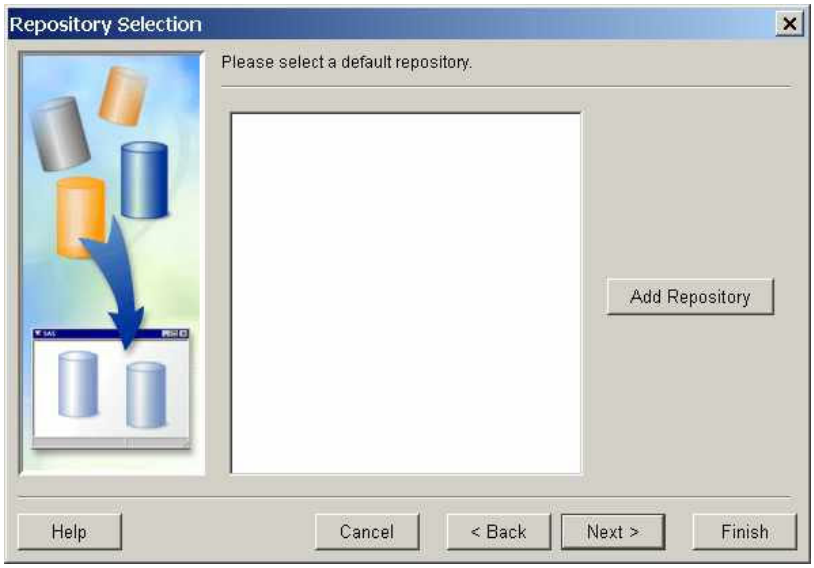
Machine	the host name or Internet Protocol address of the computer on which you started the SAS Metadata Server process.
Port	the port number that you specified in the startsrv.bat command file. This is the port to which the metadata server listens for requests.
Username	enter an authenticated user ID on the metadata server. The SAS Metadata Server supports several authentication options. In this example, we are relying on host authentication. Specify your user ID on the computer that is hosting the metadata server.
Password	enter the password associated with the user ID.

The following screen capture shows a typical set of connection properties.



Click Next to continue.

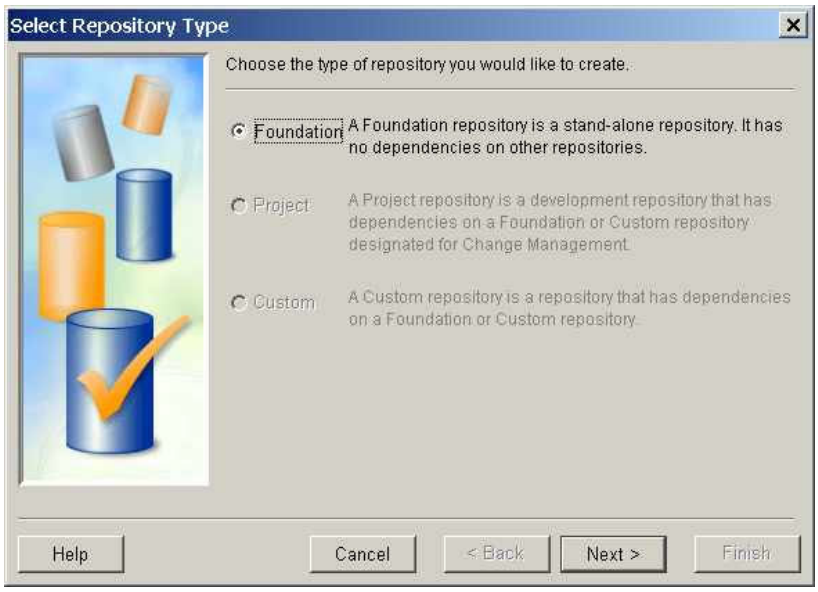
6 The Repository Selection window allows you to define a default repository. Select Add Repository to define a repository.



7 The software opens the Create 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 to a foundation or custom repository.
- 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. This repository will store metadata definitions that are shared by all repositories that you might define.



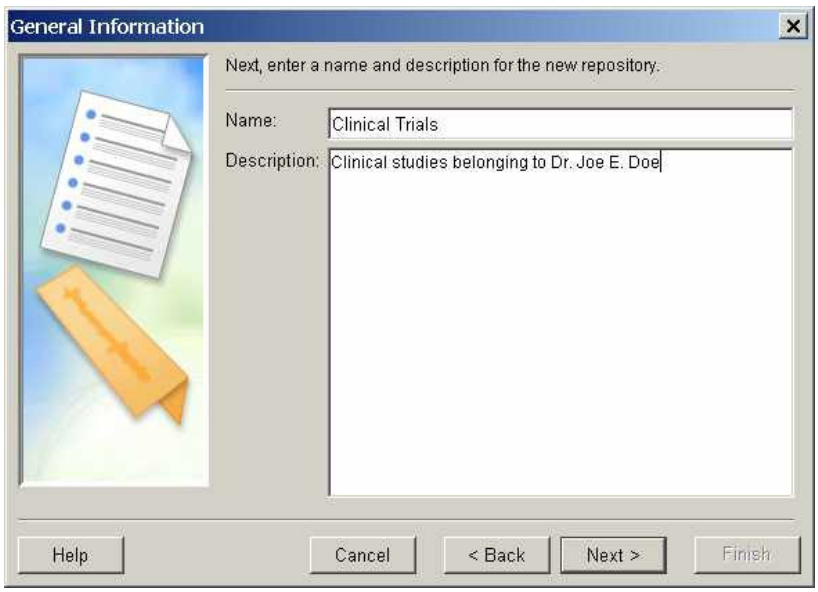
Click Next to continue.The software opens the General Information window.

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

Name is a unique name for the repository.

Description is an optional descriptive phrase for the repository.

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

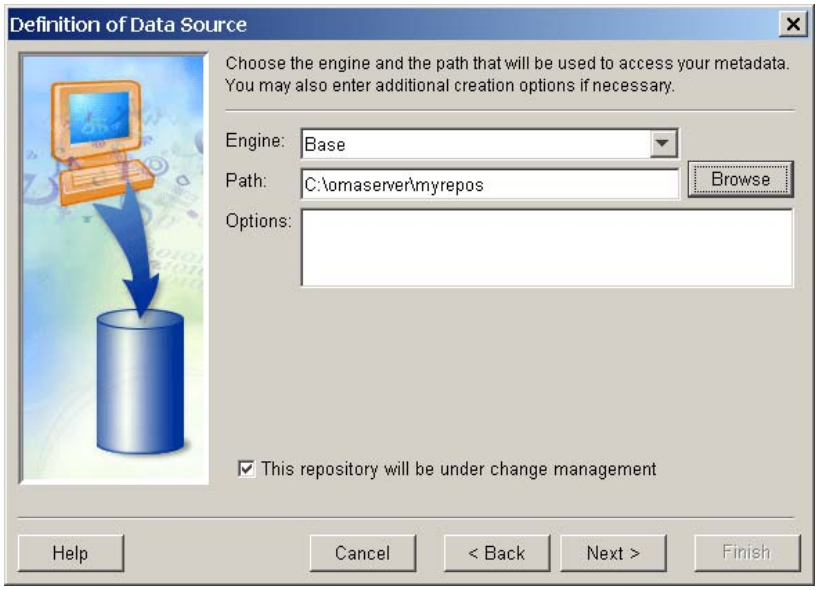


Click Next to continue. The software opens the Definition of Data Source window.

