



MESSAGEmanager FSID Web Service

MESSAGEmanager Solutions

June 2008

Successful IT systems increasingly require interoperability across platforms and flexible services that can easily evolve over time and are independent of programming language, software platform, and hardware.

With broad support across vendors and businesses, Web services enable computer systems on any platform to communicate over corporate intranets, extranets, and across the Internet with support for end-to-end security, reliable messaging, distributed transactions, and more.

Web services are based on a core set of standards that describe the syntax and semantics of software communication: XML provides the common syntax for representing data; the Simple Object Access Protocol (SOAP) provides the semantics for data exchange; and the Web Services Description Language (WSDL) provides a mechanism to describe the capabilities of a Web service. Additional specifications, collectively referred to as the WS-* architecture, define functionality for Web services discovery, eventing, attachments, security, reliable messaging, transactions, and management.

Overview

MESSAGEmanager FSID Web Service is a W3C-compliant Web Service interface to MESSAGEmanager FSID Services that enables third party client applications to send, receive, update and query outbound or inbound messages via a web services.

This web service intended for use when a client application needs to communicate with MESSAGEmanager FSID services and Microsoft's .NET framework is not available or firewalls prevent .NET operation across security boundaries.

A typical scenario is a Java application in a UNIX environment sending fax or SMS messages, or processing received messages. Another scenario is a distributed Windows environment where domain trust issues prevent COM communication between the client application and MESSAGEmanager FSID services.

FSID web service is an ASP.NET 2.0 web service with Web Services Enhancements (WSE 3.0). All file transfers use the MTOM mechanism.

How it works

The MESSAGEmanager Web Service exposes the following web methods to submit, query and update messages via a web server.

1. **QueryInboundMessages** - This web method will query and return all inbound messages matching selection criteria specified in the query argument.
2. **QueryOutboundMessages** - This web method will query and return all outbound messages matching selection criteria specified in the query argument.
3. **UpdateOutboundMessage** - This web method will updated an outbound message if it is still in the pending queue.
4. **GetFileFromInbound** - Web Method to download a received message file using MTOM.
5. **GetFileFromOutbound** - Web Method to download a sent message file using MTOM.
6. **SubmitMessage** - Web method to submit a new message to MESSAGEmanager FSID server for transmission.

All examples in this document use NetBeans IDE 6.0.1 with Java 1.5.0.

Submitting a Message

This method allows a client to submit an outbound message, including attachment files, for transmission.

Refer to OutboundMessageW below for property definitions and values.

Method definition:

```
int SubmitMessage( OutboundMessageBuilderW message )
```

Arguments:

- Message - an object of the OutboundMessageBuilderW class

Return:

This method returns an integer value as follows:

- 0 - Success
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 4 - Invalid argument (such as submitting a message without any destination number)
- 6 - Security exception, web service can not access the MESSAGEmanager FSID server
- 9 - All MESSAGEmanager servers have failed

The following example demonstrates how to create and submit a message.

```
// create new OutboundMessageBuilder and set properties
mmntwstestclient.OutboundMessageBuilderW msg =
    new OutboundMessageBuilderW();

msg.setKeyFile("This is a test from a Java application");
msg.setKeyFileFlag(KeyFileFlags.FL_TEXT);
msg.setAccount("Sales");
msg.setVoiceNumber("+61 2 8448 8890");
msg.setMessageID("Message ID");
msg.setUserType("LN");
msg.setUserID("Jane Wilson");
msg.setNotifyFlag("A");
ArrayOfString sendTo = new ArrayOfString();
sendTo.getString().add("John Smith");
msg.setSendTo(sendTo);
ArrayOfString sentBy = new ArrayOfString();
sentBy.getString().add("Jane Wilson");
msg.setSentBy(sentBy);
msg.setSubject("This is test fax from Java via Web Service");
mmntwstestclient.ArrayOfString destNumbers =
    new mmntwstestclient.ArrayOfString();
destNumbers.getString().add("8448 8860");
msg.setDestNumbers(destNumbers);
msg.setProtocol("F");
mmntwstestclient.ArrayOfPutFileRequest attachments =
    new mmntwstestclient.ArrayOfPutFileRequest();
mmntwstestclient.PutFileRequest tmpAtt =
    new mmntwstestclient.PutFileRequest();
File fl = new File("c:\\TestFax.tif");
tmpAtt.setFileName(fl.getName());
```

```

FileInputStream flStr = new FileInputStream(fl);
int size = flStr.available();
byte[] data = new byte[size];
flStr.read(data);
tmpAtt.setFileData(data);
attachments.getPutFileRequest().add(tmpAtt);
msg.setAttachments(attachments);
mmntwstestclient.MMNTws service = new mmntwstestclient.MMNTws();
mmntwstestclient.MMNTwsSoap port = service.getMMNTwsSoap12(); int
result = port.submitMessage(msg);

