XML Open Integration Module Installation and User Guide

9.12.0 v1

Symmetry™ Security Management

9600-0469

NOTE: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.

Operation of this equipment in a residential area is likely to cause harmful interference. In which case, the user will be required to correct the interference at his own expense.

XML Open Integration Module Installation and User Guide
(9600-0469)

Issue 9.12.0v1 – 9th April 2025
Applies to version 9.12.0 of the Symmetry software.

Contents

About this Manual iv

Chapter 1: Introduction 1

Overview of the XML Open Integration Module 1
- Purpose of the Open Integration Module 1
- Possible Applications 2

Benefits of the XML Open Integration Module 2
How the XML Open Integration Module Works 3
Summary of Symmetry XML Methods 4

Chapter 2: Installing the Software 5

Minimum System Requirements 5
- Symmetry Requirements 5
- Web Server PC 5
- Third-Party Application Client PC 5
- Development Machine 6

Upgrading the XML Open Integration Module 6
Step 1 – Enable IIS at the Web Server 7
Step 2 – Install the XML Module 8
Step 3 – Install the XML Module at the Web Server 10
Step 4 – Install the SSL Certificate (HTTPS Only) 11
Step 5 – Configuring Time Tolerance Settings 12
Removing the XML Open Integration Module 13

Chapter 3: Worked Example 14

Introduction 14
Completed Program Code 15
Step 1 – Design the Form 16
Step 2 – Add the G4T Service Reference 16
Step 3 – Add Namespace 18
Step 4 – Define Login Code 18
Step 5 – Define Code for Get Readers Button 19
Step 6 – Compile and Run Code 19

Chapter 4: Methods 20

About this Chapter 20
G4TAddAlarms 21
G4TAddAlarmsWithCommandLine 22
G4TAddExtendedAlarms 24
G4TClearOutstandingAlarm 25
G4TEditAccessCode 27
G4TEditFloorGroup 30
G4TEditReaderGroup 32
G4TGetAccessGroups 34
G4TGetActivity 35
G4TGetActivityDocument 39
G4TGetAdminTypes 40
G4TGetAlarmActivityCategory 41
G4TGetBadgeFormats 42
G4TGetCardHolderInformation 43
G4TGetCardHolderInformationDocument 53
G4TGetCommands 54
G4TGetCommandTypes 56

G4TGetCompanies **57**
G4TGetCurrentAntipassbackZone **58**
G4TGetDeviceAlarms **59**
G4TGetDevices **61**
G4TGetDeviceStatus **62**
G4TGetDeviceTypes **66**
G4TGetDoorStatus **67**
G4TGetImportCardHolderDataStatus **68**
G4TGetImportRecordRequests **69**
G4TGetLastCardTransactionsForAReader **70**
G4TGetLastCardTransactionsForAReaderGroup **71**
G4TGetLastReaderUsedForCardHolder **72**
G4TGetLastReadersUsedForCardHolder **73**
G4TGetOutstandingAlarms **74**
G4TGetOutstandingAlarmsDocument **77**
G4TGetPersonalDataTitles **78**
G4TGetSystemParameters **80**
G4TGetTimeCodes **81**
G4TGetVersionInformation **82**
G4TImportCardHolderData **83**
G4TLogin **89**
G4TRename **90**
G4TSendDeviceCommand **91**
G4TSendPCDoorControlAccessCommand **93**
G4TSetAlarmActivityCategory **94**
G4TSetCardHoldersAntipassbacktoNeutral **95**
G4TSetCardHoldersAntipassbackZone **96**
Base Types **97**
Types used in Several Methods **98**

Chapter 4: Troubleshooting 104

About this Manual

This manual explains the following:

- The purpose, benefits and concepts of the XML Open Integration Module.
- How to install the software.
- Technical details of each XML method, allowing developers to use them in third-party applications. Sample applications are provided, showing examples of each method.
- A worked example of how to use the Open Integration Module. The examples in this manual use C#, in the Microsoft® Visual Studio® .NET development environment.

The manual is intended to be of use to:

- Managers who need to find out the capabilities of the XML Open Integration Module.
- Personnel, such as software engineers, who intend to develop products to provide an interface to the Symmetry software.

The manual assumes knowledge of the Symmetry software. Please contact your authorized support agent for details of applicable training courses. Information is also available in the *Symmetry Software User Guide*.

This manual should be read in conjunction with the product help, which is also available in printed form as the *Symmetry Software Reference Manual*.

Chapter 1: Introduction

Overview of the XML Open Integration Module

Purpose of the Open Integration Module

The XML Open Integration Module is a software package that allows third-party applications to interface with the Symmetry software using XML and Web Communication Foundation (WCF) technologies.

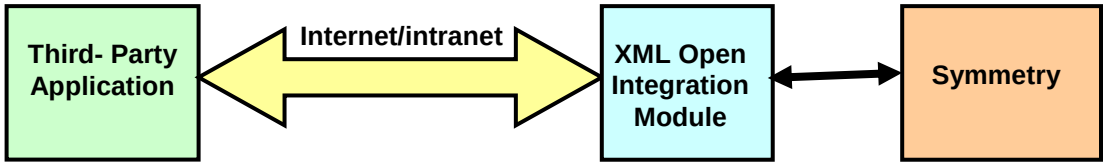


Figure 1-1: XML Open Integration Module

XML and WCF are recognized as being fundamental to distributed computing on the internet. They are an industry standard for achieving easy communication between diverse systems, irrespective of the programming languages or platforms that the different systems use. XML and WCF use web-based technologies, enabling easy and cost-effective implementation, particularly for companies that already have a web infrastructure.

The XML Open Integration Module is supported for all editions of Symmetry (Business, Professional and Enterprise).

The XML Open Integration Module license also enables the Symmetry "Operation/Data/Import" screen, if not enabled by default or by a separate license.

The Open Integration Module allows many routine Symmetry operations to be carried out, such as to:

- Import details of card holders and visitors. You can, for example, add new card holders, delete card holders, modify existing card holder data, make cards inactive and change access rights.
- Obtain the details of card holders and visitors who are already in the Symmetry database.
- View, acknowledge and clear outstanding Symmetry alarms.
- Send a command to a device (e.g. to open a door or switch on a CCTV camera).
- View the status of a Symmetry device (e.g. to determine whether a door is locked or unlocked).
- Import alarms from external equipment, such as intruder systems.

Note: Code samples are provided for Microsoft Visual Studio 2022 on the installation media. It is the customer's responsibility to carry out all programming work for any XML interfaces.

Possible Applications

The Symmetry XML Open Integration Module has many possible applications. Examples include:

- Integration of Symmetry with third-party systems, such as Intruder Detection, Building Management, Payroll, Personnel, and Enterprise Requirements Planning (ERP) systems. Such an integration could be used to enable card holder or other data to be exchanged between Symmetry and third-party systems, and alarm information to be transferred to Symmetry for centralized alarm handling.
- The development of common user interfaces to integrate different systems (e.g. Building Management, Security Management and Intruder systems).
- Implementing specific Symmetry control features from internet or intranet applications. For example, such an application could remotely raise and lower a particular barrier or open doors to allow identified individuals to pass through, without giving the user access to the Symmetry user interface.
- Specialist command and control applications (e.g. military applications).

Benefits of the XML Open Integration Module

The XML Open Integration Module can provide many benefits. For example:

- **Easy inter-platform integration** – Use of the industry standard XML and Windows Communication Foundation (WCF) technologies simplifies and standardizes the task of linking diverse systems.
- **Saves time when entering data** – An existing payroll or personnel database can be imported into Symmetry, avoiding the need to re-enter data.
- **Common user interface** – A common user interface can be used to control key elements of Symmetry and third-party systems. This can increase efficiency and reduce training costs.
- **Database sharing** – The card holder and visitor information in the Symmetry database can be accessed by a third-party system, allowing the sharing of staff data and reducing database maintenance.
- **Integration with Building Management Systems** – Monitor points set up in the Symmetry software can be used to switch on or off heating and ventilation systems. This can improve building management efficiency and minimize energy consumption.
- **Integration with Intruder Detection System** – Commands from Symmetry readers can be used to arm or disarm some types of intruder system, and Symmetry monitor points (such as motion detectors) can be used as additional sensors. This can result in the need for less hardware, and reduced costs.
- **Centralized alarm management** – Alarms can be exported to the Symmetry software to enable alarm management from a single system.

How the XML Open Integration Module Works

The operation of the XML Open Integration Module is shown schematically in Figure 1-2.

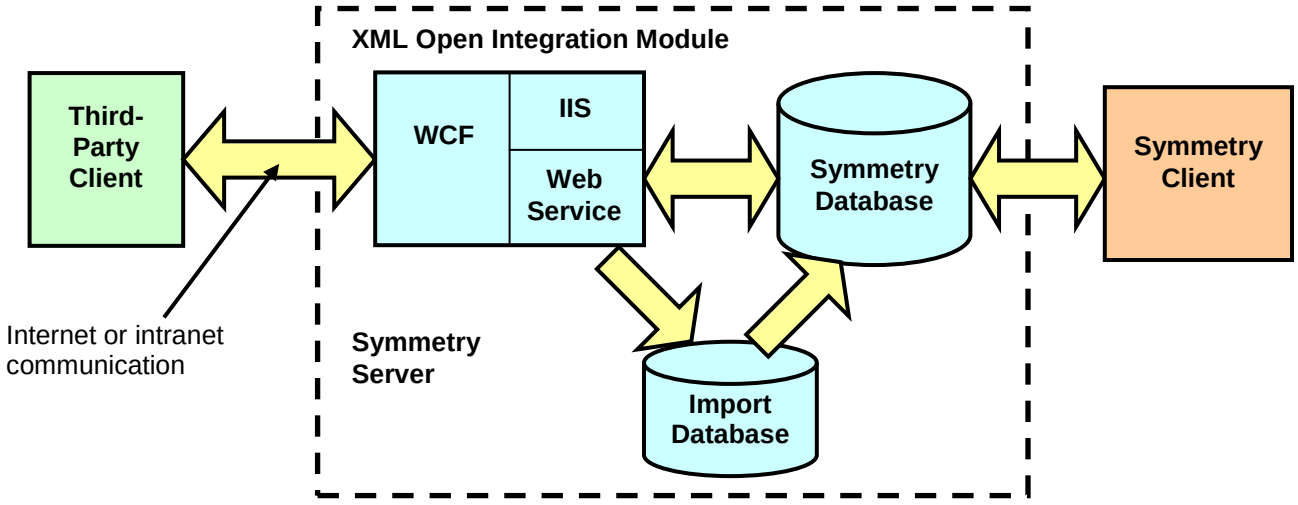


Figure 1-2: Accessing Symmetry Using the XML Open Integration Module

The XML Open Integration Module operates as a web service, named smsXMLWebService, which runs on a standard IIS (Internet Information Services) web server using WCF (Windows Communication Foundation), which is freely available from Microsoft. IIS is available as part of Microsoft operating systems.

The web service responds to API commands (otherwise known as "methods") written into the software running on the third-party client. The client may be communicating with the Symmetry server over the internet or a local intranet. A summary of methods supported by the web server is provided in the next section.

For every method call received, WS Security checks the username and password against those stored in the Symmetry database, which may be located on the Symmetry server or on a separate database server. If there is no match, WS Security returns a fault exception to the third-party client. If there is a match, WS Security allows the method call to pass to the web service, where it is processed and interacts with the main Symmetry database or the Symmetry import database. The Symmetry response is then returned to the third-party client.

When importing card holder or visitor data from the third-party client, (see the G4TImportCardHolderData method on page 83) the data is initially stored in the import database until the next automatic import time (unless the third-party client instructs an immediate import).

A user has access only to company data and devices in his or her company group, as specified in the Symmetry software. The user's company group is determined by the login username.

The Symmetry server can use a different computer from the web server.

Note: It is recommended that the third-party software is designed to minimize the number of XML requests. In common with other types of XML interface, it is not intended for the XML Open Integration Module to be used as a real-time, high-volume data provider. If appropriate, the third-party software should be designed to store retrieved Symmetry data locally in its own database, and distribute that data to the third-party clients using faster methods than are possible with an XML interface.

Summary of Symmetry XML Methods

The following methods are supported by the XML Open Integration Module:

G4TAddAlarms (page 21) – Imports an alarm from an external system (legacy method).

G4TAddAlarmsWithCommandLine (page 22) – Imports alarms from an external system.

G4TAddExtendedAlarms (page 24) – Imports an alarm from an external system, with an image.

G4TClearOutstandingAlarm (page 25) – Clears or acknowledges an outstanding alarm.

G4TEditAccessCode (page 27) – Used to add, edit or delete access codes.

G4TEditFloorGroup (page 30) – Used to add, edit or delete floor groups.

G4TEditReaderGroup (page 32) – Used to add, edit or delete reader groups.

G4TGetAccessGroups (page 34) – Retrieves access codes.

G4TGetActivity (page 35) – Retrieves all alarms/events from a specified alarm/event ID or date and time.

G4TGetActivityDocument (page 39) – Same as G4TGetActivity, but provides pagination support.

G4TGetAdminTypes (page 40) – Retrieves a description for each device type.

G4TGetAlarmActivityCategory (page 41) – Retrieves all alarm categories.

G4TGetBadgeFormats (page 42) – Retrieves the names and IDs of badge designs.

G4TGetCardHolderInformation (page 43) – Gets a list of card holders and visitors, with access rights.

G4TGetCardHolderInformationDocument (page 53) – Same as G4TGetCardHolderInformation, but provides pagination support.

G4TGetCommands (page 54) – Retrieves the set of commands of a specified type.

G4TGetCommandTypes (page 56) – Retrieves a meaningful description for each command type.

G4TGetCompanies (page 57) – Retrieves the IDs of companies.

G4TGetCurrentAntiPassbackZone (page 58) – Returns the current antipassback zone of a card.

G4TGetDeviceAlarms (page 57) – Determines how messages from devices are handled.

G4TGetDevices (page 61) – Retrieves the description and ID of all devices of the specified type.

G4TGetDeviceStatus (page 62) – Lists the status of a device.

G4TGetDeviceTypes (page 66) – Retrieves a description for each device type (legacy applications only).

G4TGetDoorStatus (page 67) – Lists the status of a door.

G4TGetImportCardHolderDataStatus (page 68) – Returns the status of a card holder/visitor import.

G4TGetImportRecordRequests (page 69) – Retrieves a description for each import operation.

G4TGetLastCardTransactionsForAReader (page 70) – Finds the last transactions for a specified reader.

G4TGetLastCardTransactionsForAReaderGroup (page 71) – Finds the last transactions for a reader group.

G4TGetLastReaderUsedForCardHolder (page 72) – Finds the last reader used by a card holder/visitor.

G4TGetLastReadersUsedForCardHolder (page 73) – Finds the last readers used by a card holder/visitor.

G4TGetOutstandingAlarms (page 74) – Retrieves alarms that have not been cleared.

G4TGetOutstandingAlarmsDocument (page 77) – Same as G4TGetOutstandingAlarms, but provides pagination support.

G4TGetPersonalDataTitles (page 78) – Retrieves card holder or visitor personal data titles.

G4TGetSystemParameters (page 80) – Returns the Symmetry system start and end dates.

G4TGetTimeCodes (page 81) – Returns the time codes.

G4TGetVersionInformation (page 82) – Retrieves the versions of the Symmetry software.

G4TImportCardHolderData (page 83) – Imports card holder or visitor details.

G4TLogin (page 89) – Makes a test call on the username and password.

G4TRename (page 90) – Enables a device to be renamed.

G4TSendDeviceCommand (page 91) – Sends a command to a Symmetry device.

G4TSendPCDoorControlAccessCommand (page 93) – Grants/denies access (PC Door Control mode).

G4TSetAlarmActivityCategory (page 94) – Allows alarm categories to be added or modified.

G4TSetCardHoldersAntipassbackToNeutral (page 95) – Resets antipassback for a specified card.

G4TSetCardHoldersAntipassbackZone (page 96) – Sets a card to be in a specified antipassback zone.

Chapter 2: Installing the Software

Minimum System Requirements

Symmetry Requirements

The XML Open Integration Module is supported for all editions of Symmetry (Business, Professional, and Enterprise) and can be installed on the Symmetry server or on the web server machine. Installation on the web server is recommended if the Symmetry server is occupied with servicing a large number of demanding devices or external connections.

Note: Although the XML Open Integration Module can be installed on the Symmetry server, it must **not** be installed on a separate Symmetry client machine (i.e. a thick client), or on a machine that hosts the Symmetry M4000 multiNODE API server software.

For Enterprise Edition, the Symmetry database can be on the Symmetry server or on a separate database server.

Web Server PC

Hardware:

The hardware requirements for the web server depend on the load applied to the web service. Factors such as the number of simultaneous connections and data retrieval /request frequency will affect performance and machine requirements. As a guideline, the minimum machine specifications for a Symmetry Professional Edition server (as specified in the *Symmetry Software Installation Manual*) should be suitable in most cases.

Software:

- Any supported web browser, as documented in the *Symmetry Software Installation Manual*.
- One of the operating systems as supported by a Symmetry Professional Edition server, as documented in the *Symmetry Software Installation Manual*. **Note:** A Windows Server operating system is recommended.

Note: The XML Open Integration Module is supported on a Windows Server computer acting as an Active Directory **member server**, or acting as a workgroup server. The XML Open Integration Module is **not** supported on a Windows Server computer acting as a Primary Domain Controller.

Third-Party Application Client PC

Hardware:

Because of the diversity of applications in which the XML Open Integration Module may be used, the minimum specification of the machine running the third-party client application will need to be determined by the client.

As a guide, the minimum machine specifications for a standard Symmetry client can be applied, as specified in the *Symmetry Software Installation Manual*.

Software:

- Microsoft .NET Framework (installed by the Open Integration Module). The XML Open Integration Module uses version 4.8 of the .NET Framework.

Development Machine

Software:

- Microsoft Visual Studio 2022.
- Microsoft .NET Framework (freely available from the Microsoft web site). The XML Open Integration Module has been tested with version 4.8 of the .NET Framework.

Upgrading the XML Open Integration Module

Upgrading from any version of the Symmetry XML Open Integration Module that uses Windows Communication Foundation (WCF) can be achieved by running the Symmetry XML installation software on the web server to overwrite the previous installation. In this case, existing applications do not need to be recompiled, providing that the connection details to the XML web server have not changed (e.g. the hostname).

Upgrading from any version of the Symmetry XML Open Integration Module that uses Web Services Enhancements (WSE) requires a complete uninstall of the old version, a new install the new version, and then applications need to be amended and recompiled as described below. WSE is no longer supported by the XML Open Integration Module.