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

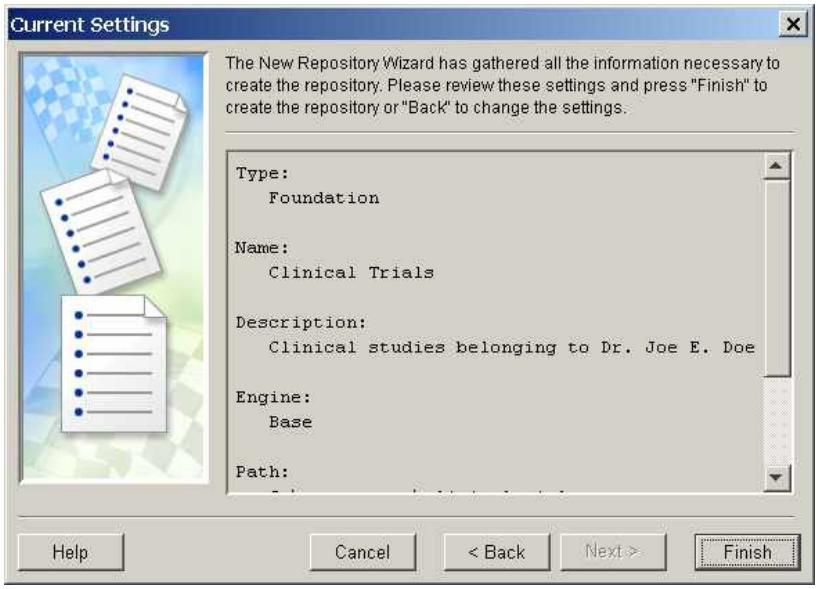
Engine	is the engine used to access this repository. Valid values are Base, DB2, and Oracle. *Base*, representing the SAS base engine, is the default engine if one is not specified.
Path	is the physical path to the repository directory. This is a required parameter when Base is selected as the engine. 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 or the *SAS Intelligence Platform: System Administration Guide* 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.



Click Next to continue.

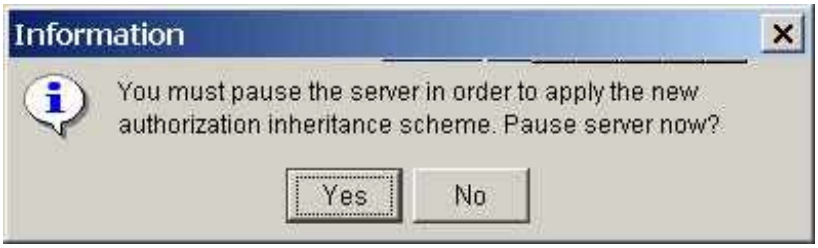
10 The wizard displays a Current Settings window for you to review the repository properties.



Click Finish to save the settings and create the repository.

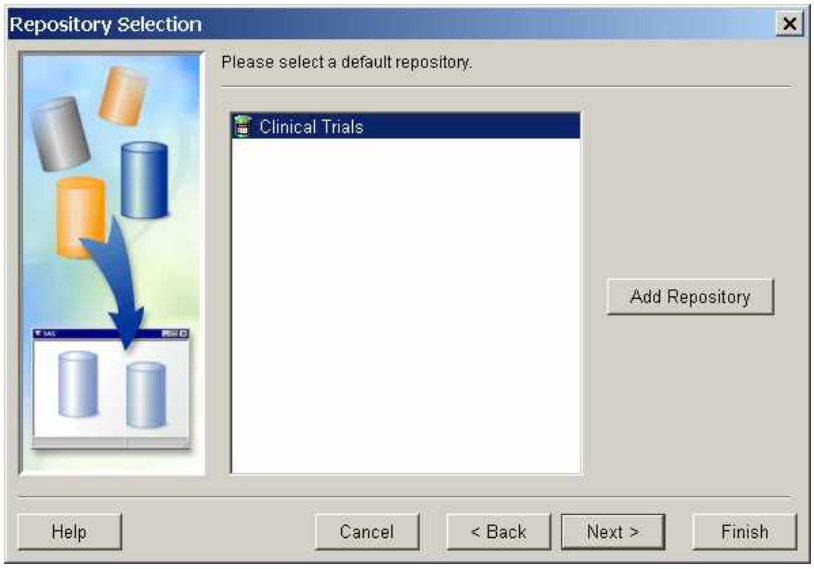
11 The software displays an informational message as it initializes the repository. Foundation repositories have a set of default resource templates and authorization metadata created in them. When the initialization is complete, it displays a window to inform you that the initialization was successful. 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.



Click Yes. *Warning:* 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.



You are now ready to create metadata.

PART 3

Working with Metadata

CHAPTER

6

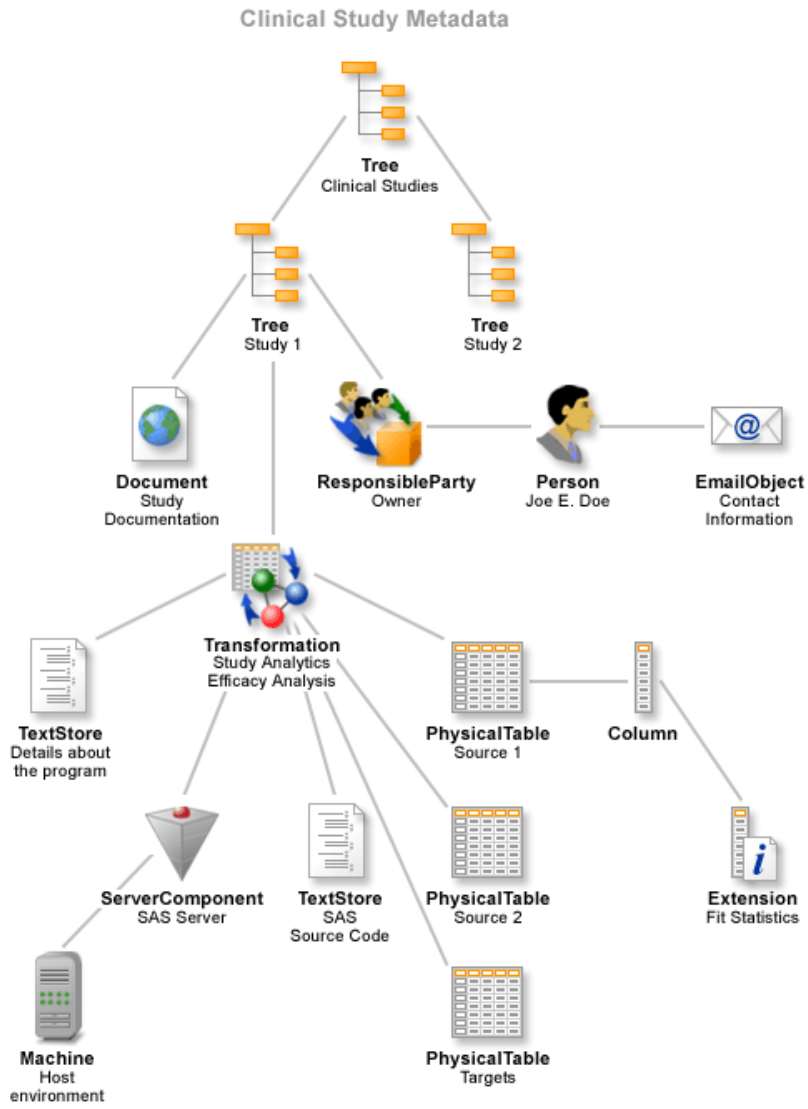
Adding Metadata Objects to the Repository

