Metering Service v1.0

Phase One: Product Specification

March, 2000

Metering Service v1.0 Phase One: Product Specification
Final Draft, 03/00

Table of Contents

Introduction
The object of this paper is to give a combined, detailed description of the individual components that makes up the Phase One Metering Service v1.0. The same information can be found in the individual product specifications, which are available for distribution as well.

Metering Service Development
Metering Service enables providers, developers and other participants to allocate and charge for services according to use. There are three phases of development planned for the Metering Service.

The development phases consist of the following:

- **Phase One**- Includes per-use measurement.
- **Phase Two-** Includes measurement of data flow.
- **Phase Three-** Includes measurement by time.

Phase One: Metering Service Specifications

DEFINITION
Phase One includes development of the components as well as a per-use measurement.

TERMINOLOGY
Note: This section is a combined list from all of the specifications.

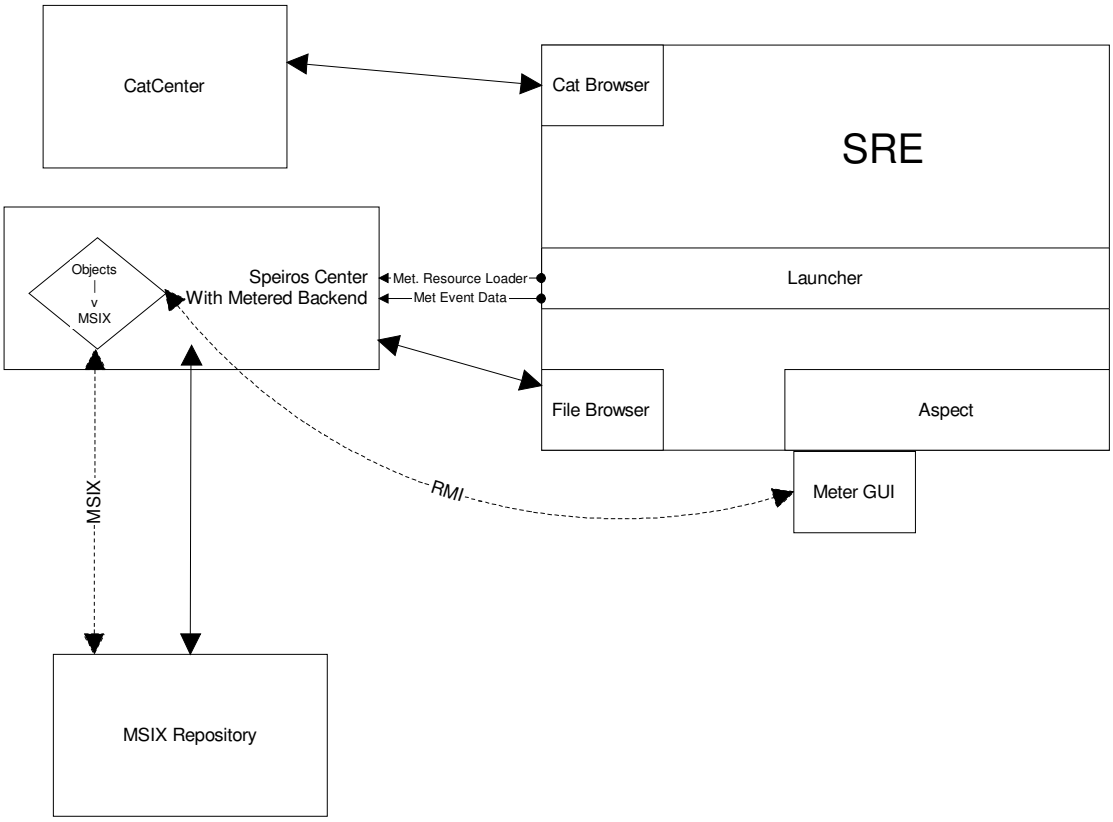
App Center- An antiquated name for the Metering Back End Service.
Client- The entity interfacing with the MBES.
MBES- Metering Back End Service. It acts as the middleman between the SRE (Client) and the MSIX (server).
Measuring unit- The metric used to calculate the cost of using a metered service (i.e per minute, per hour, per day).
MSIX- Metering Service Information eXchange. It is an XTP request-response based protocol used to transfer metering information between entities and was written by Alan Blount and Derek Young.
MSIX Library- A component of the Back End that consists of all of the MSIX requests represented as objects.
MSIX Repository- An application which handles MSIX requests, validates and stores the information, then sends back a response.
Pricing unit- The cost per metric to pay for the service (i.e. dollars, yen, francs).
RMI- Remote Method Indication
Service- A type of task that is performed by an application server for a client that can be measured by the MSIX.
Session- One instance of a particular service usage by an end user.
SRE- Speiros Run Time Environment

FEATURES
Features for Phase One include the development of the Metering Service GUI for verifying the usage of the metered sessions as part of the client side support.

