



DataConduit User Guide



Lenel OnGuard® 7.0 DataConduit User Guide
This guide is item number DOC-920, revision 4.038, June 2014

Copyright © 1995-2014 Lenel Systems International, Inc. Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Lenel Systems International, Inc.

Non-English versions of Lenel documents are offered as a service to our global audiences. We have attempted to provide an accurate translation of the text, but the official text is the English text, and any differences in the translation are not binding and have no legal effect.

The software described in this document is furnished under a license agreement and may only be used in accordance with the terms of that agreement. Lenel and OnGuard are registered trademarks of Lenel Systems International, Inc.

Microsoft, Windows, and Windows Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Integral and FlashPoint are trademarks of Integral Technologies, Inc. Crystal Reports for Windows is a trademark of Crystal Computer Services, Inc. Oracle is a registered trademark of Oracle Corporation. Other product names mentioned in this User Guide may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

Portions of this product were created using LEADTOOLS © 1991-2014 LEAD Technologies, Inc. ALL RIGHTS RESERVED.

OnGuard includes ImageStream® Graphic Filters. Copyright © 1991-2014 Inso Corporation. All rights reserved. ImageStream Graphic Filters and ImageStream are registered trademarks of Inso Corporation.

Table of Contents

<i>CHAPTER 1</i>	<i>Introduction</i>	<i>9</i>
	Documentation Contents	10
	Documentation Prerequisites	10
	Definitions, Acronyms, Abbreviations	10
	References and Applicable Documents	11
<i>CHAPTER 2</i>	<i>Getting Started</i>	<i>13</i>
	License for DataConduIT	13
	Authentication	13
	Authorization	14
	Receiving Events	14
	Using DataConduIT from a Remote Computer	14
	Viewing DataConduIT Classes with the Microsoft WMI SDK	15
	Overview of DataConduIT Functions	16
<i>CHAPTER 3</i>	<i>Using DataConduIT for Data Access</i>	<i>19</i>
	Connecting to DataConduIT	19
	Searching for Objects	19
	Adding Objects	21
	Modifying Objects	22
	Deleting Objects	23
	Features and Limitations	23
	<i>Cardholders and Visitors</i>	23
	<i>Badges</i>	23
	<i>Directory Accounts</i>	24

	<i>Visits</i>	24
	<i>Multimedia Objects</i>	24
	<i>User-Defined List Values</i>	24
CHAPTER 4	<i>Using DataConduIT to Receive Events</i>	25
	Registering to Receive Hardware Events	26
	Receiving Hardware Events	27
	Registering to Receive Software Events	27
	Receiving Software Events	28
	Using Permanent Event Consumers with DataConduIT	28
CHAPTER 5	<i>Using DataConduIT to Send Alarms to OnGuard</i>	31
CHAPTER 6	<i>Working with MobileVerify</i>	33
CHAPTER 7	<i>Troubleshooting and Advanced Options</i>	35
	Receiving Error Information from DataConduIT	35
	Before Calling Technical Support	36
	Error Logging	36
	Changing the Database Connection Pool Time	37
	Tuning Parameters	37
	Stopping and Restarting the DataConduIT Service	38
CHAPTER 8	<i>Getting Started with DataConduIT Message Queues</i>	39
	Overview of DataConduIT Message Queue Functions	40
	Supported Queue Types	40
	<i>Outgoing Queue Overview</i>	41
	Schema Overview	41
	How DataConduIT Message Queue Handles Database Layout Changes	42
	Updating the Database with Queue Changes	42
	Error Logging	42
	Installing DataConduIT Message Queue	43
	License for DataConduIT Message Queue	43
	Setting Permissions to Use DataConduIT	43
	<i>Configure the System Options</i>	43
	<i>Configure the User Permissions</i>	44
	Configuring DataConduIT Message Queue	44
	<i>Configure the DataConduIT Message Queue</i>	44
	<i>Change the Account the DataConduIT Message Service is Run With</i>	45