To amend and recompile existing WSE applications:

1. Back up each application.
2. Upgrade to Visual Studio 2022.
3. Right-click **Project** and select **Properties**. In the Application tab, set the Target framework to .net framework 4 client profile.
4. Remove references to the following in the code:

   ```
   Microsoft.Web.Services2
   using Microsoft.Web.Services2;
   using Microsoft.Web.Services2.Security.Tokens
   ```

5. Under **Solution**, right-click the existing Web Reference and select **Remove**.
6. Add the G4T Service Reference (follow Step 2 on page 16).
7. Change the following lines of code:

   ```
   G4TAPI.RequestSoapContext.Security.Tokens.Clear();
   UsernameToken token = new UsernameToken(fldUserName.Text,
   fldPassword.Text.ToLower(), PasswordOption.SendHashed);
   G4TAPI.RequestSoapContext.Security.Tokens.Clear();
   G4TAPI.RequestSoapContext.Security.Tokens.Add(token);
   ```

to the following.

```
G4TAPI = new SMSXMLWebServiceClient();
G4TAPI.ClientCredentials.UserName.UserName = fldUserName.Text;
G4TAPI.ClientCredentials.UserName.Password = fldPassword.Text.ToLower();
G4TAPI.ClientCredentials.ServiceCertificate.Authentication.CertificateValidationMode =
X509CertificateValidationMode.None;
```

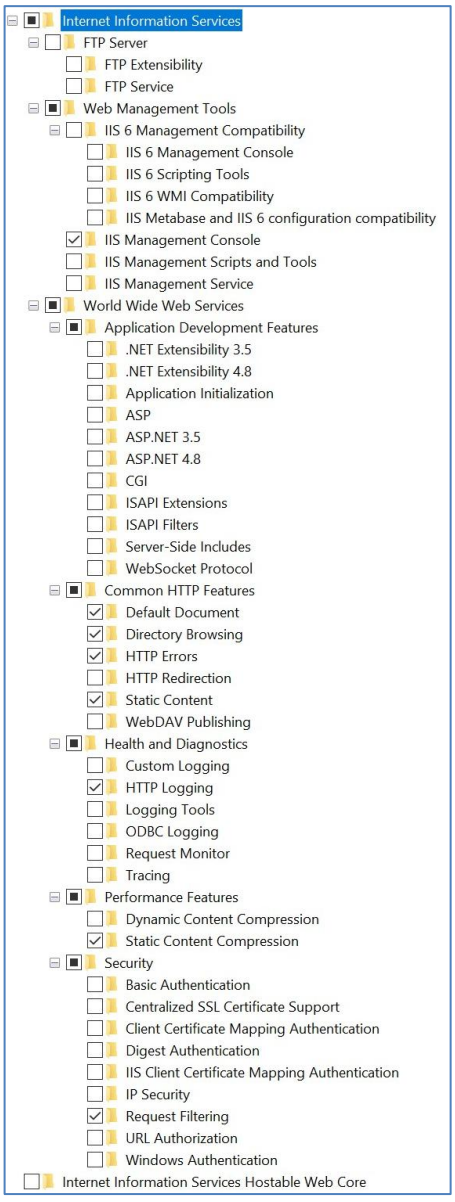
8. Recompile the applications.

Step 1 – Enable IIS at the Web Server

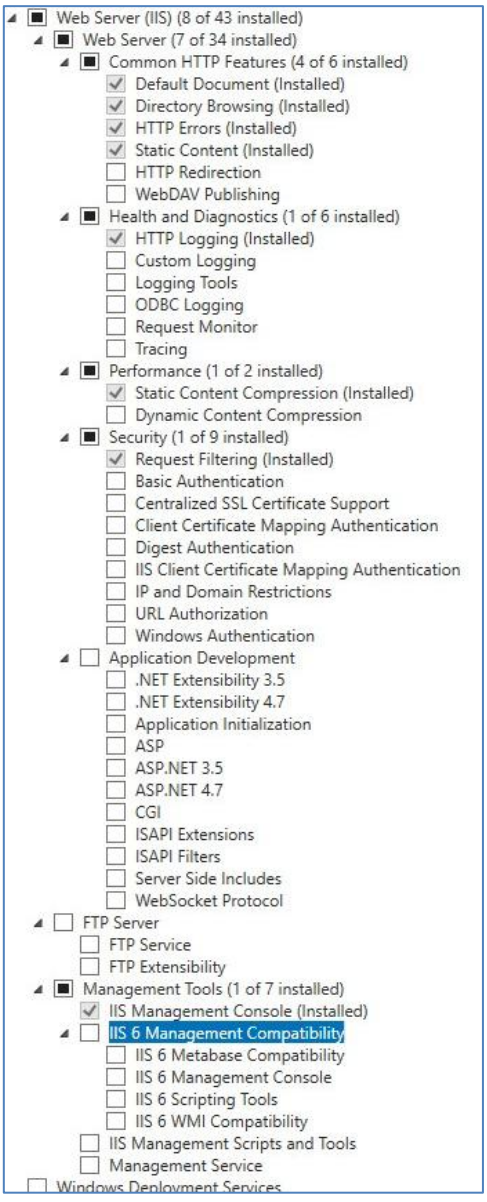
Using the Windows Control Panel, enable Internet Information Services (IIS) on the web server. The web server can be the same machine as the Symmetry server or a different machine.

The following pictures show the default IIS settings in example operating systems.

Windows 10



Windows Server 2019



The following additional IIS components are installed during the installation of the XML Open Integration Module:

- .NET Extensibility 4.x
- ASP.NET 4.x
- ISAPI Extensions
- ISAPI Filters

Step 2 – Install the XML Module

At the Symmetry server, carry out the following procedure to install the XML Open Integration Module.

Note: If the XML Open Integration Module is to be used in a clustered system, please refer to the *Cluster Installation Manual* for additional information about system installation.

1. Create a Windows account for the XML Open Integration Module services.

   If a workgrouped network is used, create the account on the Symmetry server and make it a member of the ACSServices group. The name of the group may be "ACSServices", or a different group name may have been selected for use as ACSServices during the installation of the Symmetry software (as described in the *Symmetry Software Installation Manual*). The same account name and password must exist at the web server, but membership of the ACSServices group is not needed.

   If a domain network is used, create the account in Active Directory and make it a member of the domain ACSServices group. Domain Administrator privileges are not needed.

2. Using Windows Explorer, double-click Setup.exe located on the XML Open Integration Module installation media.

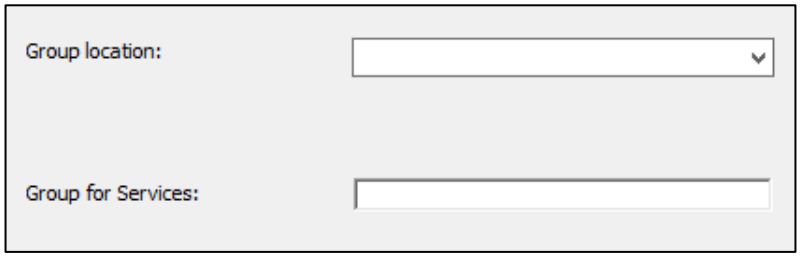
   Note: In some cases, it can take a while for the initial preparation phase to complete and display the Welcome screen.

3. The installation software may need to install certain Windows components and may prompt you to restart the computer. If you are prompted to restart the machine, do so, and re-run the installation process by double-clicking Setup.exe again.

4. Click **Next** at the Welcome screen.

5. Read the license agreement and click **I accept...** if you agree. Follow the remaining prompts:

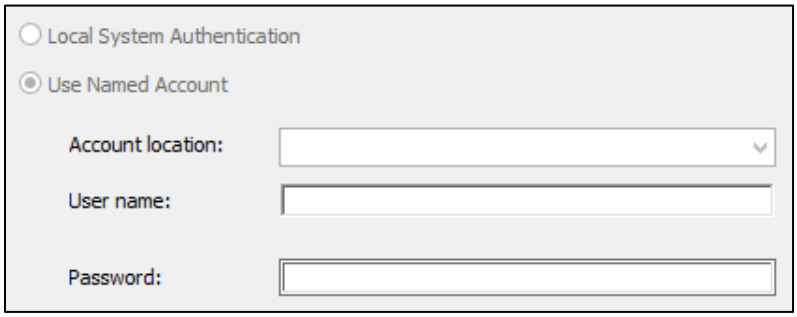
6. **Symmetry Account Groups** – Select the location and name of the group used for the symmetry services:



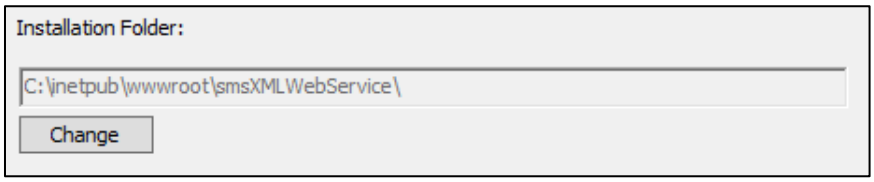
   Group location – Choose the Symmetry server (if a workgroup is used), or the domain.

Group for Services – Specify the name of the "ACSServices" group. Make sure that the spelling is exactly correct.

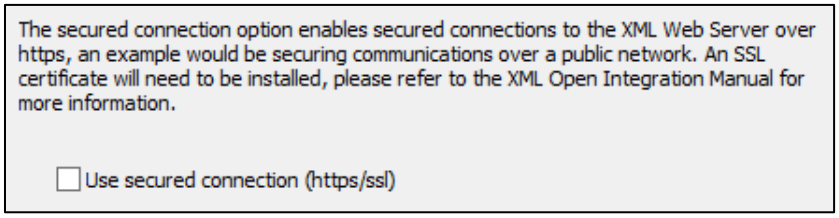
7. **Symmetry Services Account** – Select **Use Named Account** and specify the details of the account in which to run the Symmetry services (as configured in step 1). Make sure that the spelling is exactly correct:



8. **Installation Folder** – Choose the installation folder:



9. **Secure Communications** – Choose whether to use a secured (https) connection:



If you select **Secured Connection (https/ssl)**, encrypted (https) communication will be used. If you do not select this option, the Open Integration Module will use non-encrypted (http) communication, which is appropriate in most cases where a secure network is being used. If you use encrypted communication, you will need to install an SSL certificate (page 11).

10. **Ready to install XML Open Integration** – Click **Install**.

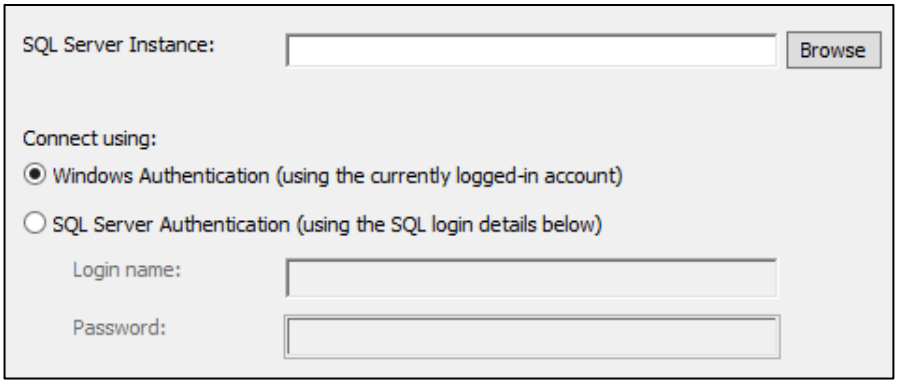
11. Click **Finish** when the installation process has completed.

12. Install an XML Open Integration license at the Symmetry server using the "Maintenance/Licensing/System Licenses" screen.

Step 3 – Install the XML Module at the Web Server

Note: This step is necessary only if the web server is not using the same computer as the Symmetry server.

At the web server, repeat steps 2 to 11 in *Step 2 – Install the XML Module*. The same prompts are displayed, with the following additional prompt displayed after accepting the license agreement:



Enter the name of the SQL Server instance you want to use, or click **Browse** to select it.

Note:

- For Enterprise Edition, you can choose the Symmetry server, or an instance on a separate database server.
- If login fails, make sure that TCP/IP is enabled (see page 14).
- If you are using a fixed port for a named database instance, it is necessary to include the port number in the format: <SQLServerName>\<InstanceName>,<PortNumber>
  For example, MySQLServer\SymmInstance,6532

Connect using specifies the authentication method used to connect to the database server while Symmetry is being installed. It is recommended to use Windows Authentication. The authentication method is used only during the installation of Symmetry to set up the database; Symmetry always uses Windows authentication after installation.

To use SQL Server Authentication, the SQL server instance must have "mixed mode" authentication enabled (not recommended), and you will need the login ID and password for the administrator account defined within the SQL Server instance.

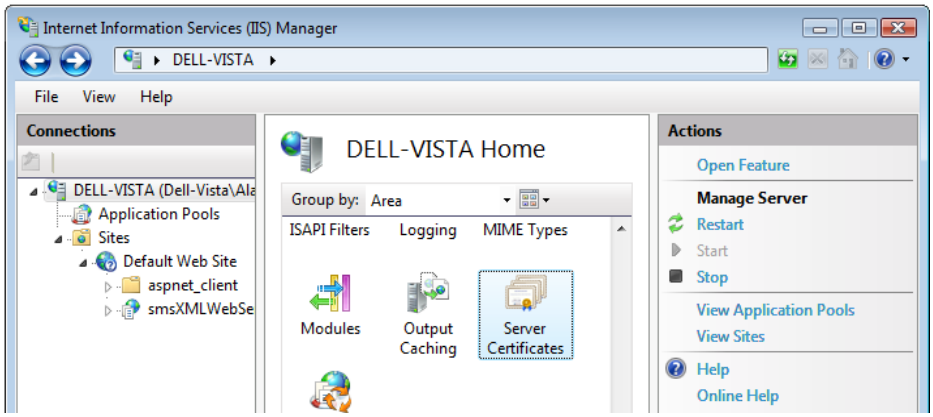
Step 4 – Install the SSL Certificate (HTTPS Only)

If you selected **Secured Connection (https)** during the installation of the XML Open Integration Module software, you will need to install an SSL certificate at the web server. The procedure may differ depending on the operating system; use the following as a guide.

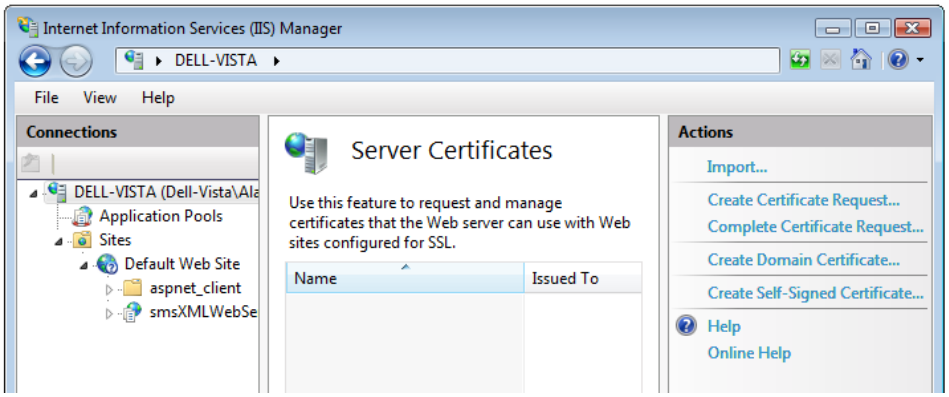
Note: If you are installing on a cluster, install on both the active and passive nodes.

1. If you do not want to generate your own certificate, obtain an SSL certificate from a certificate authority or generated from a server that has the Active Directory Certificate Services (CS) role installed. In the latter case, ask an appropriate administrator for the certificate.

2. Open the IIS Manager on the web server:



3. Select the local server name, then double-click **Server Certificates** in the IIS section. The following is displayed:



4. Click one of the following:

   Import – To import an existing certificate (.pfx file)

   Complete Certificate Request – To install a certificate received from an internal or external certificate authority.

   Create Self-Signed Certificate – To create your own certificate.

5. Ensure the certificate appears in the Server Certificates page.

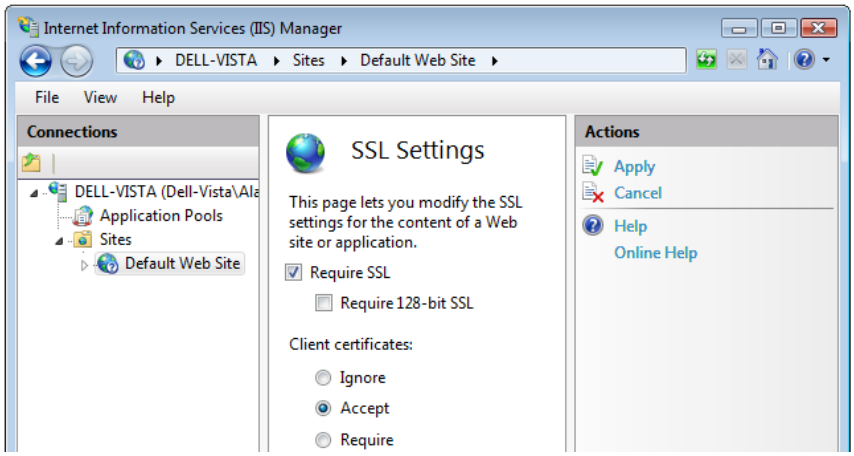
6. Right-click **Default Web Site** and choose **Edit Bindings**.

7. Click **Add**, choose **https** from the **Type** menu and select the SSL certificate. Click **OK**.

8. Optionally, remove **http** from the Site Bindings dialog. Click **Close**.

9. Click **Default Web Site**, then double-click **SSL Settings** in the IIS section.

10. Select the **Require SSL** checkbox, and either **Ignore**, **Accept** or **Require**, depending on the level of security needed. If required, please refer to the online help for further information.



11. Open Web.config using a text editor such as Notepad. The file is located in the folder:

    inetpub\wwwroot\smsXMLWebService

12. Search for instructions in Web.config that include "SSL", and make the changes as directed. Save changes.

    Note: On a cluster, both the active and passive nodes must have the same Web.config file.

13. Restart IIS.

Step 5 – Configuring Time Tolerance Settings

Time-critical functions can be compromised if the time at different computers is not the same. To reduce the possibility of a problem caused by lack of time synchronization, you can use the MaxClockSkew property in the Web.config file. Web.config includes a comment giving a placeholder where it may be appropriate to implement the property.