```

QueryInboundMessages

This method gets a list of inbound messages from one or more servers according to selection criteria specified by a query string. MESSAGEmanager FSID Web Service passes it on to one or more MESSAGEmanager FSID Servers which perform the query and hand over a list of messages to the application.

See Query Syntax for details.

Using an empty query and preferred server will return all messages from all servers.

Method definition:

```

int QueryInboundMessages(string preferredServer, uint dbType,
                        string queryString, out List<InboundMessageW> messages)

```

Arguments:

- preferredServer - string argument specifies a specific MESSAGEmanager FSID server to query, if left blank FSID Web Service will query the routing server defined in the web.config file on the web server as well as all servers in the routing server's global routing group.
- dbType - this is an unsigned integer that defines which database to query.
Acceptable values:
 - 1 - to query the inbound queue
 - 2 - to query the inbound Archive database
- queryString- see Query Syntax for details.
- Messages- this is an out reference or a holder argument for the returned messages.

Return:

This method returns an integer value as follows:

- 0 - Success
- 1 - Web service has an invalid configuration
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 4 - Invalid query string
- 5 - Invalid database type
- 6 - Security exception, web service can not access the MESSAGEmanager FSID server

This example will query all messages from the inbound queue.

```

mmntwstestclient.MMNTws service = new mmntwstestclient.MMNTws();
mmntwstestclient.MMNTwsSoap port = service.getMMNTwsSoap();
Holder<mmntwstestclient.ArrayOfInboundMessageW>inboundMessages
    = new Holder<mmntwstestclient.ArrayOfInboundMessageW>();
int result = port.queryInboundMessages( "", 1, "",
queryInboundMessagesResult, inboundMessages);

if(inboundMessages.value.getInboundMessageW() != null)
{
    //...
    // Application code to process messages
    //...
}

```

QueryOutboundMessages

This method gets a list of outbound messages from one or more servers according to selection criteria specified by a query compiled by the calling client program. MESSAGEmanager FSID Web Service passes the query on to one or more MESSAGEmanager FSID servers which perform the query and hand over a list of message to the application. See Query Syntax for details. Using an empty query and preferred server will return all messages from all servers.

Function definition:

```

int QueryOutboundMessages(string preferredServer, uint dbType,
    string queryString, out List<OutboundMessageW> messages)

```

Arguments:

preferredServer - string argument specifying a specific MESSAGEmanager FSID server to query, if left blank FSID Web Service will query the routing server defined in the web.config file on the web server as well as all servers in the routing server's Global Routing group.

dbType - this is an unsigned integer that defines which database to query. Acceptable values:

- 3 - to query the pending queue
- 4 - to query the outbound complete queue
- 5 - to query the outbound Archive database

queryString- see Query Syntax for details.

Messages- this is an out reference or a holder argument for the returned messages.