<i>CHAPTER 9</i>	<i>DataConduIT Message Queues Folder</i>	<i>47</i>
	DataConduIT Message Queues Form (General Sub-tab)	48
	DataConduIT Message Queues Form (Settings Sub-tab)	49
	DataConduIT Message Queues Form (Advanced Sub-tab)	50
	DataConduIT Message Queues Form Procedures	51
	<i>Add DataConduIT Message Queue</i>	51
	<i>Modify a DataConduIT Message Queue</i>	52
	<i>Delete a DataConduIT Message Queue</i>	52
<i>CHAPTER 10</i>	<i>DataConduIT Sources Folder</i>	<i>53</i>
	DataConduIT Sources Folder	53
	DataConduIT Source Downstream Devices	54
	Licenses Required	54
	User Permissions Required	55
	<i>DataConduIT Service Permission</i>	55
	<i>Add, Modify, and Delete DataConduIT Sources, Devices, and Sub-Devices</i>	55
	<i>Trace DataConduIT Sources, Devices, and Sub-Devices</i>	55
	DataConduIT Sources Form	55
	DataConduIT Sources Form Procedures	56
	<i>Add a DataConduIT Source</i>	56
	<i>Modify a DataConduIT Source</i>	57
	<i>Delete a DataConduIT Source</i>	57
	DataConduIT Devices Form	58
	DataConduIT Devices Form Procedures	59
	<i>Add a DataConduIT Device</i>	59
	<i>Modify a DataConduIT Device</i>	59
	<i>Delete a DataConduIT Device</i>	59
	DataConduIT Sub-Devices Form	60
	DataConduIT Sub-Devices Form Procedures	61
	<i>Add a DataConduIT Sub-Device</i>	61
	<i>Modify a DataConduIT Sub-Device</i>	61
	<i>Delete a DataConduIT Sub-Device</i>	61
<i>CHAPTER 11</i>	<i>OPC Connections</i>	<i>63</i>
	OPC Client Functions	63
	OnGuard OPC Client Scenario	63
<i>CHAPTER 12</i>	<i>Using SNMP with OnGuard</i>	<i>65</i>
	OnGuard as an SNMP Manager	67
	OnGuard as an SNMP Agent	67
	SNMP Manager Copyright Information	67

<i>CHAPTER 13</i>	<i>Reference</i>	<i>71</i>
	Data Classes	71
	<i>Lnl_AccessGroup</i>	71
	<i>Lnl_AccessLevel</i>	72
	<i>Lnl_AccessLevelAssignment</i>	72
	<i>Lnl_AccessLevelReaderAssignment</i>	73
	<i>Lnl_Account</i>	73
	<i>Lnl_AlarmDefinition</i>	74
	<i>Lnl_Area</i>	74
	<i>Lnl_AuthenticationMode</i>	75
	<i>Lnl_Badge</i>	76
	<i>Lnl_BadgeFIPS201</i>	77
	<i>Lnl_BadgeLastLocation</i>	78
	<i>Lnl_BadgeProperties</i>	79
	<i>Lnl_BadgeType</i>	80
	<i>Lnl_Camera</i>	80
	<i>Lnl_CameraGroup</i>	81
	<i>Lnl_CameraGroupCameraLink</i>	81
	<i>Lnl_Cardholder</i>	82
	<i>Lnl_DataConduITManager</i>	82
	<i>Lnl_Directory</i>	83
	<i>Lnl_Element</i>	83
	<i>Lnl_EventAlarmDefinitionLink</i>	84
	<i>Lnl_EventParameter</i>	84
	<i>Lnl_EventSubtypeDefinition</i>	84
	<i>Lnl_EventSubtypeParameterLink</i>	85
	<i>Lnl_EventType</i>	85
	<i>Lnl_Holiday</i>	86
	<i>Lnl_HolidayType</i>	86
	<i>Lnl_HolidayTypeLink</i>	87
	<i>Lnl_IncomingEvent</i>	87
	<i>Lnl_LoggedEvent</i>	90
	<i>Lnl_LogicalSystemAccount</i>	91
	<i>Lnl_MobileVerify</i>	92
	<i>Lnl_MonitoringZone</i>	93
	<i>Lnl_MonitoringZoneCameraLink</i>	93
	<i>Lnl_MultimediaObject</i>	94
	<i>Lnl_Panel</i>	94
	<i>Lnl_Person</i>	95
	<i>Lnl_Reader</i>	95
	<i>Lnl_Segment</i>	96
	<i>Lnl_SegmentGroup</i>	96
	<i>Lnl_SegmentUnit</i>	96
	<i>Lnl_Timezone</i>	97
	<i>Lnl_TimezoneInterval</i>	97
	<i>Lnl_User</i>	98
	<i>Lnl_UserAccount</i>	99
	<i>Lnl_UserPermissionGroup</i>	99
	<i>Lnl_UserFieldPermissionGroup</i>	101
	<i>Lnl_UserPermissionDeviceGroupLink</i>	101
	<i>Lnl_UserReportPermissionGroup</i>	101
	<i>Lnl_UserSecondarySegment</i>	102
	<i>Lnl_Visit</i>	102