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 Metadata Types for Our Study" on page 11 to the Clinical Trials repository. As a refresher, we identified the need to define the following:

- PhysicalTable objects describing 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 discuss 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 relationship in the SAS Metadata Model is represented by two associations. That is, the association has a different 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 Tree has a Documents association to the Document (indicating the document is one of several possible documents that is associated with the 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 is

related). 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 any given association is described in the metadata type documentation in "Alphabetical Listing of SAS Namespace Metadata Types" in the *SAS Open Metadata Interface: Reference*. This listing is only available in online versions of the reference. Look for it in *SAS Help and Documentation* or *SAS OnlineDoc*.

Tasks

Metadata objects are added to SAS metadata repositories by using the AddMetadata method. We will create the objects and associations shown above in the following order:

1. Clinical Studies tree
2. Study 2 tree
3. Study 1 tree and business information objects
4. Table objects
5. 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 to a metadata repository, you must know the repository's metadata identifier. You must also know how to write an XML metadata property string. For more information, see "Determining the Repository ID" on page 43 and "Writing a Metadata Property String" on page 44.

Determining the Repository ID

Every object in a SAS metadata repository has a unique identifier. 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. A repository is no exception. A *repository identifier* is assigned to the repository when it is registered in the SAS Metadata Server repository manager, although this identifier is in the form REPOSMGRID.REPOSID.

To obtain a repository's unique identifier, you must issue the GetRepositories method. The following is an example of a GetRepositories method call that is formatted for the inMetadata parameter of the DoRequest method:

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

This is the output returned by the SAS Metadata Server:

```
<! --Using the DOREQUEST method. -->
<! -- Information received from server : -->
<GetRepositories>
 <Repositories>
   <Repository Id="A0000001.A5ROO9ZW" Name="Clinical Trials"
    Desc="Clinical studies belonging to Dr. Joe E. Doe" DefaultNS="SAS"/>
```

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

It shows that one repository (A5ROO9ZW) has been defined in the current repository manager (A0000001). A0000001.A5ROO9ZW is the value we will use to identify the Clinical Studies repository in our AddMetadata calls.

Writing a Metadata Property String

Methods that read or write a metadata object must submit a string of properties that describe that object to the SAS Metadata Server. This property string is passed to the metadata server in the inMetadata parameter of the method call.(When a method is submitted to the metadata server via the DoRequest method, the metadata property string is enclosed in <Metadata> tags.)

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 that is used to represent 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"/>
```

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 association name is a label that describes the relationship between the main element and the subelement.The association subelement identifies the partner metadata object in the relationship. In this example, Columns is the *association name* and Column is the *association subelement*. 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.

Creating the Clinical Studies Tree

The following AddMetadata method call creates a Tree object to represent the top-level node of the clinical study information. It is an example of a simple AddMetadata method call: it specifies to add a single object containing the attributes specified in the <Metadata> element to the repository identified in the <Reposid> element. The remaining parameters are required in every AddMetadata method call. The <NS> element specifies the namespace in which to execute the request. All requests that add application objects must be executed in the SAS namespace. The <Flags> element specifies the OMI_TRUSTED_CLIENT (268435456) flag. This flag is required in all method calls that add or update metadata in a repository. The <Options> element is supported to enable you to specify optional parameters. At this time, the AddMetadata method does not support any optional parameters.

```
<AddMetadata>
 <Metadata>
 <Tree
 Desc="Top level node for clinical study information."
 Name="Clinical Studies"
 TreeType="Clinical Study">
 </Tree>
 </Metadata>
 <Reposid>A0000001.A5ROO9ZW</Reposid>
 <NS>SAS</NS>
 <!-- OMI_TRUSTED_CLIENT flag -->
 <Flags>268435456</Flags>
 <Options/>
</AddMetadata>
```

This is the output returned by the SAS Metadata Server:

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<AddMetadata><Metadata><Tree Name="Clinical Studies" Desc="Top level
node for clinical study information." TreeType="Clinical Study"
Id="A5ROO9ZW.AK00000A"/></Metadata>
<Reposid>A0000001.A5ROO9ZW</Reposid><NS>SAS</NS>
 <!-- OMI_TRUSTED_CLIENT flag --><Flags>268435456</Flags>
<Options/></AddMetadata>
```

The output mirrors the input, except the attribute Id="A5ROO9ZW.AK00000A" 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.

Creating the Study 2 Tree

The following AddMetadata method call creates a Tree object to represent Study 2 as well as an association between the Clinical Studies Tree and the new Study 2 Tree. In addition to the metadata type and metadata type attributes, the metadata property string specifies an association name element 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. The ObjRef= attribute signals to the server that a relative reference to an existing object should be created. The absence of an identifier, or specifying Id= with no value in the association subelement, indicates to the server that a new object is to be created.

```
<AddMetadata>
 <Metadata>
  <Tree Desc="All information dealing with Study 2."
        Name="Study 2"
        TreeType="Clinical Study">
    <ParentTree>
       <Tree ObjRef="A5ROO9ZW.AK00000A"/>
    </ParentTree>
  </Tree>
 </Metadata>
 <Reposid>A0000001.A5ROO9ZW</Reposid>
 <NS>SAS</NS>
 <!-- OMI_TRUSTED_CLIENT flag -->
 <Flags>268435456</Flags>
 <Options/>
</AddMetadata>
```

The output returned by the server is as follows:

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<AddMetadata><Metadata>
<Tree Desc="All information dealing with Study 2." Name="Study 2"
TreeType="Clinical Study" Id="A5ROO9ZW.AK00000B">
<ParentTree><Tree ObjRef="A5ROO9ZW.AK00000A"/></ParentTree></Tree>
</Metadata><Reposid>A0000001.A5ROO9ZW</Reposid><NS>SAS</NS>
<!-- OMI_TRUSTED_CLIENT flag --><Flags>268435456</Flags>
<Options/></AddMetadata>
```

The Study 2 Tree has been assigned the ID value A5ROO9ZW.AK00000B. The association name element and subelement are stored as properties of the metadata object.

Creating the Study 1 Tree and Business Information Objects

The following AddMetadata method call adds objects for the Study 1 Tree and Business Information metadata objects identified in“Selecting Metadata Types for Our Study” on page 11 to the Clinical Studies repository. This example creates multiple objects in one method call by stacking multiple metadata property strings in the <Metadata> element. In addition, symbolic names are used as identifiers to enable

associations to be created between the objects until real identifiers can be assigned. The example has been segmented so that explanatory text can be provided.

The first metadata property string in the <Metadata> element defines the Study 1 Tree and creates an association between the Study 1 Tree and the Clinical Studies Tree:

```
<AddMetadata>
  <Metadata>
    <Tree
      Desc="All information dealing with Study 1."
      Id="$Study1"
      Name="Study 1"
      TreeType="Clinical Study">
      <ParentTree>
         <Tree ObjRef="A5ROO9ZW.AK00000B"/>
      </ParentTree>
    </Tree>
```

Id=, Desc, Name, and TreeType= are attributes of the Tree metadata type. 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. This alias enables the Business Information objects that describe the tree to be defined and associated with the Study 1 Tree in the same request that defines the Tree object. The ParentTree association name specifies that the Study 1 Tree is created as a subtree of the Clinical Studies tree, which is identified in the request by Objref=“A5ROO9ZW.AK00000B”.

The second metadata property string 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 attributes of the Document metadata type. 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 Trees association name, Tree subelement, and the symbolic name $Study1 create an association to the Study 1 object that was defined in the first metadata property string. The symbolic name will be replaced with a real Id= value at the successful completion of the AddMetadata request.

The following metadata property strings 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>
    </Metadata>
       <Reposid>A0000001.A5ROO9ZW</Reposid>
       <Ns>SAS</Ns>
       <!-- OMI_TRUSTED_CLIENT flag -->
       <Flags>268435456</Flags>
       <Options/>
    </AddMetadata>
```

Each of the property strings define and reference symbolic names to create associations between each other and the Study 1 Tree that was defined in the first metadata property string. Specifically:

- The symbolic name $ResponsibleParty1 is defined but is not referenced.
- The symbolic name $Study1 is used to create an Objects association between the ResponsibleParty object and the Study 1 Tree object.
- The symbolic name $Person1 is used to create a Persons association between the ResponsibleParty object and a Person object, which has not yet been defined.
- The symbolic name $Person1 is used to create a Persons association between the Email object and the Person object.

The following is the output returned by the SAS Metadata Server.

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<AddMetadata>
<Metadata>
<Tree Desc="All information dealing with Study 1." Id="A5ROO9ZW.AK00000E"
Name="Study 1" TreeType="Clinical Study"><ParentTree>
<Tree ObjRef="A5ROO9ZW.AK00000A"/></ParentTree></Tree>
<Document Desc="Documentation for clinical study." Name="Specification For Clinical
Study 1" URI="http://webserver.xyz.com/doc/Study.html" URIType="URL"
Id="A5ROO9ZW.AN000003"><Trees><Tree ObjRef="A5ROO9ZW.AK00000E"/>
</Trees></Document><ResponsibleParty Desc="Owner of clinical studies."
Id="A5ROO9ZW.AO000003" Name="Owner of Study 1" Role="OWNER">
<Objects><Tree ObjRef="A5ROO9ZW.AK00000E"/></Objects>
<Persons><Person ObjRef="A5ROO9ZW.AP000003"></Persons></ResponsibleParty>
<Person Desc="Manager of clinical studies." Id="A5ROO9ZW.AP000003"
Name="Joe E. Doe" Title="Manager of Clinical Studies"/>
<Email Address="J.E.Doe@xyz.com" Desc="Primary e-mail address." Name="e-mail address
```

```
for Joe E. Doe" Id="A5ROO9ZW.AQ000003"><Persons>
<Person ObjRef="A5ROO9ZW.AP000003"></Persons></Email></Metadata>
<Reposid>A0000001.A5ROO9ZW</Reposid><NS>SAS</NS><!-- OMI_TRUSTED_CLIENT flag -->
<Flags>268435456</Flags><Options/></AddMetadata>
```

The metadata identifiers assigned to the new objects are highlighted.
Next, we will create metadata objects describing the Study 1 tables.

Creating Objects for the Study 1 Tables

In "Deciding What Information Should Be Stored" on page 9, we described the need to create table objects describing two input tables named "Patient Information" and "Visit Information" and an output table named "Study 1 Output". The following AddMetadata method call creates these objects. As in the previous example, the objects are defined by stacking multiple metadata property strings in the method's <Metadata> element and symbolic names are used to define associations between the objects. Also, as in the preceding example, the example is segmented to allow explanatory text to be interspersed among the sample code.

The first part of the AddMetadata example contains the metadata property strings that define the properties for the Patient Information table and its columns:

```
<AddMetadata>
   <Metadata>
    <PhysicalTable
      Desc="Information describing an individual patient."
      Id="$PatientInformation"
      MemberType="DATA"
      Name="Patient Information"
      SASTableName="Patient_Information"
      TableName="Patient_Information">
      <Trees>
         <Tree ObjRef="A5ROO9ZW.AK00000E"/>
      </Trees>
    </PhysicalTable>
    <Column
      Name="Patient ID"
      Desc="Patient Information"
      ColumnName="Patient_ID"
      SASColumnName="Patient_ID"
      ColumnType="12"
      SASColumnType="C"
      ColumnLength="32"
      SASColumnLength="32"
      SASFormat="$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 metadata property strings, note:

- The symbolic name $PatientInformation is defined in the PhysicalTable object definition and is later referenced in the Column object definitions to create a Table association between the column objects and the table object.
- The PhysicalTable definition creates a Trees association to the Study 1 Tree object.
- The Column object definitions specify attributes both as they are defined in a DBMS and as they are defined by SAS software.
- The Column object definitions 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="$PatientInformation"/>
    </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 for a complete list of supported integer and SQL type values. This and other metadata type descriptions are found in the "Alphabetical Listing of SAS Namespace Metadata Types," which is available only in online versions of the *SAS Open Metadata Interface: Reference*.

- The symbolic name $WeightInKgColumn is assigned to the Weight in Kilograms column and is not yet referenced.

The following metadata property string defines an Extension object for the Patient Information table's 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 object.

The following metadata property strings define the properties for the Visit Information table and its columns:

```
<PhysicalTable
   Desc="Information describing a patient visit."
   Id="$PatientVisit"
   MemberType="DATA"
   Name="Visit Information"
   SASTableName="Patient_Visit"
   TableName="Patient_Visit">
   <Trees>
      <Tree ObjRef="A5ROO9ZW.AK00000E"/>
   </Trees>
</PhysicalTable>
<Column
   Name="Visit Name"
   Desc="Visit Name"
   ColumnName="Visit_Name"
   SASColumnName="Visit_Name"
   ColumnType="12"
   SASColumnType="C"
   ColumnLength="32"
   SASColumnLength="32"
   SASFormat="$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 these metadata property strings:

- The symbolic name $PatientVisit is assigned to the Visit Information table object and is used to create a Table association between each of the columns and the table.
- The PhysicalTable definition creates a Trees association to the Study 1 Tree object.

Finally, the following metadata property strings 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="A5ROO9ZW.AK00000E"/>
    </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.A5ROO9ZW</Reposid>
<Ns>SAS</Ns>
<!-- OMI_TRUSTED_CLIENT flag -->
<Flags>268435456</Flags>
```

```
   <Options/>
</AddMetadata>
```

- The symbolic name $Study1Output is assigned to the Study 1 Output table object and is used to create a Table association between each of the columns and the table.
- The PhysicalTable definition creates a Trees association to the Study 1 Tree object.

The following is the output returned by the SAS Metadata Server.