Other features include:

- SRE v1.1 includes support for multiple resource loaders.

- δ Development of the MBES that is designed to act as a middleman between the client and the MSIX repository.
- δ Development of the MSIX Repository that processes and stores data from the client and includes Cyrus Enhancement to MSIX v1.2, Database Connectivity, and Data Reliability.



Metering Service Diagram

COMPONENTS

Phase One consists of the following components: the Client Side Support, Metering Back End Service, and the MSIX Repository.

Note: For more information on the individual components, please see their respective specifications.

1. Client Side Support

The client side support for the metering service verifies the usage of metered sessions and consists of **the Metering Service GUI** and the **Metered Resource Loader**.

1.1 Metering Service GUI

The Metering Graphical User Interface (GUI) displays the costs per measuring unit for using a metered service and displays other pertinent service information.
It also allows the user to accept the charges and invoke the service, or to cancel out.

1.2 Metered Resource Loader

The Metered Resource Loader provides the “hooks” to access the resource and allows the GUI to manipulate the usage of the resource. Reference implementation, which tracks the per use metering, is included for the SRE.

2. Metering Back End Service v1.0

The MBES acts as a middleman between the SRE and the MSIX Repository. It receives metering activity information from the client, stores that information until the session is finished, then translates that session into MSIX Requests to be sent to a MSIX Repository. It consists of the following components: RMI Server and Interface, MSIX Library and Threads, MBES Database, Communicator, and the Service Tool.

2.1 RMI Server and Interface

The RMI Server allows for the delivery of the RMI Interface, which is an abstraction of the MSIX Library, across networks.

2.2 MSIX Library and Threads

The MSIX Library threads validate all requests, store the data in the database, and throws exceptions if there are any problems.

2.3. MBES Database

The MBES database stores the session information.

2.4. Communicator

The Communicator is the component that connects the MBES to a MSIX Repository.

2.5. Service Tool

The Service Tool creates, edits and defines the services.

3. MSIX Repository

The Repository is a secure data repository that takes requests from the Client, processes it, stores the data and then sends the response back. Requests and responses conform to the MSIX v1.2 Protocol. It consists of the following components: the Connection Handler, MSIX Library, MSIX Database, and the Activity Log.

3.1. Connection Handler

Receives Incoming requests and sends outgoing responses.

3.2 MSIX Protocol

This is the message format used by Repository.

3.3 MSIX Database

This database stores all the current request and session information.

3.4 Activity Log

This log keeps track of all the Repository activity. The file is stored under the home directory.

CYCLE OF OPERATION

Note: This is a duplicate of what appears in the Function section for all of the specifications.

In order to offer a metered service, it must first be defined by the administrator of the MBES using the Service Tool. The MBES validates the service, sends an MSIX request to the MSIX Repository (which does its own validation), and if successful, records the service information. The Repository then sends back a response to the MBES. If the response is successful, the service is recorded on the MBES.

When the user launches a metered application, the Metered Resource Loader initiates the GUI as part of the client side support. The GUI contacts the MBES to get the cost information and displays it to the

user. The user then can chose to continue and accept the charges, or cancel. If the user accepts it, the GUI contacts the Speiros Server to add them to the user group, giving it the permissions to load the Java jar files. The Resource Loader downloads the jar files, contacts the MBES to record the usage, and then removes them from the user group.

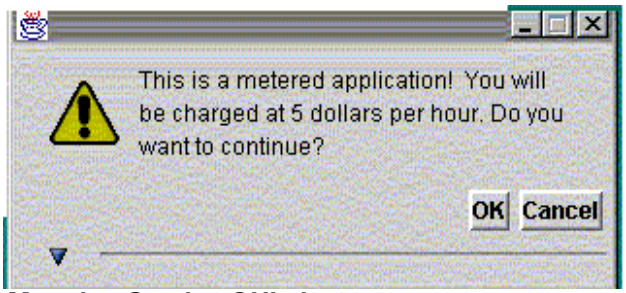
The MBES stores all the session information until the session is closed, at which point the session data is transferred to the Repository and removed from the MBES. For transactions with a per-use measurement, the session is immediately closed and the information is transferred immediately to the Repository since no more data will be needed.