Return:

This method returns an integer value as follows:

- 0 - Success
- 1 - Web service has an invalid configuration
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 4 - Invalid query string
- 5 - Invalid database type
- 6 - Security exception, web service can not access the MESSAGEmanager FSID server
-

This example will query all messages from the Outbound complete queue.

```

mmntwstestclient.MMNTws service = new mmntwstestclient.MMNTws();
mmntwstestclient.MMNTwsSoap port = service.getMMNTwsSoap();
Holder<mmntwstestclient.ArrayOfOutboundMessageW>outboundMessages
    = new Holder<mmntwstestclient.ArrayOfOutboundMessageW>();
int result = port.queryOutboundMessages( "", 4, "",
    queryOutboundMessagesResult, outboundMessages);

if(outboundMessages.value.getOutboundMessageW() != null)
{
    //...
    // Application code to process messages
    //...
}

```

UpdateOutboundMessage

Sometimes after a user has submitted a message, that message needs to be updated. A typical scenario is cancelling or postponing a message, or increasing a message's priority. This can be achieved by updating either or both of the following properties. This is only meaningful if the message is still in the pending queue, meaning **it has not been sent yet**.

1. Priority
2. DeferralTime

Refer to OutboundMessageW documentation for all possible values of those properties

Function definition:

```
int UpdateOutboundMessage( OutboundMessageW message )
```

Arguments:

- message - an object of OutboundMessageBulderW class.

Return:

This method returns an integer value which could mean following:

- 0 - Success
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 3 - Invalid operation (such as the selected database is not the pending queue)
- 6 - Security exception, web service can not access the MESSAGEmanager FSID server
- 7 - Message not found (usually because the message has just been sent and moved to the outbound complete queue)

The following example demonstrates how to cancel a message with MSN (message sequence number) 888009.

```

mmntwstestclient.MMNTws service = new mmntwstestclient.MMNTws();
mmntwstestclient.MMNTwsSoap port = service.getMMNTwsSoap();
Holder<mmntwstestclient.ArrayOfOutboundMessageW>outboundMessages
    = new Holder<mmntwstestclient.ArrayOfOutboundMessageW>();

```

```

int result = port.queryOutboundMessages( "", 3, "Msn=888009",
                                        queryOutboundMessagesResult, outboundMessages);

if(result == 0 && outboundMessages.value.getOutboundMessageW() != null)
{
    OutboundMessageW msg =
    outboundMessages.value.getOutboundMessageW().get(0);
    // to cancel a message all you need to do is set the priority to 0
    msg.priority = 0;
    result = port.updateOutboundMessage(msg);
}

```

GetFileFromInbound

Each inbound message contains a received file. This web method allows you to download that file using MTOM.

Function definition:

```

GetFileResponse GetFileFromInbound( InboundMessageW message )

```

Arguments:

- message - object of InboundMessageW class.

Return:

This method returns an object of type [GetFileResponse](#), refer to this class documentation for details.

Member variable *status* of the GetFileResponse object assigned by the web service is a status value which could be one of the following:

- 0 - Success
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 6 - Security exception, web service can not access the MESSAGEmanager FSID server
- 7 - message not found (the message has just has been purged)

The following example demonstrates how to download the message file from the inbound message with MSN 888009:

```

mmntwstestclient.MMNTws service = new mmntwstestclient.MMNTws();
mmntwstestclient.MMNTwsSoap port = service.getMMNTwsSoap12();
Holder<mmntwstestclient.ArrayOfInboundMessageW>inboundMessages
    = new Holder<mmntwstestclient.ArrayOfInboundMessageW>();
int qResult = port.queryInboundMessages( "", 1, "Msn=888009",
                                        queryInboundMessagesResult, inboundMessages);

if(qResult == 0 && inboundMessages.value.getInboundMessageW() != null)
{
    InboundMessageW msg =
        inboundMessages.value.getInboundMessageW().get(0);

    mmntwstestclient.GetFileResponse result = port.getFileFromInbound(msg);
    if(result.getStatus() == 0)

```

```

    {
        String fileName = "DownloadedFile" + result.getFileExtension();
        FileOutputStream stream = new FileOutputStream(fileName);
        stream.write(result.getFileData());
        stream.close();
    }
}

```

GetFileFromOutbound

Each successful outbound message contains a file representing the message as sent. This web method allows you to download that file using MTOM.