<i>Lnl_VisitEmailRecipient</i>	103
<i>Lnl_Visitor</i>	104
User-Defined Value Lists (<i>LNL_BadgeStatus</i> , <i>Lnl_Title</i> , ...)	104
Association Classes	105
<i>Lnl_AccessLevelGroupAssignment</i>	105
<i>Lnl_BadgeOwner</i>	105
<i>Lnl_CardholderAccount</i>	106
<i>Lnl_CardholderBadge</i>	106
<i>Lnl_CardholderMultimediaObject</i>	106
<i>Lnl_DirectoryAccount</i>	106
<i>Lnl_MultimediaObjectOwner</i>	107
<i>Lnl_PersonAccount</i>	107
<i>Lnl_ReaderEntersArea</i>	107
<i>Lnl_ReaderExitsArea</i>	108
<i>Lnl_SegmentGroupMember</i>	108
<i>Lnl_VisitorAccount</i>	108
<i>Lnl_VisitorMultimediaObject</i>	109
Event Classes	109
<i>Lnl_AccessEvent</i>	109
<i>Lnl_Alarm</i>	110
<i>Lnl_Event</i>	111
<i>Lnl_FireEvent</i>	111
<i>Lnl_FunctionExecEvent</i>	111
<i>Lnl_IntercomEvent</i>	112
<i>Lnl_OtherSecurityEvent</i>	112
<i>Lnl_SecurityEvent</i>	112
<i>Lnl_StatusChangeEvent</i>	113
<i>Lnl_TransmitterEvent</i>	113
<i>Lnl_VideoEvent</i>	114
<i>Lnl_VisitEvent</i>	114
Command and Control: Classes and Methods	115
<i>Lnl_AlarmInput</i>	115
<i>Lnl_AlarmOutput</i>	115
<i>Lnl_AlarmPanel</i>	115
<i>Lnl_Input</i>	116
<i>Lnl_IntrusionArea</i>	116
<i>Lnl_IntrusionDoor</i>	117
<i>Lnl_IntrusionOutput</i>	118
<i>Lnl_IntrusionZone</i>	118
<i>Lnl_IntrusionZoneOutput</i>	118
<i>Lnl_OffBoardRelay</i>	118
<i>Lnl_OnBoardRelay</i>	119
<i>Lnl_Output</i>	119
<i>Lnl_Panel</i>	120
<i>Lnl_Reader</i>	121
<i>Lnl_ReaderOutput</i>	123
<i>Lnl_ReaderOutput1</i>	123
<i>Lnl_ReaderOutput2</i>	123
<i>Lnl_ReaderInput</i>	123
<i>Lnl_ReaderInput1</i>	123
<i>Lnl_ReaderInput2</i>	123

Appendices	125
<i>APPENDIX A Property Qualifiers Used In DataConduIT</i>	<i>127</i>
<i>APPENDIX B Event Generator</i>	<i>129</i>
Event Generator Main Window	129
Edit Event (Simple) Window	130
Edit Event (Advanced) Window	132
Event Generator Menus	136
File	136
Edit	136
Send Event	136
Generate Events	137
Required Event Generator Files	137
Setting Up the Event Generator	137
Registering the LnIEventGeneratoru.dll	138
Adding an Event to the Event Generator	140
Adding an Event Using the Simple User Interface	140
Adding an Event Using the Advanced User Interface	140
Generating Events	140
Generating a Single Event	140
Generating Multiple Events	140
Saving an Event List	141
Loading an Event List	141
Closing the Event Generator	141
<i>APPENDIX C Common DataConduIT Problems</i>	<i>143</i>
<i>APPENDIX D Technical Support Pre-Call Checklist</i>	<i>145</i>
<i>APPENDIX E Visual Basic Demo</i>	<i>147</i>
Installing the Visual Basic Demo	147
Visual Basic Demo Configuration Prerequisites	147
Using the Visual Basic Demo	148
Logging In	148
Send Alarms to OnGuard	149
Receive Alarms from OnGuard	149
Working with Cardholders	150
Integrating OnGuard with Active Directory	151
Index	153

DataConduIT is a platform for managing OnGuard and for integrating OnGuard with IT systems. DataConduIT provides access to ID management data, access control events, and real-time notification when changes are made to cardholders and their credentials. Administrators use this platform to write scripts and applications that improve the manageability of the OnGuard system and that provide new levels of integration between OnGuard and IT systems. These scripts and applications are written using a standard Microsoft API, Windows Management Instrumentation (WMI).

The following are some common scenarios where DataConduIT can integrate OnGuard with IT systems:

- When a cardholder is created, the IT department creates a Windows account for that person. The Windows account name is derived from the OnGuard cardholder name. The account is linked to the cardholder in the OnGuard software.
- A single script creates an LDAP account, a cardholder, a badge for this cardholder (with a badge type, assigning default access levels), and a link between the account and this cardholder.
- A single script terminates a person's access to all company resources by disabling all of the person's badge(s) and LDAP accounts.
- When a cardholder is granted access to an area, that cardholder is granted access to use the computers in that area.
- A cardholder enters the building under duress. The cardholder's LDAP accounts are disabled to prevent potential unauthorized use.
- A cardholder's phone number changes in the OnGuard software. The new phone number is propagated to the associated Windows account in the company's Active Directory.

Administrators can also write scripts and applications that interact only with the OnGuard software. Examples include command line tools that automate frequent administrative tasks and web user interfaces that provide thin-client access to ID management data. In addition, since DataConduIT is built using WMI technology, administrators can use WMI-enabled third-party management tools to manage OnGuard data and events.

All the dates and time fields reported by DataConduIT will be presented in Coordinated Universal Time (UTC time) which has a GMT offset of 0 minutes. When setting values of the time fields, GMT offset must be specified or time values should be in the UTC time as well.