At the time of writing this guide, information about how to implement MaxClockSkew was available in the following articles:

"How to: Set a Max Clock Skew" (Microsoft Developer Network):

http://msdn.microsoft.com/en-us/library/aa738468(v=vs.110).aspx
(Under "To set the MaxClockSkew in configuration")

"maxClockSkew on wsHttpBinding" (Microsoft Developer Network, Visual Studio):

http://social.msdn.microsoft.com/Forums/vstudio/en-US/0e8c30ab-e5a0-40b1-9722-c1b20a09c8ad/maxclockskew-on-wshttpbinding

Removing the XML Open Integration Module

At the Symmetry server (and separate web server, if applicable):

1. Open **Add\Remove Programs** in the Windows Control Panel.
2. Highlight **XML Open Integration**.
3. Click **Uninstall** and follow the prompts.

Alternatively, the XML Open Integration Module can be removed by double-clicking Setup.exe on the installation media again. When prompted, click **Remove**.

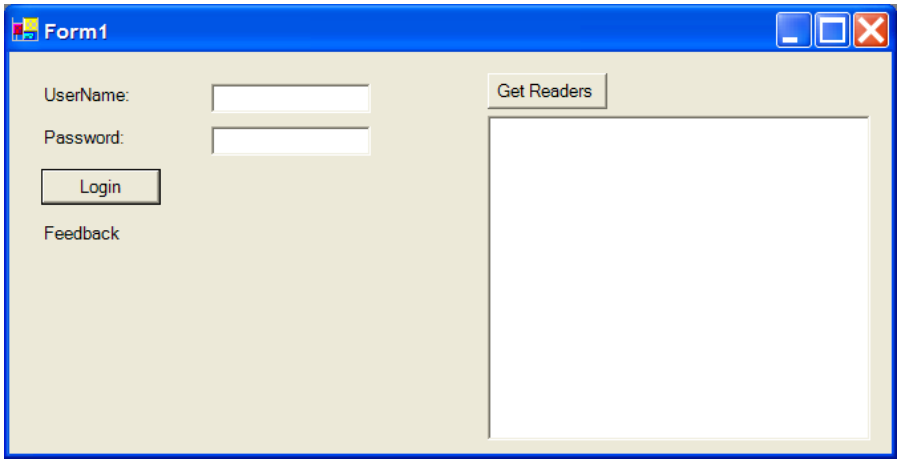
Chapter 3: Worked Example

Introduction

In this chapter, you will work through a simple example of developing a simple C# program allowing your computer to communicate with a Symmetry database via the XML Open Integration Module web service. The example uses Microsoft Visual C#, in the .NET development environment, to create a dialog that allows a user to test the login to Symmetry and obtain a list of the names of all the reader devices.

Note: After you have completed this example, you may wish to view the sample code in the Documentation folder on the XML Open Integration Module installation CD. These files provide further examples of implementing the various methods provided by the XML Open Integration Module. After copying the files to your hard drive, remove the **Read-only** attribute from the files before attempting to use them in the development environment.

The form you will create looks like this in run mode:



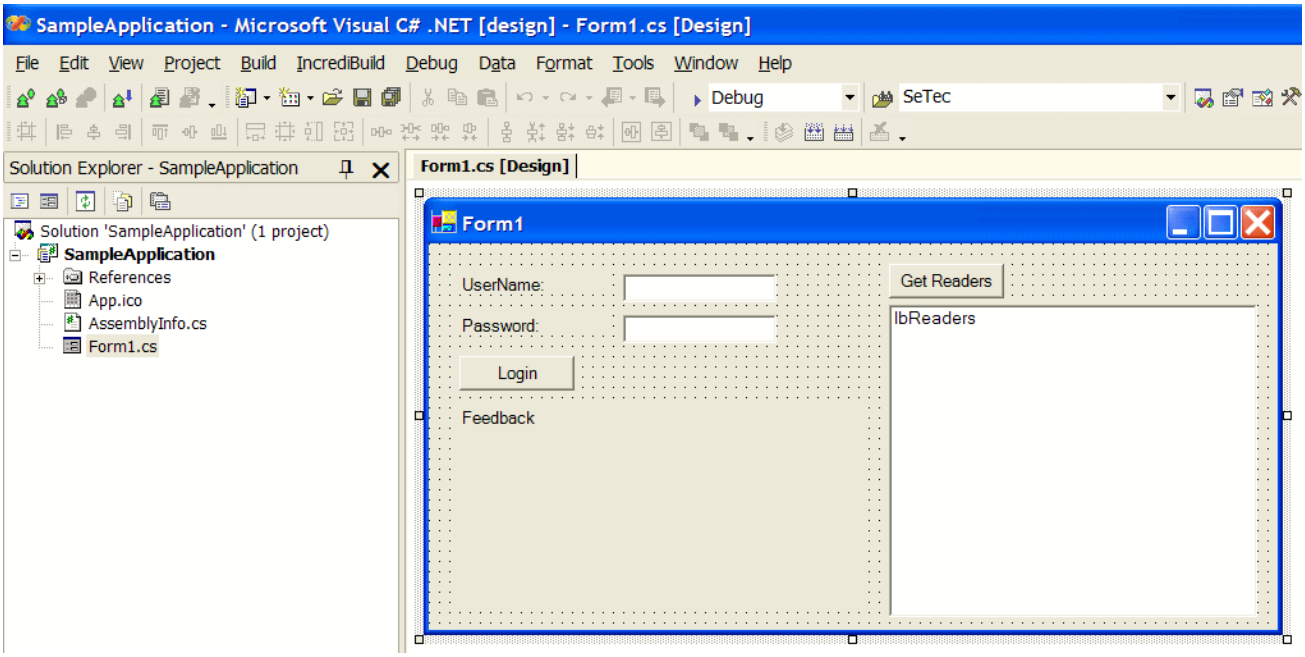
The completed program code for this example is shown next.

Completed Program Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using SampleApplication.G4T;
using System.ServiceModel.Security;
using System.ServiceModel;

namespace SampleApplication
{
    public partial class Form1 : Form
    {
        private SMSXMLWebServiceClient G4TAPI;

        public Form1()
        {
            G4TAPI = new SMSXMLWebServiceClient();

            InitializeComponent();
        }

        private void btnLogin_Click(object sender, EventArgs e)
        {
            try
            {
                G4TAPI.ClientCredentials.UserName.UserName = txtUserName.Text;
                G4TAPI.ClientCredentials.UserName.Password = txtPassword.Text.ToLower();

G4TAPI.ClientCredentials.ServiceCertificate.Authentication.CertificateValidationMode =
X509CertificateValidationMode.None;
                G4TAPI.G4TLogin();
                lblResult.Text = "You have successfully logged into the Web Service";
            }
            catch (FaultException fe)
            {
                lblResult.Text = "Sorry Login Failed";
                lblResult.Text = fe.Message;
            }
        }

        private void btnGetReaders_Click(object sender, EventArgs e)
        {
            XML_DeviceRequest MyRequest = new XML_DeviceRequest();
            MyRequest.DeviceType = enDeviceTypes.READER_DEVICE;
            try
            {
                XML_Device[] Devices = G4TAPI.G4TGetDevices(MyRequest);
                foreach (XML_Device device in Devices)
                {
                    lbReaders.Items.Add(device.DeviceDescription);
                }
            }
            catch (FaultException fe)
            {
                lblResult.Text = "Sorry Login Failed getting Readers";
                lblResult.Text = fe.Message;
            }
        }
    }
}
```

Step 1 – Design the Form

Begin by using Microsoft Visual Studio to create a new Visual C# project (Windows Application) called "SampleApplication". Design the form as shown in the following picture, using the component names in the auto-generated code shown in the previous section.



Step 2 – Add the G4T Service Reference

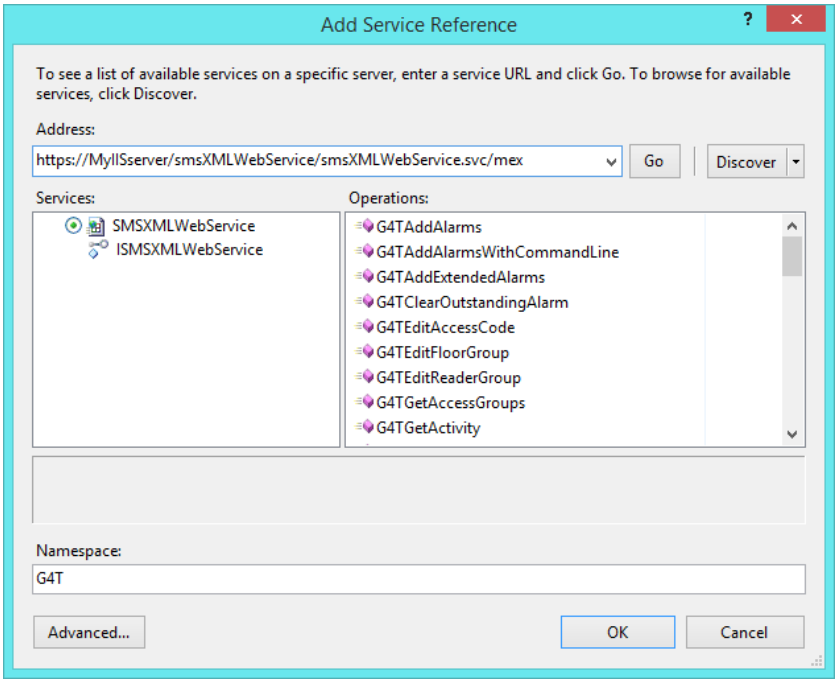
This step:

- Sets up the link to the web server, enabling the system to interrogate the service's WSDL (schema) file, which, in turn, describes the methods available through smsXMLWebService.
- Creates a C# wrapper class for the web service. The wrapper class contains the C# method definitions for the web service API commands (methods).
- Creates an "App Config file", which specifies how the program should connect to the web server. (Note that other languages may use a different mechanism.)

Carry out the following:

1. In the Solution Explorer tab, right-click on **SampleApplication** and choose **Add Service Reference**.

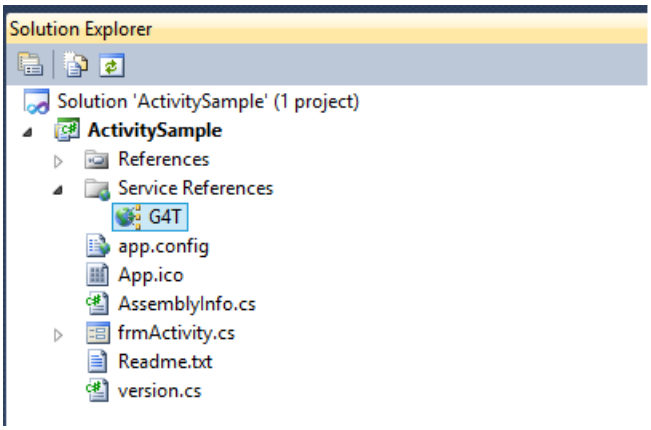
   The Add Service Reference window is displayed, as shown next.



2. Enter the URL of the smsXMLWebService web service and specify a Namespace of **G4T**. Click **Go** to check the URL. The format of the URL is:

   http://<Server Name>/smsXMLWebService/smsXMLWebService.svc (or https if encryption used)

3. Click **Advanced**, and check the **Generate asynchronous operations** checkbox. Click **OK** twice to display the following:



Step 3 – Add Namespace

1. In the code for Form1.cs, add the following namespace where shown:

```
using SampleApplication.G4T;
using System.ServiceModel.Security;
using System.ServiceModel;

namespace SampleApplication
{
```

Note: This specifies that our form will use the "G4T" namespace for all calls, saving us from having to prefix each API call with "SampleApplication.G4T".

2. Add the following statements where shown:

```
namespace SampleApplication
{
    public partial class Form1 : Form
    {
        private SMSXMLWebServiceClient G4TAPI;

        public Form1()
        {
            G4TAPI = new SMSXMLWebServiceClient();

            InitializeComponent();
        }
```

This enables the methods (API commands) on the G4T web service to be accessed.

Step 4 – Define Login Code

To log into Symmetry when the **Login** button is clicked, create the btnLogic_Click method by double clicking **Form1.cs** in the Solution Explorer, then double-clicking the **Login** button in the form designer. Modify the btnLogic_Click code as follows:

```
        private void btnLogin_Click(object sender, EventArgs e)
        {
            try
            {
                G4TAPI.ClientCredentials.UserName.UserName = txtUserName.Text;
                G4TAPI.ClientCredentials.UserName.Password = txtPassword.Text.ToLower();

G4TAPI.ClientCredentials.ServiceCertificate.Authentication.CertificateValidationMode =
X509CertificateValidationMode.None;
                G4TAPI.G4TLogin();
                lblResult.Text = "You have successfully logged into the Web Service";
            }
            catch (FaultException fe)
            {
                lblResult.Text = "Sorry Login Failed";
                lblResult.Text = fe.Message;
            }
        }
```

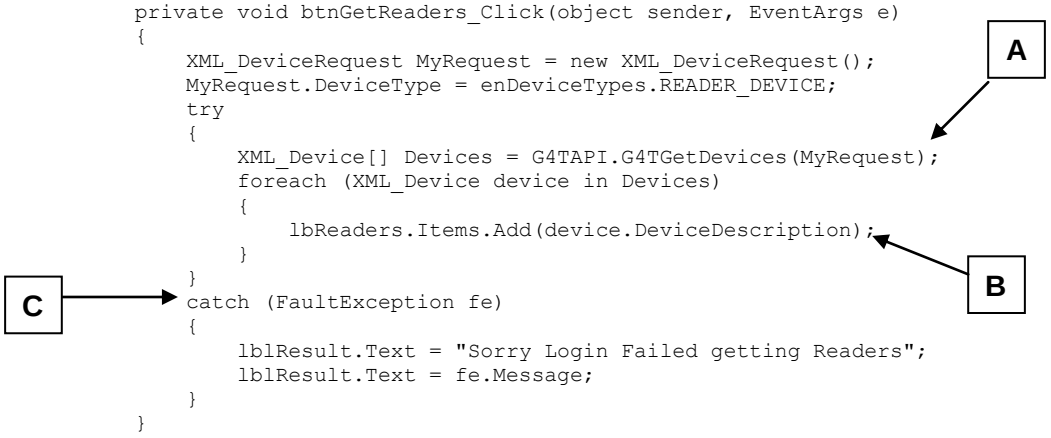
These statements cause the program to read the contents of the **UserName** and **Password** fields, then attempt to log in using the "G4TLogin" method (see Chapter 4). A catch statement is included to catch login exceptions.

Step 5 – Define Code for Get Readers Button

You now need to add the code that requests the reader device data when the **Get Readers** button is clicked.

1. Create the btnGetReaders_Click method by selecting the Form1.cs design window, then double-clicking the **Get Readers** button.

2. Add the following statements to the code.




```
private void btnGetReaders_Click(object sender, EventArgs e)
{
    XML_DeviceRequest MyRequest = new XML_DeviceRequest();
    MyRequest.DeviceType = enDeviceTypes.READER_DEVICE;
    try
    {
        XML_Device[] Devices = G4TAPI.G4TGetDevices(MyRequest);
        foreach (XML_Device device in Devices)
        {
            lbReaders.Items.Add(device.DeviceDescription);
        }
    }
    catch (FaultException fe)
    {
        lblResult.Text = "Sorry Login Failed getting Readers";
        lblResult.Text = fe.Message;
    }
}
```

Please note the following in the code:

A The "(MyRequest)" parameter specifies the type(s) of device. In this example it is defined as READER_DEVICE. If not specified, all devices are requested.

B The "device" has "DeviceDescription", "DeviceType" and "DeviceID" properties. All three could be displayed in the listbox by replacing "device.DeviceDescription" with:

```
(device.DeviceDescription + " - "+device.DeviceType.ToString() + " - "
+device.DeviceID.ToString());
```

C To identify login or reader retrieval errors.

Step 6 – Compile and Run Code

The code can now be compiled and run.

If you receive a “Sorry Login Failed” message when you attempt to log on, check the permissions you used at the web server machine during the installation of the XML Open Integration Module.

Chapter 4: Methods

About this Chapter

This chapter gives full details of each method call supported by the smsXMLWebService web service. For each method, a description is given, followed by details of the syntax, input type and response type.

Types appearing in more than one method are described at the end of the chapter, with appropriate cross-references. Standard base types used in methods are also described at the end of the chapter.

Sample applications are provided with this manual, showing working examples.

Syntax Example:

XML_AlarmStatus[ ] G4TClearOutstandingAlarm (XML_AlarmClearRequest[ ])

This example is for the G4TClearOutstandingAlarm method. The input type is shown in parentheses after the method name. The response type is shown before the method name. In this example, both the input and response types use an array of values, indicated by the square parentheses.

About Screen Permissions Needed

The description of each method includes the “Screen Permissions Needed”. This describes the permissions needed for the Symmetry user account that the method uses to access the Symmetry database, as described on page 19. Symmetry user accounts are set up from the Symmetry software client interface.

G4TLogin (page 89) can be used to perform a database login to check whether a username and password are valid.

G4TAddAlarms

Description

Allows a third-party alarm to be added to the Symmetry database.

Note: G4TAddAlarmsWithCommandLine (page 22) and G4TAddExtendedAlarms (page 24) encompass and extend the functionality provided by G4TAddAlarms. Therefore, for any new implementations, G4TAddAlarmsWithCommandLine or G4TAddExtendedAlarms should normally be used in preference to G4TAddAlarms.

Screen Permissions Needed

None

Syntax

G4TAddAlarms (XML_Alarm[ ])

Input Type

See XML_Alarm[ ] on page 76.

Response Type

None

G4TAddAlarmsWithCommandLine

Description

Allows a third-party alarm to be added to the Symmetry database. An application name or URL can be associated with the alarm, which is executed when the user clicks the **Image** button in the Acknowledge Alarms screen. The method encompasses and extends the functionality provided by G4TAddAlarms (page 21). G4TAddExtendedAlarms (page 24) provides another alternative method for providing alarms.

Alarms provided by G4TAddAlarmsWithCommandLine can be used to action trigger commands (using transaction text matching).

Screen Permissions Needed

None

Syntax

G4TAddAlarms (XML_AlarmCommandLine[ ])

Input Type

XML_AlarmCommandLine[ ]		
Property Name	**Type**	**Description**
AlarmID	int32	Gives the internal ID number of the alarm.
Who	who	Gives the first and last names of the person associated with the alarm. See page 102.
What	string	Gives a description of the alarm; e.g. "Wrong PIN".
Where	string	Gives the location where the alarm occurred.
Location	string	Gives the name of the chain from which the alarm originated.
CompanyID	int32	Specifies the ID of the company to which the alarm relates.
Company	string	Gives name of the company to which the alarm relates.
When	dateTime	Gives the date and time when alarm occurred.
AlarmStatus	enAlarmStatus	Gives the alarm status. enAlarmStatus can have one of the enumerated values specified on page 75.
AckDate	dateTime	Gives the date and time when the alarm was acknowledged.
AckUserName	string	Gives the name of the user who acknowledged the alarm.

Priority	int32	Gives the alarm priority.
ActivityCategory	enAlarmActivityCategory	Identifies the source of the alarm information. enAlarmActivityCategory can have one of the enumerated values specified on page 98.
EventOnly	boolean	Specifies whether the message is an alarm (false) or event (true).
Color	int32	Gives the number of the color assigned to the alarm.
DeviceTypeEnumerated	enDeviceTypes	Gives the type of device associated with the alarm. enDeviceTypes can have one of the enumerated values specified on page 99.
CommandLine	string	Specifies the application name or URL to associate with the alarm when the user clicks the **Image** button in the Acknowledge Alarms screen.
CommandLineParams	string	Specifies any parameters required for the application specified in CommandLine. Note: Alternatively, any parameters could be added to the CommandLine string.

Response Type

None

G4TAddExtendedAlarms

Description

Allows a third-party alarm to be added to the Symmetry database, together with associated image data. The image data may consist of multiple images. The associated images can be viewed at a Symmetry client by clicking the **Image** button in the "Home/Monitoring/Alarms" screen.

The method encompasses and extends the functionality provided by G4TAddAlarms (page 21). G4TAddAlarmsWithCommandLine (page 22) provides another alternative method for providing alarms.

Note that alarms provided by G4TAddExtendedAlarms cannot be used to action Symmetry trigger commands (using transaction text matching). Alarms provided by G4TAddAlarmsWithCommandLine (page 22) can be used to action trigger commands.

Screen Permissions Needed

None

Syntax

G4TAddExtendedAlarms (XML_ExtendedAlarm[ ])

Input Type

XML_ExtendedAlarm[ ]		
Property Name	**Type**	**Description**
Alarm	XML_Alarm	Specifies the alarm details. See page 76.
AlarmKeyFrame	XML_AlarmKeyFrame[ ]	Specifies the image data. XML_AlarmKeyFrame[ ] is a base64 encoded array of bytes. The image data should be JPEG to ensure that the data is not too large and to provide optimum performance.

Response Type

None

G4TClearOutstandingAlarm

Description

Clears or acknowledges an outstanding alarm. Outstanding alarms can be retrieved using G4TGetOutstandingAlarms (page 69).

Note: An alarm remains outstanding after it is acknowledged and until it has been cleared.

Note: Some alarms cannot be cleared until they have been reset (e.g. by closing the open door that caused the alarm).

Screen Permissions Needed

"Home/Monitoring/Alarms"

Syntax

XML_AlarmStatus[ ] G4TClearOutstandingAlarm (XML_AlarmClearRequest[ ])

Input Type

XML_AlarmClearRequest[ ]		
Property Name	**Type**	**Description**
AlarmID	int32	Specifies the alarm ID, as returned by G4TGetOutstandingAlarms.
Message	string	Comment, stating the reason for acknowledging or clearing the alarm; e.g. "Window left open".
AlarmAction	enAlarmAction	Specifies whether to acknowledge or clear the alarm. enAlarmAction can have one of the following enumerated values: Acknowledge Clear

Response Type

XML_AlarmStatus[ ]		
Property Name	**Type**	**Description**
AlarmID	int32	Gives the alarm ID of the cleared/acknowledged alarm.
AlarmActionStatus	enAlarmActionStatus	Gives the new status of the alarm. enAlarmActionStatus can have one of the following enumerated values: AcknowledgedRequiresReset Acknowledged Cleared Locked (This indicates that the alarm status is being modified by another user. The alarm was not cleared by G4TClearOutstandingAlarm.)

G4TEditAccessCode

Description

Enables the user to:

- Set up an new access code.
- Add or delete access rights from an existing access code.
- Change the name of an existing access code.
- Delete an existing access code.

The user can access only devices belonging to a company in their company group (as determined by the login username). Similarly, if a new access code is added, the specified company must be in the user's company group. Companies to which the user access are given by G4TGetCompanies (page 57).

Note: An access code cannot be changed or deleted if it is already assigned to another item, such as the access rights of a card holder or visitor. If this is attempted, an exception message is returned, which should be handled appropriately.

Screen Permissions Needed

"Operation/Times/Access Codes"

Syntax

G4TEditAccessCode (XML_EditAccessCodeDocument)

Input Type

XML_EditAccessCodeDocument		
Property Name	**Type**	**Description**
AccessCodeID	int32	Specifies the ID of the access code to add, edit or delete.
AccessCodeName	string	This is used to specify the name of a new access code. Alternatively, if you are editing an existing access code, you can use this property to specify a new name for the access code.
EditModeEnumerated	enEditMode	enEditMode can have one of the following enumerated values: • ADD – add a new access code, or add access rights to an existing access code. • DELETE_INDIVIDUAL – delete access rights from an existing access code. • DELETE_ALL – delete an access code.
CompanyID	int32	Specifies the ID of the company that will own the access code. This is used only when adding a new access code.

Readers	XML_EditReader[ ]	Specifies the reader and time code pairs to add or delete from the access code (see the following table). This is not used if enEditMode is DELETE_ALL.
ReaderGroups	XML_EditReaderGroup[ ]	Specifies the reader group and time code pairs to add or delete from the access code (see the following table). This is not used if enEditMode is DELETE_ALL.
FloorGroups	XML_EditFloorGroup[ ]	Specifies the floor group and time code pairs to add or delete from the access code (see the following table). This is not used if enEditMode is DELETE_ALL.
Areas	XML_EditArea[ ]	Specifies the area to add or delete from the access code (see the following table). This is not used if enEditMode is DELETE_ALL.

XML_EditReader[ ]

XML_EditReader[ ]		
Property Name	**Type**	**Description**
ReaderID	int32	Specifies the ID of an existing reader.
TimeCodeID	int32	Specifies the ID of the time code associated with the reader. This property is used only by G4TEditAccessCode.

XML_EditReaderGroup[ ]

XML_EditReaderGroup[ ]		
Property Name	**Type**	**Description**
ReaderGroupID	int32	Specifies the ID of an existing reader group.
TimeCodeID	int32	Specifies the ID of the time code associated with the reader group.

XML_EditFloorGroup[ ]

XML_EditFloorGroup[ ]		
Property Name	**Type**	**Description**
FloorGroupID	int32	Specifies the ID of an existing floor group.
TimeCodeID	int32	Specifies the ID of the time code associated with the floor group.

XML_EditArea[ ]

XML_EditFloorGroup[ ]		
Property Name	**Type**	**Description**
AreaID	int32	Specifies the ID of an existing area.

Response Type

None

G4TEditFloorGroup

Description

Enables the user to:

- Add existing floors to a new floor group.
- Add or delete floors from an existing floor group.
- Change the name of an existing floor group.
- Delete an existing floor group.

The user can access any floors, reader, company, or floor group, irrespective of the companies assigned to their company group (as determined by the login username).

Note: A floor group cannot be changed or deleted if it is already assigned to another item, such as the access rights of a card holder or visitor. If this is attempted, an exception message is returned, which should be handled appropriately.

Screen Permissions Needed

"Setup/Device Groups/Floor/Output"

Syntax

G4TEditFloorGroup (XML_EditFloorGroupDocument)

Input Type

XML_EditFloorGroupDocument		
Property Name	**Type**	**Description**
FloorGroupID	int32	Specifies the ID of the floor group to add, edit or delete.
FloorGroupName	string	This is used to specify the name of a new floor group. Alternatively, if you are editing an existing floor group, you can use this property to specify a new name for the floor group.
EditModeEnumerated	enEditMode	enEditMode can have one of the following enumerated values: • ADD – add a new floor group, or add floors to an existing floor group. • DELETE_INDIVIDUAL – delete floors from an existing floor group. • DELETE_ALL – delete a floor group.
CompanyID	int32	Specifies the ID of the company that will own the floor group. This is used only when adding a new floor group.
Floors	XML_EditFloor[ ]	Specifies the floors to add or delete from the floor group (see the following table). This is not used if enEditMode is DELETE_ALL.

XML_EditFloor[ ]

XML_EditFloor[ ]		
Property Name	**Type**	**Description**
FloorID	int32	Specifies the ID of an existing floor to add or delete from the floor group.
ElevatorReaderID	int32	Use this property to specify the ID of the reader to use for a new floor group. There can be only one elevator reader per floor group. The reader must be connected to the same elevator node that the floors are controlled by.

Response Type

None

G4TEditReaderGroup

Description

Enables the user to:

- Add existing readers to a new reader group.
- Add or delete readers from an existing reader group.
- Change the name of an existing reader group.
- Delete an existing reader group.

The user can access any readers, company or reader group, irrespective of the companies assigned to their company group (as determined by the login username).

Note: A reader group cannot be changed or deleted if it is already assigned to another item, such as the access rights of a card holder or visitor. If this is attempted, an exception message is returned, which should be handled appropriately.

Screen Permissions Needed

"Install/Access Control/Reader"

Syntax

G4TEditReaderGroup (XML_EditReaderGroupDocument)

Input Type

XML_EditReaderGroupDocument		
Property Name	**Type**	**Description**
ReaderGroupID	int32	Specifies the ID of the reader group to add, edit or delete.
ReaderGroupName	string	This is used to specify the name of a new reader group. Alternatively, if you are editing an existing reader group, you can use this property to specify a new name for the reader group.
EditModeEnumerated	enEditMode	enEditMode can have one of the following enumerated values: • ADD – add a new reader group, or add readers to an existing reader group. • DELETE_INDIVIDUAL – delete readers from an existing reader group. • DELETE_ALL – delete a reader group.
CompanyID	int32	Specifies the ID of the company that will own the reader group. This is used only when adding a new reader group.
Readers	XML_EditReader[ ]	Specifies the readers to add or delete from the reader group (see page 28). This is not used if enEditMode is DELETE_ALL.

Response Type

None

G4TGetAccessGroups

Description

Retrieves access codes defined in the Symmetry software and belonging to companies in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_AccessGroup[ ] G4TGetAccessGroups ( )

Input Type

None

Response Type

XML_AccessGroup[ ]		
Property Name	**Type**	**Description**
AccessGroupID	int32	Gives the ID of the access code being returned.
AccessGroupDescription	string	Gives the description of the access code being returned.
CompanyID	int32	Gives the ID of the company that owns the access code.

G4TGetActivity

Description

Retrieves one or more alarms and/or events. Alarms or events can be retrieved using one of a number of different search methods. For example, alarms from a specified ID can be retrieved, or events from a specified date and time.

The method retrieves alarms/events associated with a company in the user's company group (as determined by the login username), and those associated with no company. G4TGetCompanies (page 57) returns the companies in the user's company group.

Screen Permissions Needed

"Home/Monitoring/Activity"

Syntax

XML_Activity [ ] G4TGetActivity (XML_ActivityRequest)

Input Type

XML_ActivityRequest		
Property Name	**Type**	**Description**
ActivitySearchMethod	enActivitySearchMethod	Specifies the method to be used to retrieve alarms/events. enActivitySearchMethod can have one of the following enumerated values: Search from AlarmID Search from Date Include All Search by Who Search by What Search by Where Search by When Search by AlarmID Search by Device Type Search by Alarm Event Search by Cleared Flag Search by Card Holder ID Search by Card Number
SearchActivityFromTxnID	int32	If ActivitySearchMethod is "Search from AlarmID", this specifies the ID of the earliest alarm/event to retrieve. You may want to set this to be one greater than the AlarmID of the last alarm/event previously retrieved by the method. If ActivitySearchMethod is "Search by AlarmID", this specifies the ID of the single alarm/event to retrieve.

SearchActivityFromDateTime	dateTime	Specifies the date and time of the earliest alarm/event to retrieve. This is used only if ActivitySearchMethod is "Search from Date", "Search by Card Holder ID" or "Search by Card Number".
SearchActivityText	string	Specifies the name of a person, alarm/event message, location or alarm ID. Only alarms/events that match the selection are retrieved. This is used only if ActivitySearchMethod is "Search by Who" (e.g. Smith, Fred), "Search by What" (e.g. At Wrong Door [23]), "Search by Where" (e.g. Main Entrance) or "Search by AlarmID" (e.g. 567).
SearchActivityDateTime	dateTime	Specifies the date and time of the alarm/event to retrieve. This is used only if ActivitySearchMethod is "Search by When".
SearchActivityAlarmEventType	enAlarmEventType	Specifies whether to search for an event or alarm. This is used only if ActivitySearchMethod is "Search by Alarm Event". enAlarmEventType can have an enumerated value of "Event" or "Alarm". If "Search by Alarm Event" is not used, both alarms and events are retrieved.
SearchActivityDeviceType	enDeviceTypes	Specifies the type of device for which alarms/events are to be retrieved. This is used only if ActivitySearchMethod is "Search by Device Type".
SearchActivityClearedStatus	enClearedFlagStatus	Specifies whether to retrieve cleared or uncleared alarms. This is used only if ActivitySearchMethod is "Search by Cleared Flag". enClearedFlagStatus can have an enumerated value of "NotCleared" or "Cleared". If "Search by Cleared Flag" is not used, events and both cleared and uncleared alarms are retrieved.
SortOrder	enSortOrder	Specifies the order in which to return the alarms/events. enSortOrder can have an enumerated value of "DateTimeOfTxnAscending" (latest alarm/event first) or "DateTimeOfTxnDescending" (earliest alarm/event first).
SearchActivityToDateTime	dateTime	Specifies the date and time of the latest alarm/event to retrieve. This is used only if ActivitySearchMethod is "Search by Card Holder ID" or "Search by Card Number".
SearchActivityCardHolderID	int32	Specifies the ID of the card holder for which alarms/events are to be retrieved. This is used only if ActivitySearchMethod is "Search by Card Holder ID".

SearchActivityCardNumber	int32	Specifies the card number of the card holder for which alarms/events are to be retrieved. This is used only if ActivitySearchMethod is "Search by Card Number".
SearchActivityCustomerCode	int32	This is used only if ActivitySearchMethod is "Search by Card Number", and is applicable only to non-Homeland systems. Specify the customer code that is associated with the specified card number. If you omit the customer code, transactions for the specified card will be returned irrespective of the customer code.

Response Type

XML_Activity [ ]		
Property Name	**Type**	**Description**
AlarmID	int32	Gives the ID of the alarm/event being returned.
When	dateTime	Gives the date and time when the alarm/event occurred.
CompanyID	int32	Gives the ID of the company that the alarm/event belongs to (if known).
Company	string	Gives the name of the company that the alarm/event belongs to (if known).
Where	string	Gives the location where the alarm/event occurred.
ResponseMnemonic	int32	Gives the ID of the type of alarm/event. For example, 17238 is for "Granted Access", and 18546 if for "Access Requested". If required, you can find out a list of response mnemonics and their meaning from the ResponseTable of the Symmetry database.
TxnConditionName	string	Gives a description of the type of alarm/event.
Priority	int32	Specifies the alarm/event priority.
CardNumber	int32	Gives the card number associated with the alarm/event (if known).
Who	who	Provides the first and last names of the person associated with the alarm/event (if known). See page 102.
AckDate	dateTime	Gives the date and time the alarm was acknowledged (e.g. in the "Home/Monitoring/Alarms" screen).
AckUserName	string	Gives the name of the user who acknowledged the alarm.
ActivityCategory	enAlarmActivityCategory	Identifies the source of the alarm/event. enAlarmActivityCategory can have one of the enumerated values specified on page 98.

CardHolderID	int32	Gives the ID of the card holder associated with the alarm/event (provided only for card alarms/events).
CardIdentifier	string	Homeland only. Gives the UUID number of the card (for CMV-U cards only). For example, {78563412-bc9a-f0de-1234-56789abcdef0}.
CardNumberString	string	Gives the card number (as displayed in the Card Holders screen).
CustomerCode	int32	Gives the customer code of the card holder associated with the alarm/event (provided only for card alarms/events).