Function definition:

```
GetFileResponse GetFileFromOutbound( OutboundMessageW message )
```

Arguments:

- message - object of OutboundMessageW class.

Return:

This method returns an object of type [GetFileResponse](#), refer to this class documentation for details.

Member variable *status* of the GetFileResponse object assigned by the web service is a status value which could be one of the following:

- 0 - Success
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 6 - Security exception, web service can not access the MESSAGEmanager FSID server
- 7 - message not found (the message has just has been purged)

Following example demonstrates how to download the message file from the outbound message with MSN 888010:

```

mmntwstestclient.MMNTws service = new mmntwstestclient.MMNTws();
mmntwstestclient.MMNTwsSoap port = service.getMMNTwsSoap12();
Holder<mmntwstestclient.ArrayOfOutboundMessageW>outboundMessages
    = new Holder<mmntwstestclient.ArrayOfOutboundMessageW>();
int qResult = port.queryOutboundMessages( "", 4, "Msn=888010",
    queryOutboundMessagesResult, outboundMessages);

if(qResult == 0 && outboundMessages.value.getOutboundMessageW() != null)
{
    OutboundMessageW msg =
        outboundMessages.value.getOutboundMessageW().get(0);

    mmntwstestclient.GetFileResponse result = port.getFileFromOutbound(msg);
    if(result.getStatus() == 0)
    {
        String fileName = "DownloadedFile" + result.getFileExtension();
        FileOutputStream stream = new FileOutputStream(fileName);
        stream.write(result.getFileData());
    }
}

```

```
    stream.close();
}
}
```

Class Library

Following are class definitions.

Please note that depending on what tools you are using to connect to MESSAGEmanager FSID Web Service actual generated Java or other language code and property access methods could be slightly different.

Refer to the WSDL file for XML definitions by typing in your browser: <http://localhost/mmntws/mmntws.asmx?wsdl>, Note that you might need to replace localhost with the machine name where the MESSAGEmanager FSID Web Service is installed.

Following are the actual C# definitions of all classes exposed:

InboundMessageW - this class defines an Inbound Message object with all properties and methods.

Available Properties - all properties are read only.

uint MSN - **MESSAGEmanager message sequence number.**

string CiuFile - **The message file, if any.**

DateTime CompletionTime - **The UTC time at which message reception was completed, whether successfully or otherwise.**

string Protocol - **The message type - F for fax, S for SMS.**

uint Duration - **The call duration, in seconds.**

uint Pages - **The total number of pages in the message file (fax only).**

uint Status - **The message status. Zero means a successfully received message. Any other value indicates a problem, usually a premature call termination. Status codes can be translated to text using a table of defined error codes in the FSID documentation.**

string RecvCSID - **This is the sender's CSID.**

string ServerName - **The name of the MESSAGEmanager FSID Server which received the message.**

uint ChannelNumber - **The number of the channel on which the message was received.**

string NotifierID - **Identifies the system which should handle routing. Specified by the MESSAGEmanager FSID Server when receiving messages.**

uint Speed - **Speed at which this message was received (fax only).**

string Digits - A string containing the message's routing digits as provided by the PBX or IP gateway.

string CallerID - **The sender's phone number, if available.**

OutboundMessageW - this class defines an Outbound Message object with all properties and methods.