Documentation Contents

This documentation package contains the following files and folders:

File	Description
DataConduIT.pdf	This manual
DataConduIT Samples\ASEC	Tools for using DataConduIT with the Active Script Event Consumer
DataConduIT Samples\JScript	Sample code in the manual (in JScript)
DataConduIT Samples\Solutions	Solutions for integrating OnGuard and Active Directory (in JScript)
DataConduIT Samples\VBDemo	The Visual Basic Demo application, which can be used to demonstrate some of the capabilities of DataConduIT. (For more information, refer to Appendix E: Visual Basic Demo on page 147.)
DataConduIT Samples\VBScript	Sample code in the manual (in VBScript)

IMPORTANT: All scripts and code (“sample code”) provided with this documentation are examples of how to use DataConduIT. This sample code is for educational purposes only and is not supported by Lenel.

Some sample code requires ADsSecurity.dll to be registered on the machine. You can learn more about this DLL from Microsoft Knowledge Base article Q251390, which is available at <http://support.microsoft.com/default.aspx?scid=kb;en-us;Q251390>.

Documentation Prerequisites

This guide assumes that the reader is familiar with Microsoft scripting languages such as VBScript and JScript. All sample code given in this guide is written in JScript, but samples in both JScript and VBScript are included separately with this documentation. Basic experience with object-oriented programming is also required.

Experience with Windows Management Instrumentation (WMI) is recommended but not required.

Definitions, Acronyms, Abbreviations

Class

“A template for a type of object.”¹ For instance, the Lnl_Reader class is a template for an access control reader.

Client

A script or application that accesses DataConduIT.

1. From the WMI documentation, URL below.

Hardware event

An event that is displayed in Alarm Monitoring. These events generally originate in the security hardware. An example is when a reader grants access to a cardholder.

Namespace

“A unit for grouping classes and instances to control their scope and visibility. Namespaces are not physical locations; they are more like logical databases containing specific classes and instances. Namespaces are represented by the __Namespace system class or a class derived from it.”¹

Object/Instance

“A representation of a real-world entity that belongs to a particular class. Instances contain actual data.”²

Person

A cardholder or visitor.

SDK

Software Development Kit.

Software event

An event that occurs when an object in OnGuard is added, modified, or deleted. Examples of such objects include cardholders, visitors, and badges.

WMI

Windows Management Instrumentation. “WMI is the Microsoft portion of the Distributed Management Task Force’s (DMTF) Web-Based Enterprise Management (WBEM) initiative and provides a set of interfaces for access to components that provide management capabilities across an enterprise.”³

References and Applicable Documents

Microsoft Scripting Technologies documentation, which is located in the MSDN library at <http://msdn2.microsoft.com/en-us/library/ms950396.aspx>.

Microsoft WMI documentation, which is located in the MSDN library at <http://msdn2.microsoft.com/en-us/library/aa394582.aspx>.

-
1. From the WMI documentation, located at <http://msdn2.microsoft.com/en-us/library/aa394582.aspx>.
 2. From the WMI documentation, located at <http://msdn2.microsoft.com/en-us/library/aa394582.aspx>.
 3. From the WMI documentation, located at <http://msdn2.microsoft.com/en-us/library/aa394582.aspx>.

DataConduIT is installed as part of a standard server installation.

Note that DataConduIT must be installed on the same machine as the Linkage Server if you want to receive events through DataConduIT. DataConduIT may be run on additional server machines as well, but you will not be able to register to receive events from DataConduIT on those machines.

DataConduIT runs as a Windows service under the Local System account. It does not run as an application. Since the Local System account does not have permissions on the local network, if your database is not on the same machine as DataConduIT you will need to ensure that your ODBC connection uses TCP/IP, not named pipes. Otherwise, DataConduIT will not be able to connect to the database.

License for DataConduIT

DataConduIT is a licensed feature. The DataConduIT license is count-based; you are licensed to have a certain number of clients. The number of clients you are licensed to use is displayed in the “Maximum Number of DataConduIT Clients” setting in the General section of the license. To view this setting, start to License Administration. or more information, refer to “Using OnGuard in the Supported Operating Systems” in the Installation Guide.

Authentication

When a client makes a call into DataConduIT, whether it is to view some data, add an instance of a class, register an event query, or simply to get a class definition, the first thing DataConduIT does is decide whether the client is permitted to perform the operation. To do this, DataConduIT checks which Windows account has made the DataConduIT call. This is the account that the script or application is running from, which is generally the account of the person logged on to the machine.

Once DataConduIT retrieves this account, it attempts to perform automatic single sign-on (SSO) using this account. This is the same SSO mechanism used by all OnGuard applications. If the SSO succeeds, then the client is logged on to the system as the appropriate OnGuard user. DataConduIT then uses the OnGuard user information to decide whether the client has permission to perform the

requested operation.

Note that to perform this authentication, the client application doesn't need to call any special "Logon" method. The authentication is done implicitly based on the account running the application.

It is not possible to use OnGuard internal authentication with DataConduIT. Automatic SSO is the only authentication mechanism. Therefore, to use DataConduIT, single sign-on must be configured. To configure single sign-on in OnGuard:

1. Add the directory that you wish to use. (For more information please refer to "Add a Directory" in the Directories Folder chapter of the System Administration User Guide.)
2. Link the user account that you want to use automatic single sign-on to a directory account. (For more information please refer to "Link a User Account to a Directory Account" in the Users Folder chapter of the System Administration User Guide.)