Note: When the per-time measurement transaction is implemented in Phase Three development, the session will be opened when the user launches the metered application, updated at regular increments, and closed when the user is done. Then the session information is transferred to the Repository.

Phase One: Metering Service Client Side Support Specifications

DEFINITION

The client side support for the metering service verifies the usage of metered sessions by displaying the cost for using a metered service via the Metering Service GUI (Graphical User Interface) and utilizes the Metered Resource Loader for manipulating the usage of the resource. It also allows the user to accept the charges and invoke the service, or to cancel out.



Metering Service GUI picture

TERMINOLOGY

Client- The entity interfacing with the MBES.
Metered service- An application provided through Speiros that is measured and billed according to the amount of usage.
Pricing unit- The cost per metric to pay for the service (i.e. dollars, yen, francs).

FEATURES

One of the major features of the client side support is it gives users the option to accept or decline a metered service.

Other features include:

- δ Users can get more information on the service by clicking on a single button.
- δ Accepting the charge allows the user to launch the application.
- δ Reference implementation is included for the SRE v1.1.

COMPONENTS

The Metering client side support consists of the **Metering Service GUI** and the **Metered Resource Loader**.

1. Metering Service GUI

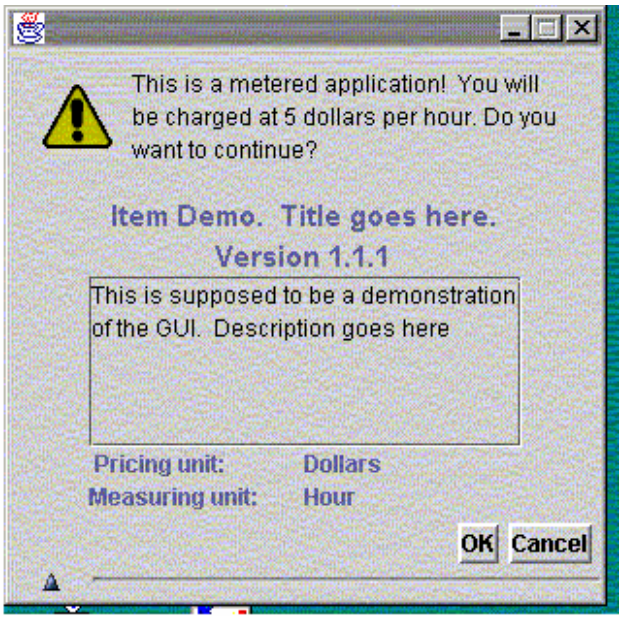
The Metering Service GUI notifies the user they are using a service that is metered and consists of the following elements: Warning Text, Down Arrow, and the Command Buttons.

1.1 Warning Text

The Warning text consists of a warning that the service is metered as well as the charge for using the service, listing the currency and the measuring unit.

1.2 Down Arrow

Clicking on the Down arrow opens the Expanded GUI that includes the **Information Pane**.



Expanded GUI picture

Note: This arrow toggles between the two commands based on what GUI is open. The default is the Less Information GUI.

The **Information Pane** consists of the following:

- **Application field-** Displays the full name of the service being metered.
- **Version field-** Displays the current version of the service.
- **Description box-** Gives a description of the highlighted field.
- **Pricing Unit field**- Gives the cost per metric for the service.
- **Measuring Unit field**- Gives the metric used for calculating the service usage.

1.3 Command Buttons
The Command buttons are located at the bottom of the GUI.

The Command buttons include:

- **OK Button-** Sends the Add Metered Group Request to the server and closes the GUI.
- **Cancel Button-** Closes the GUI without completing the operation.

1.2 Metered Resource Loader
The Metered Resource Loader provides the "hooks" to access the resource and allows the GUI to manipulate the usage of the resource. Reference implementation, which tracks the per use metering, is included for the SRE v1.1.

CYCLE OF OPERATION

The GUI automatically opens when the user launches a meter service alias. It is invoked from the Resource Loader, which then creates an instance of the Metered Server in order to get the application name, version number, description, etc, for display.