Available read-only properties:

uint MSN - **MESSAGEmanager message sequence number.**

string CiuFile - **The message file, if any.**

DateTime CompletionTime - **The UTC time at which message transmission was completed, or abandoned.**

string Protocol - **The message type - F for fax, S for SMS and E for email**

destinations.

uint Duration - **The call duration, in seconds.**

uint Pages - **The total number of pages in the message file (fax only).**

uint Status - **The message status. Zero means a successfully transmitted message. Any other value indicates a problem. Status codes can be translated to text using a table of defined error codes in FSID documentation.**

string RecvCSID - **This is the receiver's CSID.**

string ServerName - **The name of the MESSAGEmanager FSID Server which sent the message.**

uint ChannelNumber - **The number of the channel on which the message was finally sent.**

string NotifierID - **Identifies the system which should handle notification.**

string UserID - **This identifies the email address to which notifications should be sent.**

string UserType - **This identifies the email system by which notifications should be sent. This is a two-character Gateway user type. Possible values are:**

- Exchange-EX
- Domino-LN
- SMTP-SM
- XML - XM

uint Speed-**Transmission speed at which this message was finally sent.**

string MessageID- **Identifies the originating application's message identification, if any.**

string SendCSID- **Identifies the CSID string to be included in the transmitted message as if the message originated from a fax machine with this CSID.**

string Account- **Identifies the account/billing code.**

List<string> DestNumbers- **The destination fax or SMS number or email address. MESSAGEmanager supports only one destination per message, however this is a multi-valued property to support an alternative destination (of the same protocol).**

DateTime SubmissionTime- **The UTC date and time as specified by the originating application.**

DateTime AcceptanceTime- **The UTC date and time at which the sending server accepted the message.**

DateTime Try1Time- **The UTC date and time at which the sending server first tried dialling the primary destination.**

uint NTries- **The number of times the sending server has dialled the primary destination number.**

string PageHeader- **A short string which will be displayed in the per-page fax header, if the sending server is configured for a page header.**

string CoverFile- **Identifies the fax cover page template file. The template is assumed to be already stored on the server.**

List<string> SendTo- **A list of strings to identify the message recipient on the cover page, if a fax. Usually this contains two strings: the recipient's personal name and company name.**

List<string> SentBy- **A list of strings to identify the message sender on the cover page, if a fax. Usually this contains two strings: the sender's personal name and department or company name.**

string Subject- **Message subject line. For a fax this is used on the cover page. For an SMS message, this can be the whole message if no body text is provided. For an email this is the email subject line. The subject line is also often used for message tracking.**

string VoiceNumber- **Sender's phone number, just for use on the fax cover page.**

double CallCost- **The cost of the call, as estimated by the sending server, in the currency of that server.**

string ConnNumber- **The number as dialled by the sending server on the dial attempt which sent the message.**

uint PagesSoFar- **The number of pages which had been successfully sent at the time the message data was retrieved from the server.**

string Notified- **True if message notification by the application was required and is now complete. False if message notification was not required and application has processed the record.**

string NotifyFlag- **Specifies the message notification mode.**

Possible values:

- "A" for always,
- "F" for failures only
- "N" (or empty) for no notification.

Available read/write properties, if a message is still in the pending queue:

uint Priority- Message priority, from 1 (the most urgent) through 9 (least urgent). Priority zero means that the message is to be cancelled. If not specified, then priority 5 will be used (routine).

DateTime DeferralTime- If defined, the sending server will not send the message before this UTC time.

OutboundMessageBuilderW - this class defines an Outbound Message to be built and submitted to MESSAGEmanager Web Service for transmission.