ChainID	int32	Gives the ID of the chain associated with the alarm/event (provided only for node alarms/events).
DoorControlUnitID	int32	Gives the ID of the door-control unit associated with the alarm/event (provided only for node alarms/events).
DeviceNumber	int32	Gives the DeviceNumber of the device associated with the alarm/event. Each device (such as a reader or monitor point) has a unique DeviceNumber for a particular DoorControlUnitID. For readers, DeviceNumber is 0 for odd-numbered readers and 1 for even-numbered readers.
DeviceID	int32	Gives the ID of the device associated with the alarm/event (if known), e.g. reader, node, etc.
DeviceType	int32	Gives an enumerated integer ID to indicate the device type (page 99). See also DeviceTypeEnumerated below.
AlarmEventType	int32	Gives an enumerated integer ID to indicate whether the record is an alarm or event.
AckedFlag	boolean	Specifies whether the alarm has been acknowledged. The boolean is always false for an event.
ResetFlag	boolean	Specifies whether the alarm has been reset. The boolean is always false for an event.
ClearedFlag	boolean	Specifies whether the alarm has been cleared. The boolean is always false for an event.
DeviceTypeEnumerated	enDeviceTypes	Gives the device type. enDeviceTypes can have one of the enumerated values specified on page 99.
TimeZoneCode	string	This returns a three-letter code that indicates the time zone in which the device that generated the alarm is located. The code can be defined in the LAN Chain Definition screen, or by TimeCodeZone in multimax.ini.

G4TGetActivityDocument

Description

Performs the same action as G4TGetActivity (see page 35), but also provides pagination support (see page 103).

Screen Permissions Needed

Same as G4TGetActivity.

Syntax

XML_ActivityDocument G4TGetActivityDocument (XML_ActivityRequest)

Input Type

Same as G4TGetActivity, with the addition of pagination properties.

Response Type

XML_ActivityDocument		
Property Name	**Type**	**Description**
Activity	XML_Activity[]	See page 37.
Pagination properties; see page 103.		

G4TGetAdminTypes

Description

Retrieves a meaningful description for each enumerated value representing a device type. For example, "Reader device" is returned for the default enumerated value of READER_DEVICE used in C# programs. This method is necessary, because the enumerated values may not be known or readily recognizable.

Note: This method is an extension of G4TGetDeviceTypes (page 66). It should be used in preference to G4TGetDeviceTypes, since it can return a description for any device type supported by the XML Open Integration Module.

See also G4TGetDevices (see page 61).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_DeviceType[ ] G4TGetAdminTypes ( )

Input Type

None

Response Type

See XML_DeviceType[ ] on page 99.

G4TGetAlarmActivityCategory

Description

Retrieves the names and IDs of alarm categories, as defined using G4TSetAlarmActivityCategory (page 94).

Screen Permissions Needed

None

Syntax

XML_AlarmActivityCategory[ ] G4TGetAlarmActivityCategory ( )

Input Type

None

Response Type

See XML_AlarmActivityCategory[ ] on page 94.

G4TGetBadgeFormats

Description

Retrieves the names and IDs of badge designs belonging to companies in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_BadgeFormat[ ] G4TGetBadgeFormats ( )

Input Type

None

Response Type

XML_BadgeFormat[ ]		
Property Name	**Type**	**Description**
BadgeFormatID	int32	Gives the ID of the badge design.
BadgeFormatDescription	string	Gives the description of the badge design being returned; e.g. "Maintenance staff".
CompanyID	int32	Gives the ID of the company that owns the badge design.

G4TGetCardHolderInformation

Description

Gets details of the card holders and visitors belonging to companies in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57). Optionally, the card access rights can be returned.

You can apply filters to select the card holders or visitors to return.

See also G4TGetCardHolderInformationDocument (page 53).

Screen Permissions Needed

"Home/Identity/Card Holders"

Syntax

XML_CardHolder[ ] G4TGetCardHolderInformation (XML_CardHolderRequest)

Input Type

XML_CardHolderRequest		
Property Name	**Type**	**Description**
SearchMethod	enSearchMethod	Specifies the primary search method for determining the card holders or visitors to return; e.g. "last name" or "card number". enSearchMethod can have one of the enumerated values specified on page 45. The Arguments property specifies the additional information required for the chosen search method. For example, if "last name" is used, Arguments specifies the card number(s) to return.
SearchCardStatus	enSearchCardStatus	Enables you to filter the cards returned according to card status. For example, you can choose to return only expired cards. enSearchCardStatus can have one of the following enumerated values: All (default) Active ForcedInactive Expired Lost NotIssued Inactive VisitorActive VisitorClosed VisitorPending

ActiveDateTime	dateTime	Enables you to filter the cards returned according to the active date and time. Only cards that are active on or after the specified date and time are returned. If not specified, the method returns cards that are active on or before InactiveDateTime.
InactiveDateTime	dateTime	Enables you to filter the cards returned according to the inactive date and time. Only cards that have expired on or before the specified date and time are returned. If not specified, the method returns cards that are active on or after ActiveDateTime.
SearchOperand	ensearchMethod	This is applicable only if SearchMethod is SEARCH_BY_LF_PERSONALDATA or SEARCH_BY_LF_VISITORDATA. It specifies whether to apply an AND or OR operand if two items of personal data are specified in Arguments (see page 45). For example, if one data item is "Interview", the other is "John Smith" and the operand is AND, only cards that have both sets of personal data are returned. enSearchMethod can have one of the enumerated values specified on page 45.
IncludeCardData	boolean	Specifies whether or not card information is to be included with the returned data (see CardInfo on page 47).
IncludePersonalData	boolean	Specifies whether or not personal information is to be included with the returned properties (see PersonalData on page 47).
IncludeImageData	boolean	Specifies whether or not face and signature image data are to be included with response properties (see FaceImage and SignatureImage on page 47).
IncludeCardAccessRights	boolean	Specifies whether or not card access rights are to be included with response properties (see AccessGroups, ReaderGroups and Readers on page 47).
Arguments[ ]	string	Specifies the character strings to filter the cards to return; e.g. the last name "Robinson". Up to four arguments can be specified, depending on the SearchMethod.

enSearchMethod

enSearchMethod	
Type	**Description**
enSearchMethod	Specifies the primary search method to be used for determining the cards to return. When used to specify the SearchMethod, enSearchMethod can have one of the following enumerated values: SEARCH_BY_LF_LASTNAME SEARCH_BY_LF_CARDNUMBER SEARCH_BY_LF_PERSONALDATA SEARCH_BY_LF_EMPLOYEEREF SEARCH_BY_LF_VISITORLASTNAME SEARCH_BY_LF_VISITORDATA SEARCH_BY_LF_CARDNUMBERSTRING The following are for future use: SEARCH_BY_LF_CARDHOLDER SEARCH_BY_LF_VISITORCARDNAME SEARCH_BY_LF_READER SEARCH_BY_LF_READERGROUP SEARCH_BY_LF_NODE SEARCH_BY_LF_CHAIN SEARCH_BY_LF_FLOORGROUP SEARCH_BY_LF_FLAGGEDBADGES SEARCH_BY_LF_CARDSNOTUSED SEARCH_BY_LF_IMPORTEDCARDS SEARCH_BY_LF_VISITORCARDNUMBER SEARCH_BY_LF_PERSONALDATAEX SEARCH_BY_LF_VISITORDATAEX SEARCH_BY_LF_PERSONALIDENTIFIER SEARCH_BY_LF_CARDNUMBERSTRING SEARCH_BY_LF_CUSTOMERCODE When used to specify the SearchOperand, enSearchMethod can have one of the following enumerated values: SEARCH_BY_LF_OPERANDAND SEARCH_BY_LF_OPERANDOR

Arguments[ ]

The number of array elements required for Arguments and their syntax depends on the SearchMethod property specified, as shown in the following table. For example, Arguments[ ] requires two array elements if SearchMethod is SEARCH_BY_LF_CARDNUMBER; one for the card number and the other for the customer/facility code.

SearchMethod (SEARCH_BY_LF..)	Arguments[ ]			
	No.	**Name**	**Example**	**Description**
LASTNAME or VISITORLASTNAME	1	Last name	Robinson	Specifies the card holder or visitor last name to be searched. The wildcard character "*" can be included.
CARDNUMBER	1 2	Card number Customer code	10-25 999999	Specifies the card numbers and customer/facility code to be searched. The wildcard characters "-" and "*" can be included for the card number. If the customer/facility code is not specified, the method returns card data irrespective of the customer/facility code.
EMPLOYEEREF	1 2	Employee ref Customer code	D10* 999999	Specifies the employee reference to be searched. The wildcard character "*" can be included. If the customer/facility code is not specified, the method returns card data irrespective of the customer/facility code.
PERSONALDATA or VISITORDATA	1 2 3 4	Data title 1 ID Data 1 Data title 2 ID Data 2	15 Interview 16 John Smith	Specifies one or two pairs of personal data. The IDs specify the personal data titles. In the example, ID 15 may identify the visitor data title "Purpose of Visit" and ID16 "Visitor for". To obtain the data title IDs, use G4TGetPersonalDataTitles (page 77). Elements 3 and 4 are optional. If specified, an AND or OR operation is carried out between Data 1 and Data 2, depending on the SearchOperand property.

Response Type

XML_CardHolder[ ]		
Property Name	**Type**	**Description**
CardHolderID	int32	Gives the Symmetry database CardID value of the card holder or visitor. (This is not the card number.) See XML_CardInfo[] CardID property to obtain the Symmetry database CardInfoID of the card.
LastName	string	Gives the person's last name.
FirstName	string	Gives the person's first name.
InitLet	string	Gives the person's middle initial.
EmployeeReference	string	Gives the employee reference (not applicable to visitors).
CompanyID	int32	Gives the ID number of the company that the card holder or visitor belongs to.

FaceImage	byte[ ]	Gives the face image data as a base64 encoded array of bytes.
SignatureImage	byte[ ]	Gives the signature image data as a base64 encoded array of bytes.
CardInfo	XML_CardInfo[ ]	Gives the details for each of the person's cards (see page 47). A visitor can have only one card.
PersonalData	XML_PersonalData[ ]	Gives the personal details of the card holder or visitor. See page 49.
AccessGroups	XML_CardAccessGroup[ ]	Gives details of the access codes assigned to the card. Normal and advanced access rights are included. See page 49.
ReaderGroups	XML_CardReaderGroup[ ]	Gives details of the reader group access rights assigned to the card. Normal and advanced access rights are included. See page 50.
Readers	XML_CardReader[ ]	Gives details of the reader access rights assigned to the card. Normal and advanced access rights are included. See page 51.
Properties	enCardHolderProperties[ ]	Specifies whether the card record returned includes face, signature and/or access rights data. This information may help to process the card record returned. Each array element of enCardHolderProperties can have one of the following three enumerated values: Face Signature AccessRights
IntrusionAreas	XML_CardIntrusionArea[ ]	Gives details of the area access rights assigned to the card. Normal and advanced access rights are included. See page 51.

XML_CardInfo[ ]

XML_CardInfo[ ]		
Property Name	**Type**	**Description**
CardID	int32	Gives the Symmetry database CardInfoID of the card. See XML_CardHolder[] CardHolderID property to obtain the Symmetry database CardID value.
CardNumber	int32	Gives the card number. **Note:** There may be two or more card records with the same card number, but each has a different ID. Multiple records for the same card number can occur if the card has different issue levels.
CardIssueLevel	unsignedByte	Gives the card issue level.
PIN	int32	Gives the PIN number of the card (used only for a 4-digit PIN; see ExtendedPIN).

Active	boolean	When TRUE, the card is active.
ActiveDate	dateTime	Gives the date that the card became active.
ExpiryDate	dateTime	Gives the date that the card becomes inactive.
InactiveComment	string	Gives the reason why the card became inactive.
Encryption	int32	Not relevant.
CustomerCode	int32	The card's customer code.
BadgeFormatID	int32	ID of the badge design used by the card. See G4TGetBadgeFormats on page 42.
IDSCode	string	The IDS code is the card holder's personal code, which can be to gain access to intrusion options at an M2150 intrusion reader, such as to arm or disarm an area.
CardIdentifier	string	Homeland only. Gives the UUID number of the card (for CMV-U cards only). For example, {78563412-bc9a-f0de-1234-56789abcdef0}.
CardNumberString	string	Gives the card number (as displayed in the Card Holders screen).
CardFormat	enCardFormat	This is the card type. EnCardFormat can have one of the following enumerated values: Standard SeiwgV1 (Obsolete) Legacy SeiwgGSC2_1 (Obsolete) PIV_UUID PIV_FASC_N SRSeries_15Digit SRSeries_10And12Digit CMV_U CMV_F Custom150 to Custom254 Custom150 to Custom254 are used for user-defined card formats in the "Install/System/Default Settings/Card Format" screen and correspond to the **Format ID** in that screen. For example, if the **Format ID** is 167, EnCardFormat should be Custom167.
AgencyCode	int16	Homeland only. Specifies the person's agency code.
CredentialSeriesCode	int16	Homeland only; PIV cards only. Single-digit number intended to reflect the current system configuration. Incremented when a major system configuration change takes place.
OrganisationAssociation	int16	Homeland; PIV cards only. Single-digit code specifying the association between the organization and the person as follows: 0 = Employee; 1 = Civil; 2 = Executive Staff; 3 = Uniformed Service; 4 = Contractor.

OrganisationCategory	int16	Homeland; PIV cards only. Single-digit code specifying the category of the source as follows: 1 = Federal Government; 2 = State Government; 3 = Commercial Enterprise; 4 = Foreign Government.
OrganisationIdentifier	int16	Homeland; PIV cards only. Four-digit code specifying origin of the card owner as follows: 1st digit = FIPS 95-2 Federal Agency/Sub-Agency Code; 2nd digit = State Code; 3rd digit = Company Code; 4th digit = Country Code.
SystemCode	int16	Homeland only. Specifies the person's system code. **Note:** If a system code of zero is required, specify a value of -1 during card import.
UnitIDCode	int16	This is for cards that have the Legacy card type. The code specifies the standard U.S. government Unit Identification Code for the site or facility that issued the card.
ExtendedPIN	string	Gives the PIN number of the card (used only for a PIN of more than 4 digits).

XML_PersonalData[ ]

XML_PersonalData[ ]		
Property Name	**Type**	**Description**
PersonalDataID	int32	Numeric value assigned to the personal data entry in the Personal tab of the "Home/Identity/Card Holders" or "Home/Identity/Visitors" screen **Note:** The first Personal Data entry appearing in the tab has an ID value of 0.
Title	string	Title of the data title; e.g. "Gender".
Data	string	Value of the personal data; e.g. "Male".

XML_CardAccessGroup[ ]

XML_CardAccessGroup[ ]		
Property Name	**Type**	**Description**
AccessGroupID	int32	Gives the ID of the access code.
AccessGroupDescription	string	Gives the name of the access code.
CompanyID	int32	Gives the ID of the company that the access code belongs to.
CardID	int32	Gives the internal database ID of the card. This may not be the same as the card number.

AccessID	int32	Gives the location where this access right is defined in the Card Holders screen. 0=Normal access rights, 1=Advanced1, 2=Advanced2, etc. up to 16=Advanced16.
ActiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal access rights this is always 1\1\1992.
InactiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal access rights this is always 2\31\2035.
Enabled	boolean	Returns true if this access right is currently enabled for the card.
Valid	boolean	Returns true if this access right has been downloaded to nodes.
ReaderGroups	XML_CardReaderGroup[]	Gives details of the reader group access rights assigned to the access code.
Readers	XML_CardReader[]	Gives details of the reader access rights assigned to the access code.

XML_CardReaderGroup[ ]

XML_CardReaderGroup[ ]		
Property Name	**Type**	**Description**
ReaderGroupID	int32	Gives the ID of the reader group.
ReaderGroupDescription	string	Gives the name of the reader group.
TimeCodeID	int32	Gives the ID of the time code assigned to the reader group.
TimeCodeDescription	string	Gives the name of the time code assigned to the reader group.
CompanyID	int32	Gives the ID of the company that the reader group and time code belong to. This may be different from the company that owns the card.
CardID	int32	Gives the internal database ID of the card. This may not be the same as the card number.
AccessID	int32	Gives the location where this access right is defined in the Card Holders screen. 0=Normal access rights, 1=Advanced1, 2=Advanced2, etc. up to 16=Advanced16.
ActiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal rights this is the system start date (e.g. 1\1\1992).
InactiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal rights this is the system end date (e.g. 2\31\2035).

Enabled	boolean	Returns true if this access right is currently enabled for the card.
Valid	boolean	Returns true if this access right has been downloaded to nodes.

XML_CardReader[ ]

XML_CardReader[ ]		
Property Name	**Type**	**Description**
ReaderID	int32	Gives the ID of the reader.
ReaderDescription	string	Gives the name of the reader.
TimeCodeID	int32	Gives the ID of the time code assigned to the reader.
TimeCodeDescription	string	Gives the name of the time code assigned to the reader.
CompanyID	int32	Gives the ID of the company that the reader and time code belong to.
CardID	int32	Gives the internal database ID of the card. This may not be the same as the card number.
AccessID	int32	Gives the location where this access right is defined in the Card Holders screen. 0=Normal access rights, 1=Advanced1, 2=Advanced2, etc. up to 16=Advanced16.
ActiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal rights this is the system start date (e.g. 1\1\1992).
InactiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal rights this is the system end date (e.g. 2\31\2035).
Enabled	boolean	Returns true if this access right is currently enabled for the card.
Valid	boolean	Returns true if this access right has been downloaded to nodes.

XML_CardIntrusionArea[ ]

XML_CardIntrusionArea[ ]		
Property Name	**Type**	**Description**
AreaID	int32	Gives the ID of the area.
AreaDescription	string	Gives the name of the area.
CompanyID	int32	Gives the ID of the company that the area belongs to.
CardID	int32	Gives the internal database ID of the card. This may not be the same as the card number.

AccessID	int32	Gives the location where this access right is defined in the Card Holders screen. 0=Normal access rights, 1=Advanced1, 2=Advanced2, etc. up to 16=Advanced16.
ActiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal rights this is the system start date (e.g. 1\1\1992).
InactiveDate	dateTime	For advanced access rights, this gives the date and time when the access right is active. For normal rights this is the system end date (e.g. 2\31\2035).
Enabled	boolean	Returns true if this access right is currently enabled for the card.
Valid	boolean	Returns true if this access right has been downloaded to nodes.

G4TGetCardHolderInformationDocument

Description

Performs the same action as G4TGetCardHolderInformation (see page 43), but also provides pagination support (see page 103).

Note: For performance reasons, the search method SEARCH_BY_LF_CARDNUMBER is not supported for G4TGetCardHolderInformationDocument when pagination is used (i.e. UsePagination is true).

Screen Permissions Needed

Same as G4TGetCardHolderInformation.

Syntax

XML_CardHolderDocument G4TGetCardHolderInformationDocument (XML_CardHolderRequest)

Input Type

Same as G4TGetCardHolderInformation, with the addition of pagination properties.

Response Type

XML_CardHolderDocument		
Property Name	**Type**	**Description**
Card Holder	XML_CardHolder[]	See page 46.
Pagination properties; see page 103.		

G4TGetCommands

Description

Retrieves the set of commands of a specified type (e.g. manual commands), applying to a particular device defined in the Symmetry software.

Note: Not all devices of the same type have the same command set. For example, fingerprint readers have different commands from other reader types.

To send a command to a device, use G4TSendDeviceCommand (see page 91).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_DeviceCommand[ ] G4TGetCommands (XML_GetDeviceCommandRequest)