Each OnGuard software manual contains the "Log into the Application Using Single Sign-On" procedure. Refer to this procedure to log into OnGuard after single sign-on has been configured.

Authorization

For a user to be able to use DataConduIT, the user must have the DataConduIT service user permission. This permission may be set on the Software Options sub-tab of the System Permission Groups form in the Users folder in System Administration.

All functionality available through DataConduIT is controlled by the same permissions that you are already using to manage data in ID CredentialCenter. For instance, if you want to add a cardholder through DataConduIT, you must have the Add Cardholder user permission. If you want to view readers through DataConduIT, you must have the View Reader user permission.

Note: DataConduIT caches user credentials for one minute by default. This is done for performance reasons. (See [Tuning Parameters](#) on page 37 for information on how to change this default timeout.) Therefore, if a user is using DataConduIT and that user's permissions or segments change, the user will continue to have his old permissions until the one-minute timeout is reached.

Receiving Events

If you want to be able to receive events from DataConduIT, the "LS Linkage Server" service must be running. The Linkage Server sends events of all supported types to DataConduIT. The Linkage Server host name is set on the System Options form in System Administration.

In addition, if you would like to receive software events through DataConduIT, you need to select the **Generate software events** checkbox on the System Options form in System Administration.

Using DataConduIT from a Remote Computer

If you want to be able to use DataConduIT from a computer other than the one on which the DataConduIT service is running, you must first enable the appropriate WMI namespace permissions. To do this, open the Computer Management MMC snap-in. For more information, refer to "Using OnGuard in the Supported Operating Systems" in the Installation Guide.

Once opened, go to Services and Applications\WMI Control. Open the WMI control property page, and go to the Security tab. Select the root\onguard namespace, and click the security button. Once here, make sure that any account that needs to access DataConduIT remotely has the “Remote Enable” permission.

Note: You should not give the “Remote Enable” permission to users in any other namespace.

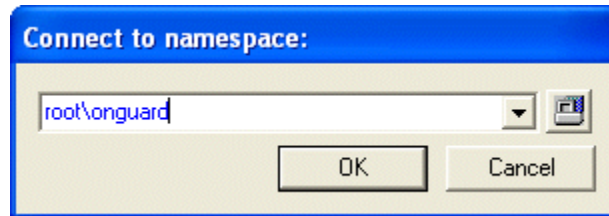
Viewing DataConduIT Classes with the Microsoft WMI SDK

Microsoft’s WMI Software Development Kit (SDK) is a useful tool for exploring the capabilities of DataConduIT. The SDK provides a convenient graphical user interface that allows users to view the WMI classes exposed by DataConduIT, to perform queries and to add, modify, and delete instances of these classes, and to register permanent event consumers that receive events from DataConduIT.

The WMI SDK may be downloaded from the MSDN subscriber downloads at <http://msdn.microsoft.com/downloads>. See the WMI SDK for instructions on installation procedures.

Note: You do not need the WMI SDK in order to use DataConduIT. The WMI SDK is a tool that can be helpful to developers who are writing DataConduIT scripts and applications.

Once you have installed the WMI SDK, open the WMI CIM Studio application. This application allows you to view and manage data through WMI. When you start this application, you will first need to select a namespace to which you want to connect. The namespace used by DataConduIT is called root\onguard. Enter this into the dialog and click [OK].

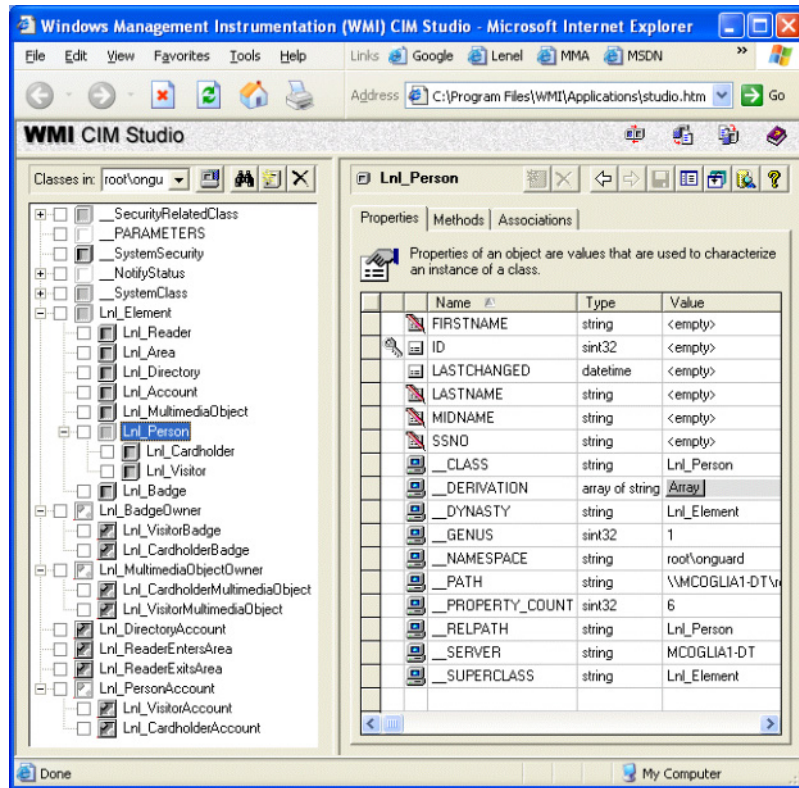


The WMI CIM Studio Login dialog will appear. Click [OK].