Properties

List<string> DestNumbers- **The destination fax or SMS number or email address. MESSAGEmanager supports only one destination per message, however this is a multi-valued property to support an alternative destination (of the same protocol).**

string UserID - **The email address of the user who should be notified when the message has been sent or failed.**

string UserType - **This identifies the email system by which notifications should be sent. This is a two-character Gateway user type. Possible values are:**

- Exchange-EX
- Domino-LN
- SMTP-SM
- XML - XM

string Account- **Identifies the account/billing code.**

List<string> SendTo- **A list of strings to identify the message recipient on the cover page, if a fax. Usually this contains two strings: the recipient's personal name and company name.**

List<string> SentBy- **A list of strings to identify the message sender on the cover page, if a fax. Usually this contains two strings: the sender's personal name and department or company name.**

string Subject- **Message subject line. For a fax this is used on the cover page. For an SMS message, this can be the whole message if no body text is provided. For an email this is the email subject line. The subject line is also often used for message tracking.**

string VoiceNumber- **Sender's phone number, just for use on the fax cover page.**

string Protocol - **The message type.**

Possible values:

- F for fax
- S for SMS
- E for email destinations.

string SendCSID- **Identifies the CSID string to be included in the transmitted message as if the message originated from a fax machine with this CSID.**

string MessageID- **Identifies the application's message identification, if any. This can be anything meaningful to the client application; the server makes no use of it.**

string NotifyFlag- **Specifies the message notification mode.**

Possible values:

- "A" for always,
- "F" for failures only
- "N" (or empty) for no notification.

DateTime SubmissionTime- **The UTC date and time as specified by the originating application.**

string PageHeader- **A short string which will be displayed in the per-page fax header, if the sending server is configured for a page header.**

string CoverFile- **Identifies the fax cover page template file. The template is assumed to be already stored on the server.**

uint TifPages- an estimate of the number of TIF pages in the attachments. This is only used as a hint to Global Routing.

uint DocPages- an estimate of the number of document pages requiring serverbased conversion. This is only used as a hint to Global Routing.

uint Priority- Message priority, from 1 (the most urgent) through 9 (least urgent). If not specified, then priority 5 will be used (routine).

DateTime DeferralTime- If defined, the sending server will not send the message before this UTC time.

string KeyFile- A string to be used as the message on the fax cover page or in the SMS or email body text. Two formats are allowed: plain text or HTML. The format is defined by KeyFileFlags property.

keyFileFlags KeyFileFlag-This defines the format of the KeyFile string.

Possible values are defined in the OutboundMessageBuilder.keyFileFlags enumeration.

List<PutFileRequest> Attachments-A list of PutFileRequest objects defining all attachments. Refer to PutFileRequest class definition.

GetFileRequest class

This class defines a single file transmission using MTOM from the web server to client application

Properties:

String FileExtension- File extension. Such as ".tif" or ".pdf" for fax messages, ".txt" for SMS messages, or anything for email messages.

Byte[] FileData - a byte array of the file's data.

int Status - **Status code from the web service regarding availability of requested file.**

Possible values:

- 0 - Success
- 2 - Cannot connect to the MESSAGEmanager FSID server
- 6 - Security exception, web service cannot access the MESSAGEmanager FSID server
- 7 - message not found (usually because the message has been purged)

PutFileRequest class

This class defines a single file transmission using MTOM from client to the web server

Properties:

String FileName- File name of the transferred file. If the ByReference flag is set this should contain a full path meaningful to the web service and the MESSAGEmanager FSID server (such as a file in a shared directory on the network). If the ByReference flag is not set then this string is only required because the MESSAGEmanager server needs to know the file extension to be able to convert to TIF or PDF.

Byte[] FileData-a byte array of the file's data to be submitted to the server. If submitting by reference this should be null.

bool ByReference - boolean flag to define if this attachment is to be transferred by reference.

Enums

OutboundMessageBuilderW.keyFileType - Defines the type of a keyFile text when submitting a message

- fiText - key file as plain text
- fiHTML - Key file as HTML
- fiRtf - key file as RTF (not supported yet by the server)

Message Query Syntax

Having specified a database type, the client application defines the records it is interested in by specifying database fields and values embedded in a simple syntax described below.

The query engine will include a message in the result list if the following Boolean is true

T1 & T2 & T3 ...

where "&" means Boolean AND and each T (for term) is of the form

PropertyName operator PropertyValue

PropertyName is the literal string value of one of the properties defined in the InboundMessage or OutboundMessage classes. An operator is one of the characters "<" "=" ">" or "?" (meaning string contains). PropertyValue is a property value expressed as a string.