Input Type

XML_GetDeviceCommandRequest		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Specifies the type of device to return; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Specifies the ID of the device to return.
CommandType	enCommandType	Specifies the type of command to return; e.g. manual command. enCommandType can have one of the enumerated values specified on page 98. **Note:** It is a good idea not to use the enumerated value of ALL_COMMAND for this method, otherwise the type of the commands returned will not be known.

Response Type

XML_DeviceCommand[ ]		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Gives the type of device that the command relates to; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Gives the ID of the device returned. The value is greater than zero.
DeviceDescription	string	Gives the description of the device; e.g. "Entrance PIR".
Commands	XML_Command[ ]	Gives the types of commands that can be sent to the device, as described next.

XML_Command[ ]

XML_Command[ ]		
Property Name	**Type**	**Description**
CommandCode	int32	Gives the identification number for the command.
CommandDescription	string	Gives the description of the command; e.g. Grant Access.
TailStructureNumber	int32	Gives additional information about the command that must be provided when sending the command with the G4TSendDeviceCommand method (see page 91).
TailInfoOption	string	Gives additional information about the command that must be provided when sending the command with the G4TSendDeviceCommand method (see page 91).

G4TGetCommandTypes

Description

Retrieves a meaningful description for each enumerated value representing a command type. For example, "Manual Command" is returned for the default enumerated value of MANUAL_COMMAND used in C# programs.

This method is necessary, because the enumerated values may not be known or readily recognizable.

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_CommandType[ ] G4TGetCommandTypes ( )

Input Type

None

Response Type

XML_CommandType[ ]		
Property Name	**Type**	**Description**
CommandType	enCommandType	Specifies the type of command being returned; e.g. manual command. enCommandType can have one of the enumerated values specified on page 98.
CommandDescription	string	Gives the description of the command type being returned; e.g. Manual Command.

G4TGetCompanies

Description

Retrieves the IDs of all companies to which the user has access through their company group.

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_Company[ ] G4TGetCompanies ( )

Input Type

None

Response Type

XML_Company[ ]		
Property Name	**Type**	**Description**
CompanyID	int32	Gives the ID of the company being returned.
CompanyDescription	string	Gives the description of the company being returned.

G4TGetCurrentAntipassbackZone

Description

Returns a card's current antipassback zone and last reader used.

This method is appropriate for zonal antipassback only.

Screen Permissions Needed

"Home/Identity/Card Holders"

Syntax

XML_CurrentAntiPassbackForCardHolder G4TGetCurrentAntipassbackZone (XML_GetCurrentAntipassbackZoneRequest)

Input Type

XML_GetCurrentAntipassbackZoneRequest		
Property Name	**Type**	**Description**
CardHolderID	int32	Gives the unique ID of the card holder or visitor. This can be obtained using G4TGetCardHolderInformation (page 43).

Response Type

XML_CurrentAntiPassbackForCardHolder		
Property Name	**Type**	**Description**
AntiPassbackZone	int32	Gives the antipassback zone in which the card is currently located.
ReaderDescription	string	Gives the description of the reader that was last used. This information can also be obtained by using G4TGetLastReaderUsedForCardHolder (page 72).
ReaderID	int32	Gives the unique ID of the reader that was last used.

G4TGetDeviceAlarms

Description

Determines how messages from devices are handled by the Symmetry software. The three options are:

- Alarm – Device message causes an alarm.
- Event – Device message is logged but does not cause an alarm.
- Disabled – Device message is ignored.

The user can access only devices belonging to a company in their company group (as determined by the login username). Companies to which the user access are given by G4TGetCompanies (page 57).

Note: You cannot use the XML Open Integration Module to change the way the messages are handled.

Screen Permissions Needed

"Operation/Alarms/Definitions"

Syntax

XML_DeviceAlarmDef[ ] G4TGetDeviceAlarms (XML_DeviceRequest[ ])

Input Type

XML_DeviceRequest[ ]		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Specifies the type of device whose message conditions are to be returned; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Specifies the ID of the device to return.

Response Type

XML_DeviceAlarmDef[ ]		
Property Name	**Type**	**Description**
AlarmDeviceType	enDeviceTypes	Gives the type of device relating to the alarm condition; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.

AlarmDeviceID	int32	Gives the ID of the device relating to the alarm condition.
AlarmDeviceDescription	string	Gives the description of the device relating to the alarm, e.g. "Entrance Reader".
Alarms	XML_AlarmDef[ ]	Gives the details of the alarm relating to this device, as described next.

XML_AlarmDef[ ]

XML_AlarmDef[ ]		
Property Name	**Type**	**Description**
AlarmID	int32	Gives the ID of an alarm within the Alarm tab of the device definition screen (e.g. Reader Definition screen).
AlarmCode	int32	Not used.
AlarmDescription	string	Gives the alarm description; e.g. "Monitor point in alarm".
AlarmMode	enAlarmMode	Gives the current action for the message originating from the device. enAlarmMode can have one of the following enumerated values: DISABLED ALARM EVENT

G4TGetDevices

Description

Retrieves the description and ID of all devices of the specified type defined in the Symmetry software and belonging to companies in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_Device[ ] G4TGetDevices (XML_DeviceRequest)

Input Type

XML_DeviceRequest		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Specifies the type of device to return; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99. **Note:** Specifying the "door" or "reader" enumerated value results in the same response for DeviceDescription.
DeviceID	int32	Not used.

Response Type

XML_Device[ ]		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Gives the type of device returned; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Gives the ID of the device returned.
DeviceDescription	string	Gives the return device description; e.g. "Entrance Reader".
ElevatorReaderID	int32	Gives the ID of the associated elevator reader. This is applicable only if DeviceType is "FLOOR_DEVICE".
CompanyID	int32	Specifies the ID of the company that owns the device.

G4TGetDeviceStatus

Description

Lists the status of a device belonging to a company in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57).

To determine the device types and IDs available in the Symmetry software, use G4TGetDevices (see page 61).

Note: G4TGetDoorStatus (page 67) provides speed benefits over this method if you frequently want to obtain the status of doors.

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_DeviceStatus[ ] G4TGetDeviceStatus (XML_DeviceRequest[ ])

Input Type

XML_DeviceRequest[ ]		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Specifies the type of device whose status is to be returned; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99. **Note:** Unlike in the Symmetry software, doors and readers do not return the same status data.
DeviceID	int32	Specifies the ID of the individual device whose status is to be returned.

Response Type

XML_DeviceStatus[ ]		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Gives the type of device returned; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Gives the ID of the device returned.
DeviceDescription	string	Gives the device description.
Status	XML_Status[ ]	Gives the status data for the device, as described next.

XML_Status[ ]

XML_Status[ ]		
Property Name	**Type**	**Description**
StatusID	int32	Not used.
StatusCode	string	Gives a three character mnemonic code to identify the status returned (see next section).
StatusDescription	string	Gives the description of the current status (see next section).

StatusCode	StatusDescription
X	Problem getting Status
Reader	
RsD	Disabled
RsE	Enabled
RsM	Card Only
RsP	Card + Pin
RsU	Customer Code - No Store
RsS	Customer Code - Stored
RsN	No Reader
RsKCI	Keycard On - In
RsKCO	Keycard On - Out
RsKCF	Keycard Off
RsCC + CardCommand	Card Command On
RsCCN	Card Command Off
RsXMF	User Code Off
RsXMC	User Code On - Code Only
RsXMP	User Code On - Code + PIN
RsFPF	Fingerprint off
RsFPO	Single fingerprint
RsFPT	Two fingerprints
RsPCF	PC Door Control Off
RsRSO	Random Search On
RsPCO	PC Door Control On
RsPCF	PC Door Control Off
RSM	Card Only

RSP	Card + Pin
RSU	Customer Code Only - No Store
RSS	Customer Code Only - Stored
RSN	No Reader
Door Status	
DsN	Closed
DsA	Open
DsS	Cable Fault Short
DsO	Cable Fault Open
DsL	Locked
DsU	Unlocked
Output Relay	
OsF	Secure Access or Output Off
OsO	Free Access or Output On
OsP	Grant Access or Output Pulse
Alarm Panel Node	
F	Output Off
N	Output On
Node	
VfS0	Node + Decoded Node Version
Monitor Point	
ASN	Normal
ASA	Alarm
ASS	Cable Fault Short
ASO	Cable Fault Open
E	Enabled
D	Disabled
DsC	Normal or Alarm
Dsc	Cable Fault Short
Dso	Cable Fault Open
Dst	Tamper Alarm or Tamper & Point in Alarm
DsT	Tamper & Point in Alarm or Tamper Alarm
PSC	Normal or Alarm
PSc	Cable Fault Short
PSo	Cable Fault Open

PSt	Tamper Alarm or Tamper & Point in Alarm
PST	Tamper & Point in Alarm or Tamper Alarm
PsC	Normal or Alarm
Psc	Cable Fault Short
Pso	Cable Fault Open
Pst	Tamper Alarm or Tamper & Point in Alarm
PsT	Tamper & Point in Alarm or Tamper Alarm

G4TGetDeviceTypes

Description

Retrieves a meaningful description for each enumerated value representing a device type. For example, "Reader device" is returned for the default enumerated value of READER_DEVICE used in C# programs. This method is necessary, because the enumerated values may not be known or readily recognizable.

Note: This method is provided for compatibility with existing applications that use the XML Open Integration Module. The method does not return descriptions for access codes, scheduled commands and trigger commands. G4TGetAdminTypes (page 39) should be used for any new applications.

See also G4TGetDevices (see page 61).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_DeviceType[ ] G4TGetDeviceTypes ( )

Input Type

None

Response Type

See XML_DeviceType[ ] on page 99.

G4TGetDoorStatus

Description

Lists the status of a door belonging to a company in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57).

To determine the IDs available in the Symmetry software, use G4TGetDevices (see page 61).

This method provides speed benefits over G4TGetDeviceStatus (page 62) if you frequently want to find the status of doors. A cache is used on the Symmetry server to facilitate this method.

Note: To use this method, **Show Door Status on Graphics** must be selected in the "Maintenance/User & Preferences/System Preferences" screen of the Symmetry software.

Note: To use this method, port 65101 must be open on the Symmetry server. Add the port to the firewall exceptions.

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_DeviceStatus[ ] G4TGetDoorStatus (XML_DeviceRequest[ ])

Input Type

See XML_DeviceRequest[ ] on page 62. DeviceType must be DOOR_DEVICE.

Response Type

See XML_DeviceStatus[ ] on page 62.

G4TGetImportCardHolderDataStatus

Description

This method returns the current status of a previous card import operation executed using G4TImportCardHolderData (page 83). For example, you can use it to confirm that data not imported immediately (and therefore held in the Symmetry import database) has subsequently been imported successfully into Symmetry main database.

Screen Permissions Needed

"Operation/Data/Data Import"

Syntax

XML_ImportDataCardHolderStatusDocument G4TGetImportCardHolderDataStatus (XML_ImportDataCardHolderStatusDocument)

Input Type

XML_ImportDataCardHolderStatusDocument		
Property Name	**Type**	**Description**
BatchReference	string	Specifies the batch reference assigned by the associated G4TImportCardHolderData method (see page 88).
CardHolderImportDataStatus	XML_ImportDataCardHolderStatus[ ]	Not used.

Response Type

XML_ImportDataCardHolderStatusDocument		
Property Name	**Type**	**Description**
BatchReference	string	Gives the batch reference number for the import operation.
CardHolderImportDataStatus	XML_ImportDataCardHolderStatus[ ]	Returns details of the card holder or visitor and the current status of the import operation. See page 102.

G4TGetImportRecordRequests

Description

Retrieves a meaningful description for each enumerated value representing a type of import operation. For example, "Add Card" is returned for the default enumerated value of AddCard used in C# programs.

This method is necessary, because the enumerated values may not be known or readily recognizable.

Screen Permissions Needed

"Operation/Data/Data Import"

Syntax

XML_ImportRecordRequest[ ] G4TGetImportRecordRequests ( )

Input Type

None

Response Type

XML_ImportRecordRequest[ ]		
Property Name	**Type**	**Description**
ImportRecordRequest	enImportRecordRequest	Gives the import request type returned; e.g. "add card" or "modify card". The property can have one of the enumerated values specified on page 100.
ImportRecordRequestDescription	string	Gives the text description of the enumerated value.

G4TGetLastCardTransactionsForAReader

Description

You can use this method to obtain the details of the last transactions performed at a selected reader, including the card number and ID associated with each transaction. You can choose the number of transactions to return.

You can use this method to determine the card holders or visitors who last used a specified reader. See also G4TGetLastCardTransactionsForAReaderGroup (page 71).

Screen Permissions Needed

"Home/Identity/Card Holders"

Syntax

XML_CardTransaction[ ] G4TGetLastCardTransactionsForAReader
(XML_LastCardTransactionsForAReaderRequest)

Input Type

XML_LastCardTransactionsForAReaderRequest		
Property Name	**Type**	**Description**
ReaderID	int32	Specifies the unique ID of the reader. This can be obtained using G4TGetDevices (page 61).
NumberOfTransactions	int32	Specifies the number of transactions to examine.

Response Type

XML_CardTransaction[ ]		
Property Name	**Type**	**Description**
CardHolderID	int32	Gives the unique ID of the card holder or visitor. This can be obtained using G4TGetCardHolderInformation (page 43).
CardNumber	int32	Gives the person's card number.
DateTimeOfTxn	dateTime	Gives the date and time of the transaction (local time at reader).
TimeZoneCode	string	If ShowTimeZoneCode is set to 1 in the multimax.ini file, this returns a three-letter code that indicates the time zone in which the reader is located. The code can be defined in the LAN Chain Definition screen, or by TimeCodeZone in multimax.ini.
CustomerCode	int32	Non-Homeland only. Specifies the card's customer code.

G4TGetLastCardTransactionsForAReaderGroup

Description

You can use this method to obtain the details of the last transactions performed at readers in a selected reader group, including the card number and ID associated with each transaction. You can choose the number of transactions to return (this is the total number of transactions to return, not the number of transactions per reader).

You can use this method to determine the card holders or visitors who last used readers in a specified reader group. See also G4TGetLastCardTransactionsForAReader (page 70).

Screen Permissions Needed

"Home/Identity/Card Holders"

Syntax

XML_CardTransaction[ ] G4TGetLastCardTransactionsForAReaderGroup (XML_LastCardTransactionsForAReaderGroupRequest)

Input Type

XML_LastCardTransactionsForAReaderGroupRequest		
Property Name	**Type**	**Description**
ReaderGroupID	int32	Specifies the unique ID of the reader group. This can be obtained using G4TGetDevices (page 61).
NumberOfTransactions	int32	Specifies the number of transactions to examine.

Response Type

The response type is XML_CardTransaction[ ]. See page 70.

G4TGetLastReaderUsedForCardHolder

Description

Obtains the ID and description of last reader that granted access to a specified card holder or visitor. This method could be used to determine whether a card holder or visitor is still on site.

Note: You can use the alternative method, G4TGetLastReadersUsedForCardHolder (notice the "s" after "Reader") to obtain the ID and description of the last "n" readers used (where "n" is a number). See page 73.

Screen Permissions Needed

"Home/Identity/Card Holders" (even if retrieving visitors)

Syntax

XML_LastReaderUsedForCardHolder G4TGetLastReaderUsedForCardHolder (XML_LastReaderUsedForCardHolderRequest)

Input Type

XML_LastReaderUsedForCardHolderRequest		
Property Name	**Type**	**Description**
CardHolderID	int32	Specifies the unique ID of the card holder or visitor. This can be obtained using G4TGetCardHolderInformation (page 43).

Response Type

XML_LastReaderUsedForCardHolder		
Property Name	**Type**	**Description**
ReaderID	int32	Gives the unique ID of the reader that was last used. G4TGetDevices (page 61) can be used to obtain the ID and description of each reader.
ReaderDescription	string	Gives the description of the reader that was last used. This information can also be obtained by using ReaderID and G4TGetDevices (page 61).
DateTimeOfTxn	dateTime	Gives the date and time of the transaction (local time at reader).
TimeZoneCode	string	If ShowTimeZoneCode is set to 1 in the multimax.ini file, this returns a three-letter code that indicates the time zone in which the reader is located. The code can be defined in the LAN Chain Definition screen, or by TimeCodeZone in multimax.ini.

G4TGetLastReadersUsedForCardHolder

Description

Obtains the ID and description of the last readers that granted access to a specified card holder or visitor. You can choose the number of transactions to examine. This method could be used to determine the route taken by a card holder or visitor.

Screen Permissions Needed

"Home/Identity/Card Holders" (even if retrieving visitors)

Syntax

XML_LastReaderUsedForCardHolder[ ] G4TGetLastReadersUsedForCardHolder (XML_LastReadersUsedForCardHolderRequest)

Input Type

XML_LastReadersUsedForCardHolderRequest		
Property Name	**Type**	**Description**
CardHolderID	int32	Specifies the unique ID of the card holder or visitor. This can be obtained using G4TGetCardHolderInformation (page 43).
NumberOfTransactions	int32	Specifies the number of transactions to examine.

Response Type

The response type is XML_LastReaderUsedForCardHolder[ ].

See XML_LastReaderUsedForCardHolder on page 72.

G4TGetOutstandingAlarms

Description

Retrieves alarms that have not been cleared and either belong to companies in the user's company group (as determined by their login username), or belong to no company. Companies to which the user has access are given by G4TGetCompanies (page 57).

To use this method, the user must have access permissions to the Alarms screen in the Symmetry software.

Note: This method can also return alarms that are not associated with any company (e.g. Hard Disk Full Warning).

See also G4TClearOutstandingAlarm (page 25), G4TGetDeviceAlarms (page 57), G4TAddAlarms (page 21) and G4TAddAlarmsWithCommandLine (page 22).

See also G4TGetOutstandingAlarmsDocument (page 77).

Screen Permissions Needed

"Home/Monitoring/Alarms"

Syntax

XML_Alarm[ ] G4TGetOutstandingAlarms (XML_AlarmRequest)

Input Type

XML_AlarmRequest		
Property Name	**Type**	**Description**
AlarmSearchMethod	enAlarmSearchMethod	Specifies the filter criterion to be used for determining the outstanding alarm data to return; e.g. "who caused the alarm" or "where did the alarm occur". If not specified, returns all outstanding alarms. enAlarmSearchMethod can have one of the following enumerated values: Include All Search by Who Search by What Search by Where Search by When Search by AlarmID Search by Alarm Status Search by Device Type Search by AlarmID could be used if known from the response properties of a previous use of G4TGetOutstandingAlarms.

SearchAlarmText	string	This is relevant only when searching by who, what or where. If searching by who, the property specifies the name of the person who caused the alarm. The wildcard character "*" can be included in the string. e.g. "Clark*" could be specified. If searching by what, the property specifies the alarm text, such as Wrong PIN. If searching by where, the property specifies the device name, such as "Entrance Reader".
SearchAlarmDateTime	dateTime	Used only when searching by when. Specifies the date and time of outstanding alarms to return. Only alarms with an exact date/time match are returned.