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<AddMetadata>
<Metadata>
<PhysicalTable Id="A5ROO9ZW.AR000004" Desc="Information describing an
individual patient." MemberType="DATA" Name="Patient Information"
SASTableName="Patient_Information" TableName="Patient_Information"><Trees>
<Tree ObjRef="A5ROO9ZW.AK00000E"/></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="$32."
Id="A5ROO9ZW.AS00000M"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000004"/>
</Table></Column>
<Column Name="Initials" Desc="Patient Initials" ColumnName="Initials"
SASColumnName="Initials" ColumnType="12" SASColumnType="C" ColumnLength="3"
SASColumnLength="3" SASFormat="$Char3." SASInformat="$3."
Id="A5ROO9ZW.AS00000N"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000004"/>
</Table></Column>
<Column Name="Sex" Desc="Sex of Patient" ColumnName="Sex" SASColumnName="Sex"
ColumnType="12" SASColumnType="C" ColumnLength="1" SASColumnLength="1"
SASFormat="$Char1." SASInformat="$1." Id="A5ROO9ZW.AS00000O"><Table>
<PhysicalTable ObjRef="A5ROO9ZW.AR000004"/></Table></Column>
<Column Name="Date Of Birth" Desc="Date Of Birth" ColumnName="Date_Of_Birth"
SASColumnName="Date_Of_Birth" ColumnType="91" SASColumnType="N" ColumnLength="9"
SASColumnLength="9" SASFormat="date9." SASInformat="date9."
Id="A5ROO9ZW.AS00000P"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000004"/>
</Table></Column>
<Column Name="Sponsor Patient ID" Desc="Sponsor Patient Information"
ColumnName="Sponsor_Patient_ID" SASColumnName="Sponsor_Patient_ID" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32." Id="A5ROO9ZW.AS00000Q"><Table><PhysicalTable
ObjRef="A5ROO9ZW.AR000004"/></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" Id="A5ROO9ZW.AS00000R"><Table>
<PhysicalTable ObjRef="A5ROO9ZW.AR000004"/></Table></Column>
<Column Id="A5ROO9ZW.AS00000S" 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="A5ROO9ZW.AR000004"/></Table></Column>
<Extension Desc="Algorithm for column." Name="Algorithm" Value="Weight_In_Lb\2.2"
Id="A5ROO9ZW.AB0000H8"><OwningObject><Column ObjRef="A5ROO9ZW.AS00000S"/>
</OwningObject></Extension>
<PhysicalTable Desc="Information describing a patient visit."
Id="A5ROO9ZW.AR000005" MemberType="DATA" Name="Visit Information"
```

```
SASTableName="Patient_Visit" TableName="Patient_Visit"><Trees><Tree
ObjRef="A5ROO9ZW.AK00000E"/></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="$32."
Id="A5ROO9ZW.AS00000T"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000005"/>
</Table></Column>
<Column Name="Sponsor Patient ID" Desc="Sponsor Patient ID"
ColumnName="Sponsor_Patient_ID" SASColumnName="Sponsor_Patient_ID" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32." Id="A5ROO9ZW.AS00000U"><Table><PhysicalTable
ObjRef="A5ROO9ZW.AR000005"/></Table></Column>
<Column Name="Patient ID" Desc="Patient ID" ColumnName="Patient_ID"
SASColumnName="Patient_ID" ColumnType="12" SASColumnType="C" ColumnLength="32"
SASColumnLength="32" SASFormat="$Char32." SASInformat="$32."
Id="A5ROO9ZW.AS00000V"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000005"/>
</Table></Column>
<Column Name="Systolic Blood Pressure" Desc="Systolic Blood Pressure"
ColumnName="Systolic_Blood_Pressure" SASColumnName="Systolic_Blood_Pressure"
ColumnType="4" SASColumnType="N" ColumnLength="4" SASColumnLength="4"
SASFormat="4.0" SASInformat="4.0" Id="A5ROO9ZW.AS00000W"><Table>
<PhysicalTable ObjRef="A5ROO9ZW.AR000005"/></Table></Column>
<Column Name="Diastolic Blood Pressure" Desc="Diastolic Blood Pressure"
ColumnName="Diastolic_Blood_Pressure" SASColumnName="Diastolic_Blood_Pressure"
ColumnType="4"  SASColumnType="N" ColumnLength="4" SASColumnLength="4"
SASFormat="4.0" SASInformat="4.0" Id="A5ROO9ZW.AS00000X"><Table>
<PhysicalTable ObjRef="A5ROO9ZW.AR000005"/></Table></Column>
<Column Name="Visit Number" Desc="Visit Number" ColumnName="Visit_Number"
SASColumnName="Visit_Number" ColumnType="4" SASColumnType="N" ColumnLength="4"
SASColumnLength="4" SASFormat="4.0" SASInformat="4.0" Id="A5ROO9ZW.AS00000Y">
<Table><PhysicalTable ObjRef="A5ROO9ZW.AR000005"/></Table></Column>
<Column Name="Occurrence Number" Desc="Occurrence Number"
ColumnName="Occurrence_Number" SASColumnName="Occurrence_Number" ColumnType="4"
SASColumnType="N" ColumnLength="4" SASColumnLength="4" SASFormat="4.0"
SASInformat="4.0" Id="A5ROO9ZW.AS00000Z"><Table><PhysicalTable
ObjRef="A5ROO9ZW.AR000005"/></Table></Column>
<PhysicalTable Desc="Information output from Study 1." Id="A5ROO9ZW.AR000006"
MemberType="DATA" Name="Study 1 Output" SASTableName="Study_1_Output"
TableName="Study_1_Output"><Trees>
<Tree ObjRef="A5ROO9ZW.AK00000E"/></Trees></PhysicalTable>
<Column Name="Patient ID" Desc="Patient Information" ColumnName="Patient_ID"
SASColumnName="Patient_ID" ColumnType="12" SASColumnType="C" ColumnLength="32"
SASColumnLength="32" SASFormat="$Char32." SASInformat="$32."
Id="A5ROO9ZW.AS000010"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000006"/>
</Table></Column>
<Column Name="Visit Name" Desc="Visit Name" ColumnName="Visit_Name"
SASColumnName="Visit_Name" ColumnType="12" SASColumnType="C" ColumnLength="32"
SASColumnLength="32" SASFormat="$Char32." SASInformat="$32."
Id="A5ROO9ZW.AS000011"><Table><PhysicalTable ObjRef="A5ROO9ZW.AR000006"/>
</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" Id="A5ROO9ZW.AS000012"><Table><PhysicalTable
```

```
ObjRef="A5ROO9ZW.AR000006"/></Table></Column></Metadata>
<Reposid>A0000001.A5ROO9ZW</Reposid><NS>SAS</NS><!-- OMI_TRUSTED_CLIENT flag -->
<Flags>268435456</Flags><Options/></AddMetadata>
```

The metadata identifiers assigned to the new objects are highlighted.

Now we will create the TransformationActivity, TextStore, and ServerComponent objects.

Creating the TransformationActivity, TextStore, and ServerComponent Objects

The following AddMetadata method call creates the TransformationActivity, TextStore, and ServerComponent objects described in “Selecting Metadata Types for Our Study” on page 11.

The first metadata property string in the <Metadata> element defines the TransformationActivity object:

```
<AddMetadata>
  <Metadata>
    <TransformationActivity
      Desc="Program for calculating drug efficacy."
      Id="$TransformationActivity1"
      Name="Program For Calculating Drug Efficacy">
      <TransformationSources>
         <PhysicalTable ObjRef="A5ROO9ZW.AR000004"/>
         <PhysicalTable ObjRef="A5ROO9ZW.AR000005"/>
      </TransformationSources>
      <TransformationTargets>
         <PhysicalTable ObjRef="A5ROO9ZW.AR000006"/>
      </TransformationTargets>
      <Trees>
         <Tree ObjRef="A5ROO9ZW.AK00000E"/>
       </Trees>
    </TransformationActivity>
```

In the metadata property string:

- 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 “Creating Objects for the Study 1 Tables” on page 49. 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 metadata property strings 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>
```

In each string, the 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.