The main browser window should now display the contents of this namespace. On the left side of the window are all of the classes in the namespace. These include system classes, which are prefixed by two underscore characters, and classes provided by DataConduIT, which are prefixed with ‘Lnl_’. A class’s subclasses appear below the class in the tree. Expand nodes in the tree to view all of the classes provided by DataConduIT.

Note: If you do not see the Lnl_Person, Lnl_Cardholder, Lnl_Visitor, Lnl_Visit, and Lnl_Badge classes, then you have not correctly configured the user’s permissions to use DataConduIT.

On the right are all the properties of the currently selected class. System properties are prefixed by two underscore characters.



Note: Additional classes are available if the system is segmented.

Overview of DataConduIT Functions

DataConduIT provides access to the following objects:

Object(s)	Class	Properties	Operations
Cardholders and visitors	Lnl_Person and subclasses	System and user-defined	All
Badges	Lnl_Badge	System and user-defined	All
Visits	Lnl_Visit	System and user-defined	All
Cardholder directory accounts	Lnl_Account	System and user-defined	All
Cardholder photos and signatures	Lnl_MultimediaObject	All	All
Visit E-mail Recipients	Lnl_VisitEmailRecipients	All	View only
User-defined value types	Lnl_Building, Lnl_BadgeStatus, Lnl_Title, Lnl_Department, Lnl_VisitType, Lnl_Location	All	All

Object(s)	Class	Properties	Operations
Directories	LnI_Directory	All	View only
Panels	LnI_Panel	Essential	View only
Readers	LnI_Reader	Essential	View only
APB Areas	LnI_Area	Essential	View only
Alarms	LnI_Alarm	Essential	View only
Access Levels	LnI_AccessLevel	Essential	All
Access Level Assignments	LnI_AccessLevelAssignments	All	All
Access Groups	LnI_AccessGroup	Essential	View only
Badge Types	LnI_BadgeType	Essential	View only
Segments (in segmented systems only)	LnI_Segment and subclasses	Essential	View only
Manager	LnI_DataConduITManager	None	Custom
Sending Alarms to OnGuard	LnI_IncomingEvent	None	Custom
Mobile Verify	LnI_MobileVerify	None	Custom

DataConduIT also provides a number of association classes that relate these classes. For example, the LnI_BadgeOwner class relates badges with the cardholders and visitors that own them. Querying for all instances of LnI_BadgeOwner will return a list of associations between each badge and its owner.

DataConduIT provides access to the following events:

Event(s)	Class	Properties
Intercom events	LnI_IntercomEvent	All
Function execution events	LnI_FunctionExecEvent	All
Status changes	LnI_StatusChangeEvent	All
Video events	LnI_VideoEvent	All
Fire events	LnI_FireEvent	All
Transmitter events	LnI_TransmitterEvent	All
Other hardware events	LnI_OtherSecurityEvent	All
Access granted and access denied hardware events	LnI_AccessEvent	All
Cardholder and visitor software events Badge software events Cardholder directory account software events	__InstanceOperationEvent and subclasses	All properties listed above are in embedded instances. Event data includes previous and current instances for modification events.

For more details on these classes and their properties, refer to [Chapter 13: Reference](#) on page 71.

Connecting to DataConduIT

In order to access data and events through DataConduIT, you must first connect to DataConduIT. To connect to the namespace used by DataConduIT, root\onguard, you can use the GetObject() call from JScript or VBScript. For example, in JScript:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
```

Here, wbemServices is a **SWbemServices** COM component defined in the WMI Scripting Library. This component will be our main interface for accessing data and events from DataConduIT.

The ‘.’ in the above code sample means that you are connecting to the namespace on the local computer. To connect to DataConduIT on a remote machine, swap the name of the computer for the ‘.’.

Searching for Objects

Now that you are connected to DataConduIT, you can use the **SWbemServices** component to list and search for objects in the OnGuard software. **SWbemServices** provides a couple ways to search for objects. The simplest way is to use its InstancesOf() method. InstancesOf() is passed in a class name, and it returns a list of all the instances of that class. The client can then scroll through these instances and access their properties. For example, here is a simple script that prints the first and last names of

all the cardholders in OnGuard:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var cardholderSet = wbemServices.InstancesOf("Lnl_Cardholder");
for ( var e = new Enumerator( cardholderSet ); !e.atEnd(); e.moveNext() )
{
    var cardholder = e.item();
    WScript.Echo(cardholder.FirstName + " " + cardholder.LastName);
}
```

Let's examine the sample above in detail. On the first line, we connect to DataConduIT as described above. Next, we retrieve a list of all the instances of the Lnl_Cardholder class. (This list is an **SWbemObjectSet** component.) Now that we have this list, we iterate through it using the JScript Enumerator object. Each item in the list is a **SWbemObject** component, which is accessed using the enumerator's item() method. Finally, this **SWbemObject** (stored here in the cardholder variable) can be used to access all the properties of the particular instance. These properties are accessed simply by specifying the property name, as in cardholder.FirstName in the above example. Note that property names are case insensitive.