SearchAlarmStatus	enAlarmStatus	Used only when searching by alarm status. Specifies the alarm status type to return from the search. enAlarmStatus can have one of the enumerated values specified on page 75.
SearchAlarmDeviceType	enDeviceTypes	Used only when searching by device type. Specifies the type of device to be searched for alarms; e.g. reader or monitor point. enDeviceTypes can have one of the enumerated values specified on page 99. **Note:** Door and reader alarms are different; e.g. "Door Forced" for a door or "Wrong PIN" for a reader.

enAlarmStatus

enAlarmStatus	
Type	**Description**
enAlarmStatus	Specifies the alarm status type. enAlarmStatus can have one of the following enumerated values: NotInAlarm InAlarm AcknowledgedButNotReset AcknowledgedAndReset

Response Type

XML_Alarm[ ]		
Property Name	**Type**	**Description**
AlarmID	int32	Gives the internal ID number of the alarm.
Who	who	Gives the first and last names of the person associated with the alarm. See page 102.
What	string	Gives a description of the alarm; e.g. "Wrong PIN".
Where	string	Gives the location where the alarm occurred.
Location	string	Gives the name of the chain from which the alarm originated.
CompanyID	int32	Specifies the ID of the company to which the alarm relates
Company	string	Gives name of the company to which the alarm relates.
When	dateTime	GIves the date and time when alarm occurred.
AlarmStatus	enAlarmStatus	Gives the alarm status type returned. enAlarmStatus can have one of the enumerated values specified on page 75.
AckDate	dateTime	GIves the date and time when the alarm was acknowledged.
AckUserName	string	Gives the name of the user who acknowledged the alarm.
Priority	int32	Gives the alarm priority.
ActivityCategory	enAlarmActivityCategory	Identifies the source of the alarm information. enAlarmActivityCategory can have one of the enumerated values specified on page 98.
EventOnly	boolean	Specifies whether the message being inputted is an alarm (false) or event (true). No value is returned for this property by G4TGetOutstandingAlarms (relevant only for G4TAddAlarms).
Color	int32	Gives the number of the color assigned to the alarm.
CardIdentifier	string	Homeland only. Gives the UUID number of the card (for CMV-U cards only). For example, {78563412-bc9a-f0de-1234-56789abcdef0}.
CardNumberString	string	Gives the card number (as displayed in the Card Holders screen).
DeviceTypeEnumerated	enDeviceTypes	Gives the device type. enDeviceTypes can have one of the enumerated values specified on page 99.

G4TGetOutstandingAlarmsDocument

Description

Performs the same action as G4TGetOutstandingAlarms (see page 74), but also provides pagination support (see page 103).

Screen Permissions Needed

Same as G4TGetOutstandingAlarms.

Syntax

XML_AlarmDocument G4TGetOutstandingAlarmsDocument (XML_AlarmRequest)

Input Type

Same as G4TGetOutstandingAlarms, with the addition of pagination properties.

Response Type

XML_AlarmDocument		
Property Name	**Type**	**Description**
Alarm	XML_Alarm[]	See page 76.
Pagination properties; see page 103.		

G4TGetPersonalDataTitles

Description

Retrieves the card holder or visitor personal data titles for a specified company in the user's company group (as determined by their login username). The method allows data title IDs to be obtained in order to search the Symmetry database for card holder or visitor data using G4TGetCardHolderInformation (see page 43).

To determine the company IDs required for this command, use G4TGetCompanies (see page 57).

Note: The personal data titles are set up in the "Setup/Identity/Personal Data/Card Holder Titles" and "Setup/Identity/Personal Data/Visitor Titles" screens in the Symmetry software.

Screen Permissions Needed

None

Syntax

XML_DataTitle[ ] G4TGetPersonalDataTitles (XML_DataTitleRequest)

Input Type

XML_DataTitleRequest		
Property Name	**Type**	**Description**
CompanyID	int32	Specifies the ID of the company.
Visitors	boolean	Specifies whether visitor (true) or card holder (false) data titles are being requested.

Response Type

XML_DataTitle[ ]		
Property Name	**Type**	**Description**
DataTitleID	int32	Gives the numeric value assigned to this personal data title in the Symmetry software; e.g. "14".
DataTitle	string	Gives the data title; e.g. "Hair color".
Mandatory	boolean	Returns true if it is mandatory for information to be specified for the data title in the Card Holders or Visitors screen.

FieldType	enFieldType	Determines how a user selects/enters personal data for this personal data title. enFieldType can have one of the following enumerated values: EditList (enter personal data or select from predefined list) ListOnly (select from predefined list only) String (enter personal data only) ExpiryDate (enter date only)
CategoryType	enCategoryType	Determines the type of information entered for this personal data title. enCategoryType can have one of the following enumerated values: Custom (as below) Date General (no required format) Time Email
Mask	string	Determines the required format of the information, depending on the selected CategoryType: **General** – Mask not used. **Date** – Specify dd/mm/yy, yy/mm/dd, dd/mm/yyyy, mm/dd/yyyy or mm/dd/yyyy. **Time** – Specify hh:mm (24-hour clock) or h:mm $ (12-hour clock). In the latter case, the $ indicates that either am or pm would be accepted (in lowercase) when specifying the personal data, e.g. 11:15 am. **Custom** – Use # to indicate an alphanumeric character, and 0 for a numeric character. For example, if you specify #0000, the personal data specified for this title must consist of one alphanumeric character, followed by four numeric characters (e.g. E1500). If you include any other character, the personal data must include the character at the specified position. For example, if you specify 00.00, the personal data must include four numeric characters, with a period (decimal point) after the first two (e.g. 10.99).

G4TGetSystemParameters

Description

Returns the system start and end dates used by the Symmetry software. The Symmetry software accepts only those dates that are within this range. Any date outside this range is automatically set by the Symmetry software to the nearest start or end date.

The method enables dates to be checked before being sent to the Symmetry software.

Screen Permissions Needed

None

Syntax

XML_SystemParameters G4TGetSystemParameters ( )

Input Type

None

Response Type

XML_SystemParameters		
Property Name	**Type**	**Description**
SystemStartDate	dateTime	Gives the start date and time of the system.
SystemEndDate	dateTime	Gives the end date and time of the system.
StandardTimeZoneName	string	Gives the name of the time zone selected on the machine where the XML Open Integration Module is installed.
DaylightSavingTimeZoneName	string	Gives the same string as StandardTimeZoneName if **Automatically adjust clock for daylight saving changes** is selected in Windows. Otherwise, the string will be empty.
CurrentDateAndTime	dateTime	Gives the current date and time at the machine where the XML Open Integration Module is installed.
IsDayLightSavingTime	boolean	Returns true if the time at the machine where the XML Open Integration Module is installed is currently in a daylight-savings period.
UTC	dateTime	Gives the Greenwich Mean Time (GMT).
UTCTimeOffsetMins	double	Gives the offset in minutes between the current time at the XML Open Integration Module machine and GMT.

G4TGetTimeCodes

Description

Returns the time codes belonging to companies in the user's company group (as determined by their login username). Companies to which the user has access are given by G4TGetCompanies (page 57).

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

XML_TimeCode[ ] G4TGetTimeCodes ( )

Input Type

None

Response Type

XML_TimeCode[ ]		
Property Name	**Type**	**Description**
TimeCodeID	int32	Gives the ID of the time code being returned.
TimeCodeDescription	string	Gives the description of the time code being returned.
CompanyID	int32	Gives the ID of the company whose time code is being returned.

G4TGetVersionInformation

Description

Retrieves the versions of the Symmetry client software, Symmetry database and SQL Server software.

Screen Permissions Needed

None

Syntax

XML_Version[ ] G4TGetVersionInformation ( )

Input Type

None

Response Type

XML_Version[ ]		
Property Name	**Type**	**Description**
What	string	For the Symmetry client software, this returns the name of the client machine. For the Symmetry database, this returns "MultiMAX Databases". For SQL Server, this returns "SQL Server".
MinorVersion	string	Not currently used.
MajorVersion	string	Not currently used.
ReleaseVersion	string	Gives the Symmetry client software version number, database version number or the version of SQL Server in use.
LanguageVariant	string	Not currently used.
BuildLevel	string	Gives the build number/level of the Symmetry database (returned only when What is "MultiMAX Databases".
OS	string	Gives the Operating System running on the Symmetry client machine (returned only when What is the name of a client machine).

G4TImportCardHolderData

Description

This method gives you access to the Symmetry database to:

- Add new card holders or visitors.
- Delete card holders or visitors.
- Modify existing card holder or visitor data.
- Make cards inactive.
- Modify or remove card access rights.

Notes:

1. For additional information on the method properties, please refer to the Symmetry software online help. Of particular relevance is the help for the Card Holders Definition screen, Data Import screen and Visitor Definition screen.

2. The ImportDataVisitorInfo property determines whether the import is for a visitor or card holder. See page 85.

3. For performance reasons, if the ImportDataImmediately property is set to true, all card import requests in a single batch are added to the Symmetry import database with the ImportNow field set to 0. A final empty request is automatically injected with ImportNow set to 1, which does not affect the card data in the Symmetry database, but begins the import process for all requests in the batch. Please be aware that empty requests are visible in the Symmetry import database and will be reported by G4TGetImportCardHolderDataStatus (page 68). The empty requests should be periodically removed from the DataImportTable of the Symmetry database, perhaps using a simple SQL job.

Screen Permissions Needed

"Operation/Data/Data Import"

Syntax

XML_ImportDataCardHolderStatusDocument G4TImportCardHolderData (XML_ImportDataCardHolderDocument)

Input Type

XML_ImportDataCardHolderDocument		
Property Name	**Type**	**Description**
ImportDataImmediately	boolean	Specifies whether or not the import should take place immediately. If not, the import will occur at the next automatic import time or when this option is next set to be true (see note 3 above).

CardHolderImportData	XML_ImportDataCardHolder[ ]	Specifies the type of import being requested (e.g. "add card" or "modify card") together with person's details (e.g. name and employee reference) and the card itself (e.g. PIN and expiry date), as described next.

XML_ImportDataCardHolder[ ]

XML_ImportDataCardHolder[ ]		
Property Name	**Type**	**Description**
AreaOccupancyCard	boolean	Specifies whether the card has the indicated status.
CardWatch	boolean	
CommandCard	boolean	
ConditionalCard	boolean	
Executive	boolean	
ExtendedDoorTimes	boolean	
Keycard	boolean	
PatrolCard	boolean	
VisitorEscort	boolean	
ExemptAntiPassback	boolean	
ExemptAutoDelete	boolean	
ExemptPIN	boolean	
ExemptRandomSearchField	boolean	
ImportRequest	enImportRecordRequest	Specifies the import process being requested; e.g. "add card" or "modify card". The type can have one of the enumerated values specified on page 100.
LastName	string	Specifies the person's last name.
FirstName	string	Specifies the person's first name.
InitLet	string	Specifies the person's middle initial.
EmployeeReference	string	Specifies the card holder's employee reference. This is not relevant for visitors.
CompanyID	int32	Specifies the ID of the person's company, as determined by G4TGetCompanies (see page 57).

AssignDefaultBadge	boolean	Specifies that the default badge design is used. The default can be different for card holders and visitors. This setting is not used if a value for BadgeFormatID (page 42) is specified.
FaceImage	byte[ ]	Specifies the face image data as a base64 encoded array of bytes.
SignatureImage	byte[ ]	Specifies the signature image data as a base64 encoded array of bytes.
ImportDataCardInfo	XML_ImportDataCardInfo[ ]	Specifies the card details being imported; e.g. PIN number or active date, as described in the next section.
ImportDataVisitorInfo	XML_ImportDataVisitorInfo[ ]	Provides additional details for a visitor import. See page 87. If you do not want to create a visitor record, do not include any XML_ImportDataVisitorInfo[] properties. A visitor record will be created if any property of XML_ImportDataVisitorInfo[] has a value or is null.
ImportDataPersonalData	XML_ImportDataPersonalData[ ]	Specifies the personal details for the card holder or visitor; e.g. hair color. See page 88.
ImportDataAccessRights	XML_ImportDataAccessRights[ ]	Specifies the access rights for the card holder or visitor; e.g. readers accessible and time codes. See page 88. Specify one of the following: a time code and reader, a time code and a reader group, or an access code.

XML_ImportDataCardInfo[ ]

XML_ImportDataCardInfo[ ]		
Property Name	**Type**	**Description**
Lost	boolean	Specifies whether the card is lost.
PIN	int32	Specifies the PIN number of the card (used only for a 4-digit PIN; see ExtendedPIN).
ActiveDate	dateTime	Specifies the date and time that the card became active.
ExpiryData	dateTime	Specifies the date and time that the card expires.
InactiveComment	string	Used if ImportRequest specifies MakeCardInactive. Specifies the reason why the card has been made inactive. This information is added to the notes of the card definition.

Encryption	int32	This is used to protect against the import of unauthorized data. If a password is specified in the Data Import screen of the Symmetry software, the password must be encrypted and specified in this property. The encrypted password is obtained by performing an exclusive OR between the password and a concatenation of certain data. For full details of the encryption method, please see the *Data Connect Manual*.
BadgeFormatID	int32	ID of the person's badge design, as determined from G4TGetBadgeFormats (page 42).
CardFormat	enCardFormat	This is the card type. EnCardFormat can have one of the following enumerated values: Standard SeiwgV1 (Obsolete) Legacy SeiwgGSC2_1 (Obsolete) PIV_UUID PIV_FASC_N SRSeries_15Digit SRSeries_10And12Digit CMV_U CMV_F Custom150 to Custom254 Custom150 to Custom254 are used for user-defined card formats in the "Install/System/Default Settings/Card Format" screen and correspond to the **Format ID** in that screen. For example, if the **Format ID** is 167, EnCardFormat should be Custom167.
CardNumber	int32	Specifies the card number.
CustomerCode	int32	Non-Homeland only. Specifies the card's customer code.
CardIssueLevel	unsignedByte	Non-Homeland only. Specifies the card issue number.
CredentialIssueLevel	unsignedByte	Homeland only. Specifies the card issue number.
CredentialNumber	int32	Homeland only; non-PIV cards. Specifies the person's credential number.
SystemCode	int16	Homeland only. Specifies the person's system code. **Note:** If a system code of zero is required, specify a value of -1 during card import.
AgencyCode	int16	Homeland only. Specifies the person's agency code.
PersonIdentifier	int64	Homeland; PIV cards only. A number, such as passport number, used to identify the card owner.
OrganisationCategory	int16	Homeland; PIV cards only. Single-digit code specifying the category of the source as follows: 1 = Federal Government; 2 = State Government; 3 = Commercial Enterprise; 4 = Foreign Government.

OrganisationIdentifier	int16	Homeland; PIV cards only. Four-digit code specifying origin of the card owner as follows: 1st digit = FIPS 95-2 Federal Agency/Sub-Agency Code; 2nd digit = State Code; 3rd digit = Company Code; 4th digit = Country Code.
OrganisationAssociation	int16	Homeland; PIV cards only. Single-digit code specifying the association between the organization and the person as follows: 0 = Employee; 1 = Civil; 2 = Executive Staff; 3 = Uniformed Service; 4 = Contractor.
CredentialSeriesCode	int16	Homeland only; PIV cards only. Single-digit number intended to reflect the current system configuration. Incremented when a major system configuration change takes place.
IDSCode	string	The IDS code is the card holder's personal code, which can be to gain access to intrusion options at an M2150 intrusion reader, such as to arm or disarm an area.
CardIdentifier	string	Homeland only. The UUID number of the card (for CMV-U cards only). For example, {78563412-bc9a-f0de-1234-56789abcdef0}.
ExtendedPIN	string	Specifies the PIN number of the card (used only for a PIN of more than 4 digits).

XML_ImportDataVisitorInfo[ ]

XML_ImportDataVisitorInfo		
Property Name	**Type**	**Description**
Representing	string	Specifies the name of the visitor's company name.
Vehicle	string	Specifies the details of the visitor's vehicle.
ArrivalDateTime	dateTime	Specifies the date and time when the visitor arrived.
DepartureDateTime	dateTime	Specifies the date and time when the visitor is to depart.
Visiting	string	Specifies the person who the visitor is seeing.
ContactNumber	string	Specifies the contact telephone number of the visitor.
Message	string	Specifies any message in connection with the visitor.
BusinessCard	byte[ ]	Gives a scanned image of the visitor's business card, as a base64 encoded array of bytes.

XML_ImportDataPersonalData[ ]

XML_DataPersonalData[ ]		
Property Name	**Type**	**Description**
PersonalDataID	int32	Specifies the ID of the personal data title; e.g. Hair color may have an ID of 14. (The personal data fields appear in the Personal tab of the Card Holder or Visitors screen.) To identify the personal data IDs for each data title available, use G4TGetPersonalDataTitles (page 77). **Note:** The first Personal Data entry appearing in the Personal tab has an ID value of 0.
Data	string	Specifies the value of this personal data; e.g. "Brown".

XML_ImportDataAccessRights[ ]

XML_ImportDataAccessRights[ ]		
Property Name	**Type**	**Description**
TimeCodeID	int32	Specifies the ID of the time code. See G4TGetTimeCodes (page 81) for retrieving time code IDs.
AccessGroupID	int32	Specifies the ID of the access code that defines the person's access rights. See G4TGetAccessGroups (page 34) for retrieving access codes.
ReaderID	int32	Specifies the ID of the individual reader to which the person has access rights. G4TGetDevices (page 61) for retrieving reader IDs.
ReaderGroupID	int32	Specifies the ID of the reader group to which the person has access rights. G4TGetDevices (page 61) for retrieving reader group IDs.
AreaID	int32	Specifies the ID of an M2150 intrusion area to which the person has access rights. G4TGetDevices (page 61) for retrieving area IDs.

Response Type

XML_ImportDataCardHolderStatusDocument		
Property Name	**Type**	**Description**
BatchReference	string	Gives a unique reference number for this import operation.
CardHolderImportDataStatus	XML_ImportDataCardHolderStatus[ ]	Returns details of the card holder or visitor and the status resulting from the import operation. See page 102.

G4TLogin

Description

For every method received, the WS Security (see page 3) checks the username and password against those stored in the Symmetry database. If there is no match, WS Security returns a fault exception. If there is a match, the WS Security allows the method call to pass to the web service where it is processed and passed to the Symmetry software.

G4TLogin is a dedicated login method to make a test call on the username and password. If there is a match, the method returns true. If not, a fault exception is returned. See page 18 for an example.

Screen Permissions Needed

None

Syntax

G4TLoginResult G4TLogin ( )

Input Type

None

Response Type

G4TLoginResult		
Property Name	**Type**	**Description**
G4TLoginResult	boolean	Returns true for a successful match. (If no match, returns a fault exception.)

G4TRename

Description

Enables the user to rename a device in the database.

The user can access devices belonging to any company.

Screen Permissions Needed

Permissions are required to the definition screen appropriate to the device being renamed. For example, when renaming a monitor point, permissions to the "Install/Access Control/Monitor Point" screen are required.

Doors and readers are the same objects.

Camera devices require permissions to the appropriate screen in "Install/Digital Video".

Syntax

G4TRename (XML_RenameRequest)

Input Type

XML_RenameRequest		
Property Name	**Type**	**Description**