Example terms are:

UserID = Debra // UserID exactly matches "Debra"

Duration > 60 // Duration greater than 60 seconds

AcceptanceTime < 1209960013 // Acceptance time before 05/05/2008 04:00:13

Note: all date/time values are expressed as UTC time in seconds from 01/01/1970 00:00:00

An example query is:

SendTo = Barry & DestNumbers ? 84488840

This selects messages in which SendTo exactly matches "Barry" and DestNumbers contains the string 84488840.

If a PropertyValue string contains spaces or characters which are used for Boolean operators then the query can use double quotes to enclose the string, as in this example:

SendTo = "Barry Jones" & DestNumbers ? 84488840

The query engine ignores letter case and leading and trailing blanks. In the case of the "?" operator, it also ignores embedded blanks. In the case of multi-valued properties it examines only the first value.

There is no provision for bracketed expressions or sums within products. For example, this is not a valid query:

(SendTo = Barry) & (DestNumbers ? 84488840 + MSN = 89006)

MESSAGEmanager FSID Web Service Installation

Prerequisites:

1. Windows 2003 Server
2. IIS 6.0
3. ASP .NET 2.0 or higher
4. MESSAGEmanager Server R1-2008 licensed for Web Services

Installation sequence:

1. Make sure IIS and .NET2.0 are installed.
2. Run setup.exe from MMNTws install kit.

Configuring Security settings to access web service:

1. Open IIS manager from shortcut menus or executing this command line:
%SystemRoot%\system32\inetmgr\iis.msc
2. Expand Applications Pool and select MMNTwsPool
3. Right click and select properties
4. Then go to the Identity tab
5. If the FSID server is on the same machine as MESSAGEmanager FSID Web Service you need only select Local System in the drop down list. If the FSID server is on a different machine you will need to select a domain account which has access to that FSID server. You may also need to configure COM security on the remote FSID server, see **"THAT DOCO ABOUT DCOM CNFG"**

6. To see the MESSAGEmanager Web Service interface in a web browser use this URL: <http://localhost/mmntws/mmntws.asmx>
7. To get the WSDL file (an XML file which defines the web service) use this URL: <http://localhost/mmntws/mmntws.asmx?wsdl>

Running the example java application

1. Install Java 1.5 or higher
2. Copy MMNTwsTestJava folder to any directory on your test machine.
3. Execute this command line, using the directory as above:
`Java -jar %path from number2%\dist\MMNTwsTestJava.jar`

Note: the sample Java application was created by NetBeans IDE 6.0.1 www.netbeans.org. You can edit the sample code using the same IDE and opening the project by pointing the IDE to the root of MMNTwsTestJava.

Troubleshooting the MESSAGEmanager FSID Web Service

The MESSAGEmanager FSID Web Service has trace logging built in, which can be enabled via its web.config file. The log can be viewed using the ASP.NET tracing mechanism. Find the web.cong file (usually in C:\Program Files\MESSAGEmanager Solutions\MESSAGEmanager\MMNTws). Find the <trace> tag. Change <trace enabled="false" to <trace enabled="true". To view the trace type the following URL in your browser: <http://localhost/mmntws/trace.axd>

To enable tracing to a file read and follow the comments in the web.config file.



Level 8, 9 Help Street,
Chatswood, NSW 2067
Australia
www.mmanager.com
info@mmanager.com

SYDNEY
Tel: +61 2 8448 8800
Fax: +61 2 8448 8811

SINGAPORE
Tel: +65 6779 4769
Fax: +61 2 8448 8839

NEW ZEALAND Toll Free: 0800 445 308

CANADA Toll Free: 1877 3701 261

UNITED KINGDOM Toll Free: 0800 169 8226

USA Toll Free: 1877 8841 664