Accessing instance properties is straightforward for text and numeric fields. Text fields are represented as the string property type, and numeric fields are represented as the sint32 type for integers, and the real64 type for floating point numbers. Date fields are represented as the datetime type, which is actually a string containing the date in the DMTF format. This format is described in the Microsoft WMI documentation.

List fields, such as those configured through the List Builder, are specified as the database ID of the list value. This ID is mapped to the list value using the Values and ValueMap property qualifiers. Examples in the supplied sample code show how to enumerate and find the list values in these qualifiers.

The InstancesOf() allows you to retrieve all instances of a particular class, but what if you want to perform a more complicated query? This is done using ExecQuery() method in the **SWbemServices** component. Queries are specified in the WMI Query Language (WQL), which is a subset of the Structured Query Language (SQL) supported by most databases. One main difference between a SQL query and a WQL query is that the FROM clause in a SQL query contains a list of table names, whereas in WQL it contains a single class name. To give you a feel for WQL, here are a few WQL queries that you could use with DataConduIT:

```
Find all directories with a hostname of "windows.mydomain.com":
    select * from Lnl_Directory where HostName="windows.mydomain.com"
Find all people (cardholders and visitors) whose last name is not "Lake":
    select * from Lnl_Person where LastName!="Lake"
Find all active badges that are APB exempt:
    select * from Lnl_Badge where Status=1 and APBExempt = TRUE
Find all readers:
    select * from Lnl_Reader
```

The second example demonstrates how you can specify a superclass in the query. In this case, Lnl_Person is the superclass of the Lnl_Cardholder and Lnl_Visitor classes. When you specify a superclass, all instances of that class and its subclasses matching the query will be returned.

Note that executing the fourth query is equivalent to calling `InstancesOf("Lnl_Reader")`.

Let's take a look at how we would use a WQL query with the `ExecQuery()` method:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var cardholderSet =
    wbemServices.ExecQuery("select * from Lnl_Visitor where
Zip='14534'");
for ( var e = new Enumerator( cardholderSet ); !e.atEnd(); e.MoveNext() )
{
    // access properties in the same way as above...
}
```

This sample searches for all visitors who have a zip code of 14534. It then enumerates these visitors as in the previous example.

WQL supports a subset of the regular SQL syntax. See the Microsoft WMI documentation for more information.

You can also access a single instance of a class in `DataConduIT` by using the `Get()` method in **`SWbemServices`**. The `Get()` method can be used to get a class definition or an instance of a class. Here, we'll focus on using it to get an instance. The `Get()` method takes as a parameter an object path, which is basically the class name plus a list of the class keys and their values. You can determine which class properties are keys by looking for the "key" property qualifier. In the WMI SDK, key properties are identified by a key symbol next to the property name.

For instance, the key property for `Lnl_Person` is `ID`. (Note that `ID` is the internal database ID, not the person's social security number or other identification number - that property is named `SSNO`.) Here's an example of how you would get a cardholder if you know the cardholder's ID:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var cardholder = wbemServices.Get("Lnl_Cardholder.ID=1");
// access properties in the same way as above...
```

If the class has multiple key properties, such as `Lnl_Reader`, those properties would be separated by commas:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var reader = wbemServices.Get("Lnl_Reader.PanelID=1,ReaderID=1");
// ...
```

Adding Objects

Some classes in `DataConduIT` allow you to add, modify, and delete instances of those classes. Adding a new instance of a class takes four steps. First, you get the class for which you want to create an instance. Second, you spawn an instance of that class. Third, you assign values to properties of that

instance. Finally, you tell DataConduIT to add the instance. Here's a code sample that adds a new cardholder:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var cardholderClass = wbemServices.Get("Lnl_Cardholder");
var cardholder = cardholderClass.SpawnInstance_();

cardholder.FirstName = "John";
cardholder.LastName = "Smith";
cardholder.City = "Rochester";
var cardholderPath = cardholder.Put_();

cardholder = wbemServices.Get(cardholderPath);
// use cardholder object...
```

Earlier, it was mentioned that the Get() method can be used to get a class definition. Line 2 of this sample shows how this is done. Instead of listing the key properties in the object path, only the class name is specified. Line 3 uses this class definition to create an instance of the class.

Lines 4-6 assign values to the properties of this new instance. Properties are used here to set values just as in [Searching for Objects](#) on page 19 where they were used to get values.