The user can accept the charges by clicking on the OK button, or refuse the charges by clicking on the Cancel button. If they cancel, they will not have access to the service represented by the alias. To get

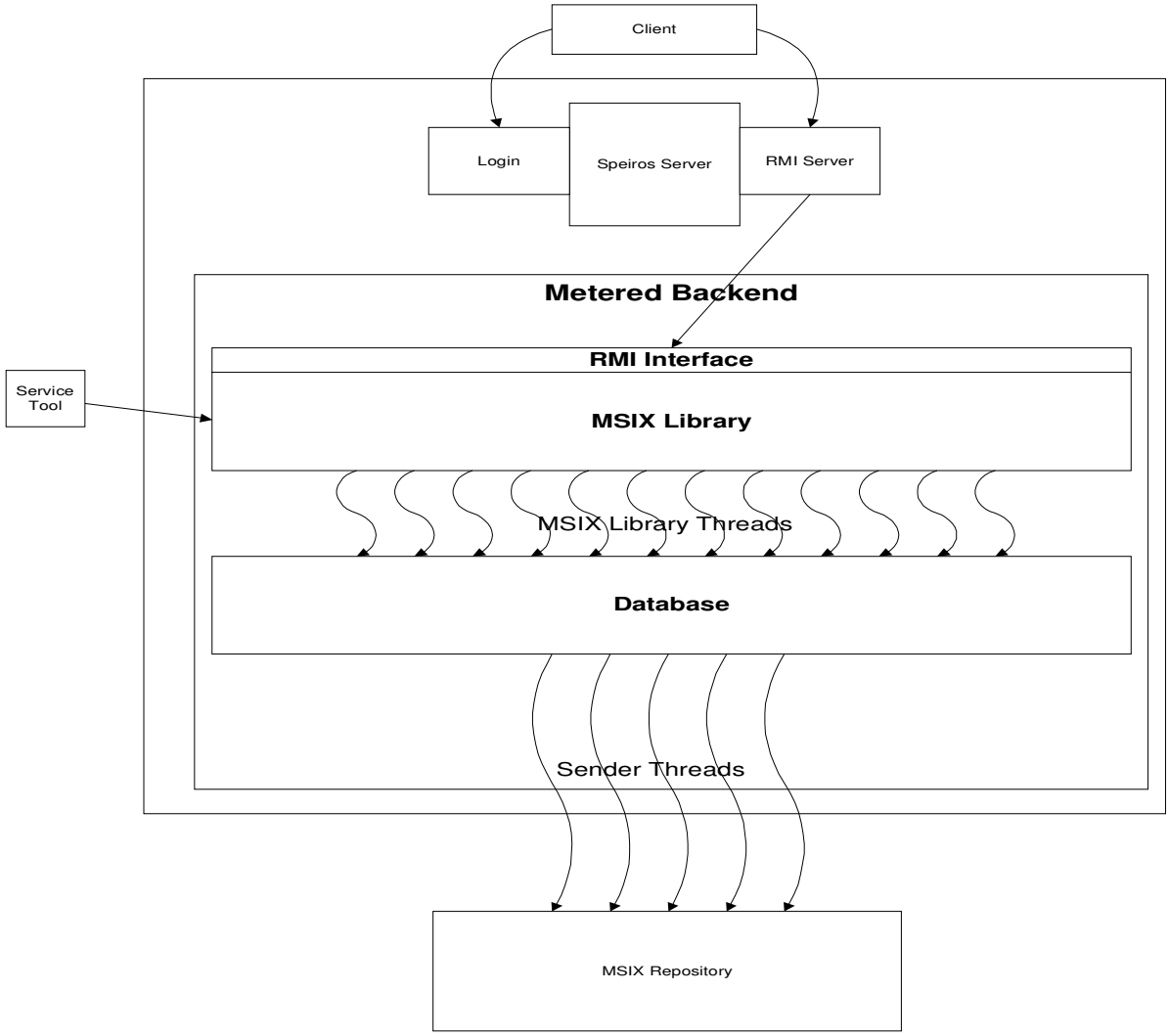
more information regarding the metered service, the user can click on the Down arrow and still have the option to accept or refuse the charges.

When the metered service is accepted, the Resource Loader sends a message to the MBES to add the to a metered group request, giving authorization for the user who is then billed accordingly.