Finally, the following metadata property string defines the ServerComponent object, which describes the software that performs the transformation, and the elements which complete 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.A5ROO9ZW</Reposid>
  <Ns>SAS</Ns>
  <!-- OMI_TRUSTED_CLIENT flag -->
  <Flags>268435456</Flags>
  <Options/>
```

```
</AddMetadata>
```

In the metadata property string:

- 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 following is the output returned by the SAS Metadata Server:

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<AddMetadata>
<Metadata>
<TransformationActivity Desc="Program for calculating drug efficacy."
Id="A5ROO9ZW.AM000002" Name="Program For Calculating Drug Efficacy">
<TransformationSources><PhysicalTable ObjRef="A5ROO9ZW.AR000004"/>
<PhysicalTable ObjRef="A5ROO9ZW.AR000005"/></TransformationSources>
<TransformationTargets><PhysicalTable ObjRef="A5ROO9ZW.AR000006"/>
</TransformationTargets>
<Trees><Tree ObjRef="A5ROO9ZW.AK00000E"/></Tree></TransformationActivity>
<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"
Id="A5ROO9ZW.AF00009D"><Objects><TransformationActivity ObjRef="A5ROO9ZW.AM000002"/>
</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" Id="A5ROO9ZW.AF00009E">
<AssociatedTransformation><TransformationActivity ObjRef="A5ROO9ZW.AM000002"/>
<TransformationActivity ObjRef="A5ROO9ZW.AM000002"/></TextStore>
<ServerComponent Desc="SAS Software" IsLicensed="1" Major="9" Minor="0"
Name="SAS Software on olive.us.xyz.com" ProductName="The SAS System"
SoftwareVersion="9.1.3" Vendor="SAS Institute" Id="A5ROO9ZW.AT000001">
<ComputeTasks><TransformationActivity ObjRef="A5ROO9ZW.AM000002"/>
</ComputeTasks></ServerComponent></Metadata><Reposid>A0000001.A5ROO9ZW</Reposid>
<NS>SAS</NS><!-- OMI_TRUSTED_CLIENT flag --><Flags>268435456</Flags><Options/>
</AddMetadata>
```

The repository has been populated with metadata objects.

For more information about adding objects, see the AddMetadata method in "Methods for Reading and Writing Metadata (IOMI Class)" in the *SAS Open Metadata*

Interface: Reference. Also see "Adding Metadata Objects" in the *SAS Open Metadata Interface: User's Guide*.

We are now ready to query the repository.

CHAPTER 7

Querying the Repository

Overview of Querying the Repository

Now that we have objects in our Clinical Studies 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 GetMetadataObjects methods for performing queries.

- The GetMetadataObjects method gets all metadata objects of a specified type when passed the repository and type.
- The GetMetadata method gets a specified metadata object from a repository.

This section contains method calls that show you how to

- list objects of a given metadata type
- request properties for a specific object
- use a search string to filter an object 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 Open Metadata Interface: User's Guide*.

Listing All Objects of a Given Metadata Type

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

```
<GetMetadataObjects>
  <Reposid>A0000001.A5ROO9ZW</Reposid>
  <Type>Tree</Type>
  <Objects/>
  <NS>SAS</NS>
  <Flags>0</Flags>
```

```
  <Options/>
</GetMetadataObjects>
```

The <Reposid> element identifies the repository to query. The <Type> element specifies the requested metadata type.

The following is the output returned by the SAS Metadata Server:

```
<GetMetadataObjects>
  <Reposid>A0000001.A5ROO9ZW</Reposid>
  <Type>Tree</Type>
  <Objects>
  <Tree Id="A5ROO9ZW.AK00000A" Name="Clinical Studies"/>
  <Tree Id="A5ROO9ZW.AK00000B" Name="Study 2"/>
  <Tree Id="A5ROO9ZW.AK00000E" Name="Study 1"/>
  </Objects>
  <NS>SAS</NS>
  <Flags>0</Flags>
  <Options/>
</GetMetadataObjects>
```

The metadata objects that were found are listed in the <Objects> element.

The GetMetadataObjects method returns general, identifying information about an object (the Id= and Name= attributes) by default. 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.

For more information, see the GetMetadataObjects method in "Methods for Reading and Writing Metadata (IOMI Class)" in the *SAS Open Metadata Interface: Reference* and "Querying All Metadata of a Specified Type" in the *SAS Open Metadata Interface: User's Guide*.

Requesting Properties for Specific Objects

To retrieve properties for a specific metadata 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
- a 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.
- 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 the documentation for the GetMetadata method in "Methods for Reading and Writing Metadata (IOMI Class)" in the *SAS 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> element of a GetMetadata method call. The OMI_TEMPLATE (4) flag must also 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. For detailed information about the use of templates, see first "Querying Specific Metadata Objects" and then "Using Templates" in the *SAS Open Metadata Interface: User's Guide*.