DeviceName	int32	Specifies the new name to use for the device.
DeviceType	enDeviceTypes	Specifies the type of device to rename. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Specifies the ID of the device to rename.

Response Type

None

G4TSendDeviceCommand

Description

Sends a command to a Symmetry device.

Screen Permissions Needed

"Home/Monitoring/Command Center"

Syntax

G4TSendDeviceCommand (XML_SendDeviceCommandRequest[ ])

Input Type

XML_SendDeviceCommandRequest[ ]		
Property Name	**Type**	**Description**
DeviceType	enDeviceTypes	Specifies the type of device; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified on page 99.
DeviceID	int32	Optional – Specifies the ID of the device. Value must be greater than zero.
CommandType	enCommandType	Specifies the type of command; e.g. manual command. enCommandType can have one of the enumerated values specified on page 98.
CommandCode	int32	Specifies the identification number for the command, as returned by G4TGetCommands (see page 49).
TailStructureNumber	int32	Provides additional information about the command that is essential for the Symmetry software to action the command correctly. This must be the same as the TailStructureNumber obtained for the command using by G4TGetCommands (see page 49).
TailInfoOption	string	Provides additional information about the command that is essential for the Symmetry software to action the command correctly. This must be the same as the TailInfoOption obtained for the command using by G4TGetCommands (see page 49).
Value1	int32	Specifies the number of seconds or minutes, as defined by Value2 below; e.g. 10. Used only for the Pulse command for auxiliary outputs.
Value2	string	Specifies the unit of time for Value1 above. Use S for seconds, or M for Minutes. Used only for the Pulse command for auxiliary outputs.

Response Type

None

G4TSendPCDoorControlAccessCommand

Description

Sends a grant access or deny access command to a reader that is in PC Door Control mode.

Transactions requiring a grant access or deny access response have an "access requested" ResponseMnemonic, which can be identified using the G4TGetActivity method (page 35). G4TGetActivity also provides the property values necessary for G4TSendPCDoorControlAccessCommand to identify the reader to send the grant/deny access command to.

Note: XML-issued commands can be reviewed from the Symmetry client interface by reporting user activity.

Screen Permissions Needed

None

Syntax

G4TSendPCDoorControlAccessCommand (XML_PCDoorControlRequest)

Input Type

XML_PCDoorControlRequest		
Property Name	**Type**	**Description**
DoorControlAction	enPCDoorControlAction	Specifies whether to grant or deny access. enPCDoorControlAction can have one of the following enumerated values: GrantAccess DenyAccess
DoorControlUnitID	int32	Specifies the ID of the door-control unit that the reader is connected to.
DeviceNumber	int32	Specifies the DeviceNumber of the reader. Each reader has a unique DeviceNumber for a particular DoorControlUnitID, which is 0 for odd-numbered readers and 1 for even-numbered readers.
CardNumber	int32	Specifies the card number that is being granted or denied access.
CardHolderID	int32	Specifies the ID of the card holder who is being granted or denied access.
NodeID	int32	Specifies the ID of the access-control node to send the message to. DeviceID from G4TGetActivity provides the node ID for an "access requested" ResponseMnemonic.

Response Type

None

G4TSetAlarmActivityCategory

Description

Allows alarm categories, such as "Alarm Panel", "Fire System" and "Building Control System" to be added or modified. When an alarm is added using G4TAddAlarms (page 21) or G4TAddAlarmsWithCommandLine (page 22), the alarm category is specified to indicate the source of the alarm.

If AllowReporting is set to TRUE, the alarm category will be displayed as a checkbox in the System Activity area of the "Reports/History/Activity" screen to enable reports to be filtered for the device that they relate to. The text next to the checkbox is as specified by AlarmActivityDescription.

Screen Permissions Needed

None

Syntax

G4TSetAlarmActivityCategory (XML_AlarmActivityCategory[ ])

Input Type

XML_AlarmActivityCategory[ ]		
Property Name	**Type**	**Description**
ActivityID	enAlarmActivityCategory	Specifies the ID of the external system that is providing the alarm information. enAlarmActivityCategory can have one of the enumerated values specified on page 98.
AlarmActivityDescription	string	The description of the alarm category (e.g. "Intruder Panel")
AllowReporting	boolean	When TRUE, the alarm activity description is added as a checkbox in the "Reports/History/Activity" screen in the Symmetry software.
ReportbyDefault	boolean	Specifies the default state of the checkbox in the "Reports/History/Activity" screen. TRUE = selected by default.

Response Type

None

G4TSetCardHoldersAntipassbacktoNeutral

Description

For zonal antipassback, this method places the specified card into the antipassback neutral zone. For timed antipassback, it causes the delay period for reuse of the card to be zeroed.

Screen Permissions Needed

None

Syntax

G4TSetCardHoldersAntipassbacktoNeutral (XML_CardholderAntipassbackRequest)

Input Type

See G4TSetCardHoldersAntipassbackZone for details of the input type (XML_CardholderAntipassbackRequest).

Note: ReaderID is not used in the input type.

Response Type

None

G4TSetCardHoldersAntipassbackZone

Description

Creates a transaction at the specified reader and sets a card's antipassback zone to the reader's "Into Zone"; that is, the zone that the reader gives access to.

This method is appropriate for zonal antipassback only.

Screen Permissions Needed

None

Syntax

G4TSetCardHoldersAntipassbackZone (XML_CardHolderAntipassbackRequest)

Input Type

XML_CardHolderAntipassbackRequest		
Property Name	**Type**	**Description**
CardID	int32	Specifies the Symmetry database CardID value of the card that is to have its antipassback zone set. (This is not the card number.) See XML_CardInfo[] CardID property to obtain the Symmetry database CardInfoID of the card.
ReaderID	int32	Specifies the ID of the reader (see above).

Response Type

None

Base Types

The following are standard types occurring in two or more methods:

byte[ ]

An array of bytes used to define binary data, such as an image. The data is encoded by the XML Open Integration Module using base64 format.

boolean

True or false.

dateTime

Standard date/time value.

double

Double-precision floating point value.

int16

A 16-bit integer, greater than or equal to zero.

int32

A 32-bit integer, greater than or equal to zero.

int64

A 64-bit integer, greater than or equal to zero.

string

A text string variable.

unsignedByte

An unsigned single byte value.

Types used in Several Methods

The following are types occurring in two or more methods.

enAlarmActivityCategory

enAlarmActivityCategory	
Type	**Description**
enAlarmActivityCategory	Identifies the source of the alarm, as defined by G4TSetAlarmActivityCategory (page 94). enAlarmActivityCategory can have one of a hundred possible enumerated values in the form System<n>, where <n> is from 1 to 100 inclusive (e.g. System15). When enAlarmActivityCategory is a return type, the value "NotApplicable" indicates that the alarm source has not been specified. This is used for alarms produced by the access-control system.

enCommandType

enCommandType	
Type	**Description**
enCommandType	Specifies the type of command. Used as an input property by G4TGetCommands (page 49) and G4TSendDeviceCommand (page 91), and returned by G4TGetCommandTypes (page 55). By default, enCommandType has one of the following enumerated values when using C#: ALL_COMMAND MANUAL_COMMAND CONDITIONAL_COMMAND (for a trigger command) SCHEDULED_COMMAND If you are not using C# or the default values, you can use G4TGetCommandTypes (page 55) to determine the enumerated values for enCommandType.

XML_DeviceType[ ]

XML_DeviceType[ ]		
Property Name	**Type**	**Description**
TypeID	enDeviceTypes	Gives the type of device returned; e.g. reader or auxiliary output. enDeviceTypes can have one of the enumerated values specified in the following table.
TypeDescription	string	Gives the description of the device type being returned; e.g. "Reader device".

enDeviceTypes

enDeviceTypes	
Type	**Description**
enDeviceTypes	Specifies the type of device. Used as an input and response property by many methods. By default, enDeviceTypes has one of the following enumerated values when using C#: UNKNOWN_DEVICE DOOR_DEVICE MONITORPOINT_DEVICE NODE_DEVICE AUXILIARY_OUTPUT READER_DEVICE FLOOR_GROUP_DEVICE FLOOR_DEVICE READER_GROUP_DEVICE INTRUSION_AREA CAMERA_DV_DEVICE ACCESS_CODE_DEVICE (an access code) SCHEDULED_COMMAND_DEVICE (a scheduled command) TRIGGER_COMMAND_DEVICE (a trigger command) If you are not using C# or the default values, you can use G4TGetAdminTypes (page 39) to determine the enumerated values for enDeviceTypes. **Note:** Additional device types may be added in later versions of the XML Open Integration Module. Therefore, do not rely on a fixed device count being returned by this method.

enImportRecordRequest

enImportRecordRequest	
Type	**Description**
enImportRecordRequest	Specifies the import operation requested by G4TImportCardHolderData (page 83). Also returned by G4TGetImportCardHolderDataStatus (page 68) and G4TGetImportRecordRequests (page 69). By default, enImportRecordRequest can have one of the following enumerated values when using C#. The RemoveAllThen... values indicate removal of all normal and advanced access rights before applying the specified new access rights. AddCard DeleteCard ImportMultipleCards MakeCardActive MakeCardInactive ModifyCard ModifyCardAccessRights ModifyAdvanced1AccessRights ModifyAdvanced2AccessRights ModifyAdvanced3AccessRights ModifyAdvanced4AccessRights ModifyAdvanced5AccessRights ModifyAdvanced6AccessRights ModifyAdvanced7AccessRights ModifyAdvanced8AccessRights ModifyAdvanced9AccessRights ModifyAdvanced10AccessRights ModifyAdvanced11AccessRights ModifyAdvanced12AccessRights ModifyAdvanced13AccessRights ModifyAdvanced14AccessRights ModifyAdvanced15AccessRights ModifyAdvanced16AccessRights RemoveAllThenAddCardNormalAccessRights RemoveAllThenAddCardAdvanced1AccessRights RemoveAllThenAddCardAdvanced2AccessRights RemoveAllThenAddCardAdvanced3AccessRights RemoveAllThenAddCardAdvanced4AccessRights RemoveAllThenAddCardAdvanced5AccessRights RemoveAllThenAddCardAdvanced6AccessRights RemoveAllThenAddCardAdvanced7AccessRights RemoveAllThenAddCardAdvanced8AccessRights RemoveAllThenAddCardAdvanced9AccessRights RemoveAllThenAddCardAdvanced10AccessRights RemoveAllThenAddCardAdvanced11AccessRights RemoveAllThenAddCardAdvanced12AccessRights RemoveAllThenAddCardAdvanced13AccessRights RemoveAllThenAddCardAdvanced14AccessRights RemoveAllThenAddCardAdvanced15AccessRights RemoveAllThenAddCardAdvanced16AccessRights RemoveCardAccessRights RemoveCardSingleAccessRight If you are not using C# or the default values, you can use G4TGetImportRecordRequests (page 69) to determine the enumerated values for enImportRecordRequest.

enImportRecordResult

enImportRecordResult	
Type	**Description**
enImportRecordResult	Gives the result of an import operation requested by G4TImportCardHolderData (page 83). Also returned by G4TGetImportCardHolderDataStatus (page 68). enImportRecordResult can have one of the following enumerated values: AccessGroupAlreadyExistsForCardID AccessGroupConflict ActiveorInactiveDateOutOfRange AreaAlreadyExistsForCardID AreaConflict BiometricImportError CardAlreadyExists CardInUseByAConditionalCommand CardIssueLevelOutOfRange CardIssueLevelsNotSet CardNumberOutOfRange CredentialNumberOutOfRange EmployeeReferenceNotUnique FaceFileNotFound GSCFieldOutOfRange IDSCodeNotUnique ImportEncryptionError ImportFailedToIdentifyUniqueCardHolder ImportFormatError ImportGeneralError ImportRecord ImportRecordNotFound ImportSuccessful InvalidAccessGroupID InvalidActiveDataRange InvalidAreaID InvalidBadgeFormatID InvalidCardUsageRemaining InvalidCompanyID InvalidCustomerCode InvalidIDSCode InvalidPINCode InvalidReaderGroupID InvalidReaderID InvalidSystemCode InvalidTimeCodeID IPMODataInvalid KeycardHolderAndExecutiveCardOptionsMutuallyExclusive MandatoryFieldsMissing MandatoryPersonalDataError MissingCardNumber MoreThanOneCompany MoreThanOneCustomerCode MultipleCardsPerCardholderNotEnabled NoVisitorsEscortAssigned ReaderGroupTimeCodeAlreadyExistsForCardID ReaderGroupTimeCodeConflict ReaderTimeCodeAlreadyExistsForCardID ReaderTimeCodeConflict SignatureFileNotFound UnknownRecordRequest **Note:** ImportRecord means that the import data is awaiting transfer to the main Symmetry database.

who

who		
Property Name	**Type**	**Description**
FirstName	string	Gives the person's first name.
LastName	string	Gives the person's last name.

XML_ImportDataCardHolderStatus[ ]

XML_ImportDataCardHolderStatus[ ]		
Property Name	**Type**	**Description**
LastName	string	Specifies the person's last name.
FirstName	string	Specifies the person's first name.
InitLet	string	Specifies the person's middle initial.
EmployeeReference	string	Specifies the person's employee reference (not relevant to visitors).
CompanyID	int32	Specifies the ID of the company.
ImportRequest	enImportRecordRequest	Gives the operation to which the status response relates; e.g. "Add card". See page 100.
ImportResult	enImportRecordResult	Gives the result of the operation to which the status response relates; e.g. "Import Successful". See page 101.
ImportResultDescription	string	A Symmetry message, summarizing the outcome of the import operation; e.g. "Cannot add this card as it already exists in the database".

Pagination

When using G4TGetActivity, G4TGetCardHolderInformation and G4TGetOutstandingAlarms, all records matching the specified criteria are returned, which may cause a problems for performance and memory management. The G4TGetActivityDocument, G4TGetCardHolderInformationDocument and G4TGetOutstandingAlarmsDocument methods return the same data, but use the pagination properties, which allow the number of records returned at one time to be limited. The Bookmark field, which is returned together with the data, may be used in the next call to the same method to retrieve the next page of results.

Pagination		
Property Name	**Type**	**Description**
Bookmark	string	This is an opaque reference to a position within a collection of records. It is returned with the results of a call to a method that supports pagination, and can be supplied to the next call to the same method to retrieve the next page of results. No assumptions may be made regarding the content of this field.
PageSize	int	Specifies the number of records to return.
UsePagination	bool	Specifies whether or not to use pagination. If false, all records are returned.

Chapter 4: Troubleshooting

"The Operation has timed-out"

This exception can occur when retrieving a large number of records. If you notice this message when using G4TGetOutstandingAlarms, the situation can normally be rectified by regularly clearing outstanding alarms or by using alarm filtering to ensure that only the alarms required are retrieved. This example highlights the need to anticipate the conditions/workloads of the environment in which the integration will be placed.

If manual alarm clearance is not being carried out onsite, it is recommended that alarms are acknowledged and cleared automatically using G4TClearOutstandingAlarm. The method could be called at scheduled intervals to acknowledge and clear alarms previously retrieved from G4TGetOutstandingAlarms. The application may, for example, be set up to acknowledge and clear alarms that occurred before a specified date. All alarms retrieved using G4TGetOutstandingAlarms could be written to a log file for later examination.

This error can also be created by overloading the XML web service with data requests (for example from multiple clients), or by making repetitive requests within short time intervals. It is important that this situation is avoided. The design and testing of third-party software is the responsibility of the third-party development team and should always be performed prior to usage on site to ensure proper performance.

Verbose Error Reporting

By default, verbose reporting is not provided when running an application that accesses the web service from a web client. Not using verbose reporting may provide insufficient information to determine the cause of an application error.

Verbose reporting can be obtained either by running the application on the web server, or by changing the customErrors setting in the Web.config file to customErrors="Off". The setting is normally "On" or "RemoteOnly".

Changing the setting does not require IIS to be restarted, but the client application will need to be restarted to create a new connection to the web service.

Once the application is released, customErrors should be returned to its normal setting.

Cannot Connect to Web Service

Attempt to connect to the web service by entering the web service URL into Internet Explorer on the development machine. The format of the URL is:

http://<Server Name>/smsXMLWebService/smsXMLWebService.svc (or https if encryption used)

A list of methods should be displayed if the connection is successful.

"The option has not been configured to allow for sms Web Service Interaction"

This exception indicates that the XML Open Integration Module has not been found on the Symmetry server. Install the XML Open Integration Module if it is not installed. Check that the licenses activation code is entered in the "Maintenance/Licensing/System Licenses" screen.

"The security token could not be authenticated or authorised"

This is displayed if there is an authentication problem. Check that the login user name and password are correct, the web service has sufficient privileges to use SQL Server and that SQL Server is running.

Verbose reporting (see page 104) may provide additional information to determine the source of the problem. You may also find that the SQL Profiler application helps to locate the problem, as described next.

Using the SQL Profiler

The SQL Profiler tool, which is provided with Microsoft SQL Server, can be used to confirm that messages are being passed between the XML service and the Symmetry database. The Profiler can be useful to debug applications that are using the web service. To view the messages:

1. Start the SQL Profiler.
2. Select **File/New Trace**.
3. Select the Symmetry server and enter the authentication details.
4. Click **Run** in the Trace Properties dialog (leaving the settings unchanged).
5. Re-generate the XML method call and view the displayed messages. You can click the **Stop** button in the toolbar to stop the trace, if required.

Debugging a Method

If an XML method is not operating as expected, try running the supplied sample code (containing the relevant method) to confirm that the basic code is working.

You may also find the SQL Profiler useful, as described above.

Requesting Large Amounts of Data

When requesting large amounts of data from the XML web service, it may be necessary to change the 'maxReceivedMessageSize' in the client application as follows:

1. Open up the client application's app.config file.
2. Locate the following line:

   <binding name="WSHttpBinding_ISMSXMLWebService"
3. Change to:

   <binding name="WSHttpBinding_ISMSXMLWebService" maxReceivedMessageSize="x">

   Where x is the size in bytes (default is 65536).

WCF and Self Signed SSL Certificates

By default, Windows Communication Foundation (WCF) blocks the use of self-signed certificates, which produces a Transport Layer Security (TLS) error when connecting a client application to the web service. To prevent this error (only recommended in a test / developer environment), add the following code to your application:

```
private Label lblCompanyID;
private ComboBox cboCompany;
   private System.ComponentModel.Container components = null;
   public ClientTestApp ()
   {
      InitializeComponent();
      System.Net.ServicePointManager.ServerCertificateValidationCallback =
      ((sender, certificate, chain, sslPolicyErrors) => true);
   }
```

MSXMLwebService/G4TAddAlarms is incorrect...

The following message indicates a timeout issue:

> "MSXMLWebService/G4TAddAlarms is incorrect or because the message contains an invalid or expired security context token or because there is a mismatch between bindings. The security context token would be invalid if the service aborted the channel due to inactivity. To prevent the service from aborting idle sessions prematurely increase the Receive timeout on the service endpoint's binding."

Try adding the following to the client application's app.config file:

```
<system.net>
  <connectionManagement>
    <add maxconnection = "200" address ="*" />
  </connectionManagement>
</system.net>
```