Phase One: Metering Back End Service Specifications

DEFINITION

The Metering Back End Service (MBES) acts as a middleman between the SRE and the MSIX Repository. It receives metering activity information from the client, stores that information until the session is finished, then translates that session into MSIX Requests to be sent to a MSIX Repository.



Metering Back End Service Diagram

FEATURES

One of the major features for the MBES is that clients can choose to commit a session or sessions can be aborted after a Begin Session request was sent.

Other features include:

- Previously defined services can be retrieved from the server.
- Clients can begin a session of any available service provided by the MBES.

- δ Any particular information about each session can be updated at any time.
- δ Clients can find out the cost of each service and the sessions and will be charged accordingly.
- δ Clients can interface with the MBES through the RMI Server or through the MSIX Library.
- δ Services can be created, edited and defined with the Service Tool.

COMPONENTS

The MBES consists of the following components: Speiros Authenticator, RMI Server, RMI Interface, MSIX Library, MBES Database, Senders, Communicator, and the Service Tool.

1. Speiros Authenticator

The Speiros Authenticator is a part of the Speiros server and handles the user authentication using Natz Protocol.

2. RMI Server

The RMI Server allows for the delivery of the RMI Interface across networks. SRE makes RMI calls to pass in parameters either to look up a session, begin a session, update a session, commit a session or abort a session. The RMI server will then initialize MSIX Library threads to take in those parameters and start

3. RMI Interface

The RMI Interface is an abstraction of the MSIX Library used to meter service usage.

4. MSIX Library and Threads

The MSIX Library threads validate all requests, store the data in the database, and throws exceptions if there are any problems.

5. MBES Database

The MBES database stores the session information.

6. Sender Threads

The Sender threads take closed sessions from the database and send MSIX requests to the MSIX Repository. It then handles the response and, if successful, delete the session information from the database.

7. Communicator

The Communicator is the component that connects the MBES to a MSIX Repository.

8. Service Tool

The Service Tool creates, edits and defines the services.

CYCLE OF OPERATION

The client connects either through the RMI Sever or directly through the MSIX Library. The MSIX Library threads take the information from the RMI Interface, validate and, if necessary, store the session information in the database.

The Sender threads search the database for closed sessions and when they find one, they mark the session as being transferred for recovery purposes. They then take the MSIX Request from the database, get a connection to the MSIX Repository, send the request to the Repository, and wait for the response. If the Sender thread receives a success response from the Repository, the data has been accurately transferred to the Repository. The Sender thread can then safely delete the session information from the MBES's database.

The client interfaces with the MBES through the RMI Server or through the MSIX Library. The user should be able to get information about each available service through the lookup function provided by the Metered Back End.

Phase One: MSIX Repository Specifications

DEFINITION
The MSIX Repository is a secure data repository that takes requests from the client, processes it, stores the data and then sends the response back. Requests and responses conform to the MSIX v1.2 protocol.

Note: While one Repository can service many Clients, the component will be referred to in the singular.

FEATURES
One of the major features for the Repository is it can service more than one client.

Other features include:

- It implements the MSIX v1.2 protocol.
- Cyrus Intersoft has extended the Repository Protocol to include Query Session, which returns information about sessions or services.
- It can communicate with different types of JDBC compliant databases.
- Session data is not lost by service interruptions.
- Data is transmitted securely using SSL protocol.
- Access is controlled by user authentication.

1. MSIX Transaction Protocol
MSIX protocol involves a Service and a Session. The service of any particular type must first be defined at *msix.org* in order for a session to start. Version 1.2 of the MSIX Protocol is implemented.

Note: To fully understand the MSIX Protocol, please refer to the MSIX Protocol Specifications v1.2.

2. Repository Requests
The Repository can process the MSIX requests as defined by the v1.2 Protocol.

COMPONENTS
The Repository consists of the following components: the Connection Handler, MSIX Library, MSIX Database, and the Activity Log.

1. Connection Handler
Receives Incoming requests and sends outgoing responses.

2. MSIX Protocol
This is the message format used by Repository.

3. MSIX Database
This database stores all the current request and session information.

4. Activity Log
This log keeps track of all the Repository activity. The file is stored under the home directory.

CYCLE OF OPERATION
The client connects with the MSIX Repository through an SSL connection and sends the MSIX request to the Repository. The Repository parses the request, determines what the request command is and

creates the appropriate command object. The command validates the request and stores a recovery command object in the database to retrieve if there is an interruption in service. It then processes the command, sends back the response to the client and deletes the recovery command object from the database.