The following is an example of a GetMetadata method 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, template components are bolded.

```
<GetMetadata>
  <Metadata>
    <Tree Id="A5ROO9ZW.AK00000E">
       <Members/>
       <ResponsibleParties/>
    </Tree>
  </Metadata>
  <NS>SAS</NS>
  <!-- OMI_TEMPLATES flag-->
  <Flags>4</Flags>
  <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.

The following is the output returned by the SAS Metadata Server:

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<GetMetadata><Metadata><Tree Id="A5ROO9ZW.AK00000E" Desc="All information
dealing with Study 1." Name="Study 1" TreeType="Clinical Study"><Members>
<Document Id="A5ROO9ZW.AN000003" Desc="Documentation for clinical study."
Name="Specification For Clinical Study 1"
URI="http://webserver.xyz.com/doc/Study.html" URIType="URL"/>
<PhysicalTable Id="A5ROO9ZW.AR000004" Desc="Information describing
an individual patient." MemberType="DATA" Name="Patient Information"
SASTableName="Patient_Information" TableName="Patient_Information"><Columns>
<Column Id="A5ROO9ZW.AS00000M" Name="Patient ID" Desc="Patient Information"
ColumnName="Patient_ID" SASColumnName="Patient_ID" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000N" Name="Initials" Desc="Patient Initials"
ColumnName="Initials" SASColumnName="Initials" ColumnType="12"
SASColumnType="C" ColumnLength="3" SASColumnLength="3" SASFormat="$Char3."
SASInformat="$3."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000O" Name="Sex" Desc="Sex of Patient" ColumnName="Sex"
SASColumnName="Sex" ColumnType="12" SASColumnType="C" ColumnLength="1"
SASColumnLength="1" SASFormat="$Char1." SASInformat="$1."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000P" Name="Date Of Birth" Desc="Date Of Birth"
ColumnName="Date_Of_Birth" SASColumnName="Date_Of_Birth" ColumnType="91"
SASColumnType="N" ColumnLength="9" SASColumnLength="9" SASFormat="date9."
SASInformat="date9."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000Q" Name="Sponsor Patient ID" Desc="Sponsor Patient
Information" ColumnName="Sponsor_Patient_ID" SASColumnName="Sponsor_Patient_ID"
ColumnType="12" SASColumnType="C" ColumnLength="32" SASColumnLength="32"
SASFormat="$Char32." SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000R" Name="Weight In Pounds" Desc="Patient Weight
In Pounds" ColumnName="Weight_In_Lb" SASColumnName="Weight_In_Lb"
ColumnType="6" SASColumnType="N" ColumnLength="6" SASColumnLength="6"
SASFormat="6.2" SASInformat="6.2"><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000S" Name="Weight In Kilograms" Desc="Patient Weight
In Kilograms" ColumnName="Weight_In_Kg" SASColumnName="Weight_In_Kg"
ColumnType="6" SASColumnType="N" ColumnLength="6" SASColumnLength="6"
SASFormat="6.2" SASInformat="6.2"><Extensions/><Extension Id="A5ROO9ZW.AB0000H8"
Desc="Algorithm for column." Name="Algorithm" Value="Weight_In_Lb\2.2"/>
</Extensions></Column></Columns></PhysicalTable>
<PhysicalTable Id="A5ROO9ZW.AR000005" Desc="Information describing a patient
visit." MemberType="DATA" Name="Visit Information" SASTableName="Patient_Visit"
TableName="Patient_Visit"><Columns><Column Id="A5ROO9ZW.AS00000T"
Name="Visit Name" Desc="Visit Name" ColumnName="Visit_Name"
SASColumnName="Visit_Name" ColumnType="12" SASColumnType="C" ColumnLength="32"
SASColumnLength="32" SASFormat="$Char32." SASInformat="$32."><Extensions/>
</Column><Column Id="A5ROO9ZW.AS00000U" 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="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000V" Name="Patient ID" Desc="Patient ID"
ColumnName="Patient_ID" SASColumnName="Patient_ID" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000W" Name="Systolic Blood Pressure" Desc="Systolic
Blood Pressure" ColumnName="Systolic_Blood_Pressure" SASColumnName=
"Systolic_Blood_Pressure" ColumnType="4" SASColumnType="N" ColumnLength="4"
SASColumnLength="4" SASFormat="4.0" SASInformat="4.0"><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000X" Name="Diastolic Blood Pressure" Desc="Diastolic
Blood Pressure" ColumnName="Diastolic_Blood_Pressure"
SASColumnName="Diastolic_Blood_Pressure" ColumnType="4" SASColumnType="N"
ColumnLength="4" SASColumnLength="4" SASFormat="4.0"
SASInformat="4.0"><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000Y" Name="Visit Number" Desc="Visit Number"
ColumnName="Visit_Number" SASColumnName="Visit_Number" ColumnType="4"
SASColumnType="N" ColumnLength="4" SASColumnLength="4" SASFormat="4.0"
SASInformat="4.0"><Extensions/></Column>
<Column Id="A5ROO9ZW.AS00000Z" Name="Occurrence Number" Desc="Occurrence Number"
ColumnName="Occurrence_Number" SASColumnName="Occurrence_Number" ColumnType="4"
SASColumnType="N" ColumnLength="4" SASColumnLength="4" SASFormat="4.0"
SASInformat="4.0"><Extensions/></Column></Columns></PhysicalTable>
<PhysicalTable Id="A5ROO9ZW.AR000006" Desc="Information output from Study 1."
MemberType="DATA" Name="Study 1 Output" SASTableName="Study_1_Output"
TableName="Study_1_Output"><Columns>
<Column Id="A5ROO9ZW.AS000010" Name="Patient ID" Desc="Patient Information"
ColumnName="Patient_ID" SASColumnName="Patient_ID" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS000011" Name="Visit Name" Desc="Visit Name"
ColumnName="Visit_Name" SASColumnName="Visit_Name" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS000012" 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">
<Extensions/></Column></Columns></PhysicalTable>
<TransformationActivity Id="A5ROO9ZW.AM000002" Desc="Program for calculating
drug efficacy." Name="Program For Calculating Drug Efficacy"><ComputeLocations>
<ServerComponent Id="A5ROO9ZW.AT000001" Desc="SAS Software" IsLicensed="1"
Major="9" Minor="0" Name="SAS Software on olive.us.xyz.com" ProductName="The SAS
System" SoftwareVersion="9.1.3" Vendor="SAS Institute"/></ComputeLocations>
<Notes><TextStore Id="A5ROO9ZW.AF00009D" Desc="Details About The Program For
Calculating Normalized SBPW Coefficient." Name="Details About The Program For
Calculating Normalized SBPW Co" 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"/>
</Notes><SourceCode>
<TextStore Id="A5ROO9ZW.AF00009E" 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 procsort
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"/></SourceCode>
<TransformationTargets><PhysicalTable Id="A5ROO9ZW.AR000006" Desc="Information
output from Study 1." MemberType="DATA" Name="Study 1 Output"
SASTableName="Study_1_Output" TableName="Study_1_Output"><Columns>
<Column Id="A5ROO9ZW.AS000010" Name="Patient ID" Desc="Patient Information"
ColumnName="Patient_ID" SASColumnName="Patient_ID" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS000011" Name="Visit Name" Desc="Visit Name"
ColumnName="Visit_Name" SASColumnName="Visit_Name" ColumnType="12"
SASColumnType="C" ColumnLength="32" SASColumnLength="32" SASFormat="$Char32."
SASInformat="$32."><Extensions/></Column>
<Column Id="A5ROO9ZW.AS000012" 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"><Extensions/></Column>
</Columns></PhysicalTable></TransformationTargets></TransformationActivity>
</Members><ResponsibleParties><ResponsibleParty Id="A5ROO9ZW.AO000003"
Desc="Owner of clinical studies." Name="Owner of Study 1" Role="OWNER">
<Persons><Person Id="A5ROO9ZW.AP000003" Desc="Manager of clinical studies."
Name="Joe E. Doe" Title="Manager of Clinical Studies"><EmailAddresses>
<Email Id="A5ROO9ZW.AQ000003" Address="J.E.Doe@xyz.com" Desc="Primary e-mail
address." Name="e-mail address for Joe E. Doe"/></EmailAddresses></Person>
</Persons></ResponsibleParty></ResponsibleParties></Tree>
</Metadata><NS>SAS</NS><Flags>4</Flags><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/><TransformationSources/><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>
```

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 <Options> element of the GetMetadataObjects call. The <XMLSelect> element enables you to pass a search string to the server. When you specify the <XMLSelect> element, you must also set the OMI_XMLSELECT (128) flag so that the server knows to look for the element in the <Options> element.

The <XMLSelect> search syntax is described in "Filtering a GetMetadataObjects Request" in " Querying All Metadata of a Specified Type" in the *SAS 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 method 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.A5ROO9ZW</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.

The following is an example of the output returned by the SAS Metadata Server:

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  -->
<GetMetadataObjects><Reposid>A0000001.A5ROO9ZW</Reposid><Type>TextStore</Type>
<Objects><TextStore Id="A5ROO9ZW.AF00009E" Name="Source code for calculating
Normalized SBPW Coefficient"/></Objects><NS>SAS</NS><Flags>128</Flags>
<Options><XMLSelect search="@TextRole = 'SOURCE'"/></Options>
</GetMetadataObjects>
```

CHAPTER

8

Updating Metadata Objects in the Repository

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 A5ROO9ZW.AF00009E with the text supplied in the method call.

```
<UpdateMetadata>
   <Metadata>
     <TextStore
       Id="A5ROO9ZW.AF00009E"
       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 method output exactly mirrors the method input, so it is not provided here.

The UpdateMetadata method can also be used to add or modify associations between objects. For more information, see UpdateMetadata in “Methods for Reading and Writing Metadata (IOMI Class)” in the *SAS Open Metadata Interface: Reference*. Also see “Updating Metadata Objects” in the *SAS Open Metadata Interface: User’s Guide*.

CHAPTER

9

Deleting Metadata Objects in the Repository

Deleting Metadata Objects in the Repository

You delete metadata objects 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="A5ROO9ZW.AF00009D"/>
   </Metadata>
   <NS>SAS</NS>
   <!--OMI_TRUSTED_CLIENT + OMI_RETURN_LIST flags-->
   <Flags>268436480</Flags>
   <Options/>
</DeleteMetadata>
```

Here is the output returned by the SAS Metadata Server:

```
<!-- Using the DOREQUEST method. -->
<!-- Information received from server :  --><DeleteMetadata>
<Metadata><TextStore Id="A5ROO9ZW.AF00009D"/></Metadata></Metadata>
<NS>SAS</NS><!--OMI_TRUSTED_CLIENT + OMI_RETURN_LIST flags-->
<Flags>268436480</Flags><Options/></DeleteMetadata>
```

The <Metadata> element identifies the metadata object to be deleted. In the <Flags> element, we have set two flags. The OMI_TRUSTED_CLIENT (268435456) flag is required when you update, add, or delete metadata in a repository. We have opted to include the OMI_RETURN_LIST (1024) flag in order to verify the operation as well. The OMI_RETURN_LIST flag returns a list of deleted object IDs as well as the IDs of any dependent objects that might have been deleted. A dependent object is an object that has a 1:1 cardinality to the deleted object, and therefore cannot exist without the deleted object, so the server will delete it as well. TextStore object A5ROO9ZW.AF00009D has no dependent objects; therefore, the method output mirrors the method input. For more information, see DeleteMetadata in "Methods for Reading and Writing Metadata (IOMI Class)" in the *SAS Open Metadata Interface: Reference*. Also see "Deleting Metadata Objects" in the *SAS Open Metadata Interface: User's Guide*.

CHAPTER 10

Stopping the SAS Metadata Server

Overview of Stopping the SAS Metadata Server

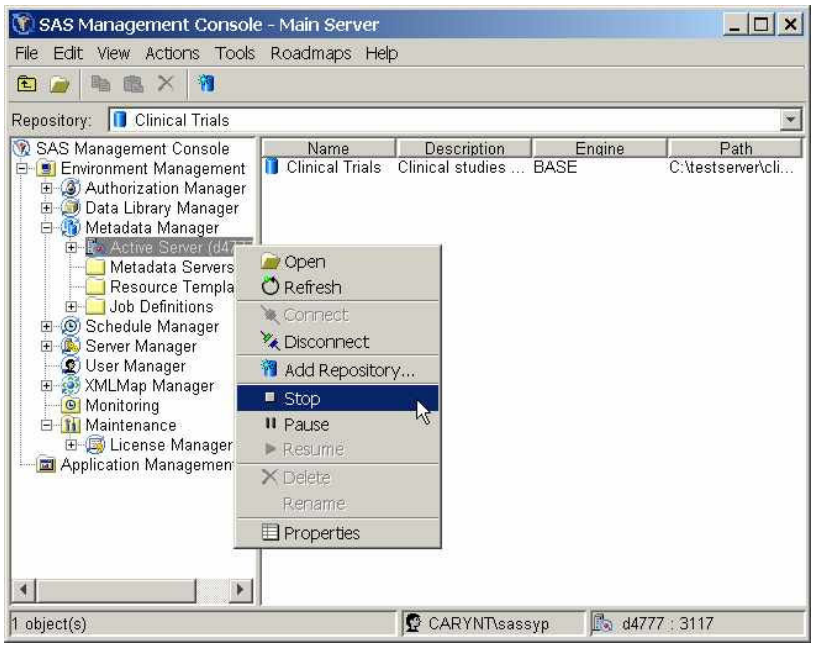
Note: You must be an administrative user of a SAS Metadata Server in order to stop the metadata server. You are an administrative user if you connect to the metadata server using the same user ID that was used to start the metadata server, or if your user ID is listed in an adminUsers.txt file in the directory defined for the metadata server. For more information about the adminUsers.txt file, see the *SAS Intelligence Platform: System Administration Guide*. △

The steps outlined in this section are intended to be used to stop a personal SAS Metadata Server. Do not use them to stop an enterprise SAS Metadata Server. An enterprise SAS Metadata Server is configured to run as a service and is depended upon by other servers. Stopping an enterprise SAS Metadata Server using one of the methods described here could adversely affect these other servers.

A personal SAS Metadata Server can be stopped by using SAS Management Console or by using PROC METAOPERATE.

Stopping the Server Using 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, and select **`Stop`** from the File pop-up menu.



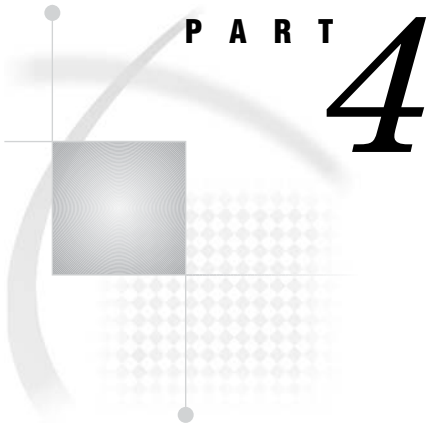
3 The software will open a dialog box asking you to confirm the operation. Click OK.

Stopping the Server Using PROC METAOPERATE

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

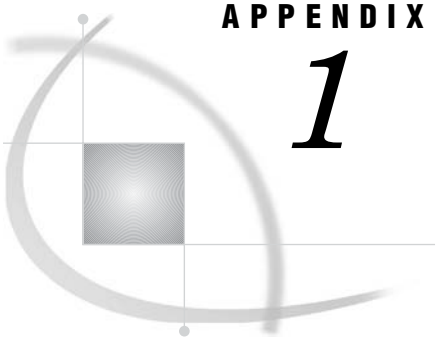
```
PROC METAOPERATE
   SERVER="host_name_of_computer_running_the_server"
   PORT=port_number
   USERID="userid"
   PASSWORD="password"
   PROTOCOL=BRIDGE


   ACTION=STOP;
RUN;
```

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



PART 4

Appendix



APPENDIX

1

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Open Metadata Interface: Reference*
- *SAS Open Metadata Interface: User's Guide*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Your Turn

If you have comments or suggestions about *Getting Started with SAS® 9.1.3 Open Metadata Interface, Second Edition*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**