Next, line 7 actually commits the changes. Note that if the Put_() method is not called, the instance will not be sent to DataConduIT, and therefore the change will not be made in the OnGuard database. If successful, the Put_() method returns the object path to the newly created instance. If you plan to use this instance for further operations, you should re-get the instance using this path. This is because DataConduIT will set default properties for you, and those values will not be reflected in the instance that you called the Put_() method on. To get those default values, you need to re-get the instance from DataConduIT.

Note that the above example did not assign a value to the ID key property for the Lnl_Cardholder instance. This is because DataConduIT auto-generates the value for you.

Modifying Objects

The process of modifying objects in DataConduIT is similar to the process of adding them. First, you search up the object that you want to modify. This can be done in any of the ways described in [Searching for Objects](#) on page 19. Next, you set new values to the object's properties. Finally, you call the same Put_() method that was used for adding objects. Here's an example:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var visitor = wbemServices.Get("Lnl_Visitor.ID=2");

visitor.Address = "1050 Pittsford-Victor Road";
var visitorPath = visitor.Put_();

visitor = wbemServices.Get(visitorPath);
```

As you can see, modifying an object is very similar to adding one. Just as when we added a new object, we re-get the object after we have committed our modifications. This makes sure that all fields are refreshed. For instance, DataConduIT sets the LastChanged property on instances of Lnl_Cardholder, Lnl_Visitor, and Lnl_Badge when an instance of one of those classes is added or modified. You must re-get the object in order to view the updated LastChanged time.

Deleting Objects

There are two ways to delete an object in DataConduIT. The easiest way is to search up the object you want to delete, and then just call the Delete_() method on that object. For example:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
var visitor = wbemServices.Get("Lnl_Visitor.ID=2");
visitor.Delete_();
```

You can also delete an instance if you know its object path. The example below is equivalent to the one above, but it is more efficient because the actual visitor object is never requested:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
wbemServices.Delete("Lnl_Visitor.ID=2");
```

Features and Limitations

The following features and limitations are specific to class.

Cardholders and Visitors

Each cardholder and visitor instance has all of its user-defined fields (UDFs) exposed through DataConduIT. This includes system fields such as first name (FIRSTNAME), last name (LASTNAME), social security number (SSNO), and internal ID (ID). All fields except for the internal ID and last changed timestamp are available for read/write access, subject to additional UDF validation and field/page viewing permissions.

If cardholders/visitors are segmented, an additional property named PrimarySegmentID will be made part of the Lnl_Cardholder/Lnl_Visitor class. If the client is a member of only one segment, this property will default to that segment ID. Otherwise, the client must specify the primary segment ID when a new cardholder/visitor is added.

Badges

Each badge instance has all of its UDFs exposed through DataConduIT. This includes system fields such as badge ID (ID), badge type (TYPE), badge status (STATUS), and the internal ID (BADGEKEY). All fields except for the internal ID, number of badge prints, last changed, and last printed timestamps are available for read/write access subject to the validation described above.

The PIN code is exposed in a manner similar to the way it is done in ID CredentialCenter. You can set the badge PIN code by setting the property during an add or modify operation. However, if you search up a badge and attempt the read the PIN code, the property will always contain a null value.

A client will be able to assign access levels to a new badge by giving it a badge type. The new badge will be assigned the default access levels for that badge type.

In a segmented system, the client cannot change the badge type if it controls a different set of segments than the previous badge type. This is because changing the badge type of a badge could possibly remove access levels from that badge without user confirmation.

Directory Accounts

Adding an instance of Lnl_Account is equivalent to linking a directory account to a cardholder or visitor in ID CredentialCenter. Similarly, deleting an instance is equivalent to unlinking the account. When adding an instance of Lnl_Account, all fields except for the ID are required. The AccountID property refers to the value of the LDAP attribute provided in the Lnl_Directory.AccountIDAttr property. For Microsoft Active Directory accounts, this defaults to the account security identifier, or SID. Other LDAP directories will probably use a different LDAP attribute.

Visits

Each visit instance has all of its UDFs exposed through DataConduIT. This includes system fields such as host id (CARDHOLDERID), type (TYPE), visitor id (VISITORID), and the internal ID (ID). All fields except for the internal ID, last changed, time in, and time out are available for read/write access subject to the validation described above.

Once a visit has been signed in, scheduled time in cannot be changed, nor can the cardholder or visitor of the visit, same thing with signing out a visitor.

E-mail recipients configured through Lnl_Visit cannot be viewed through Lnl_Visit; Lnl_VisitEmailRecipient must be used for viewing.

Multimedia Objects

Signatures and photos are exposed, however, biometric templates are not. Trying to add/delete/view biometric templates through DataConduIT will result in an error.

User-Defined List Values

All user-defined list (populated via List Builder) are available for view/add/modify/delete. The only values that cannot be modified are:

- Active BadgeStatus (ID = 1)
- Supervisor Two Man Type
- Team Member Two Man Type

The previous section described how to receive and modify data using DataConduIT. This section describes how to receive real-time events. DataConduIT produces two types of events - hardware events and software events. Hardware events are generally events that originate in the access control hardware. Software events occur when data in the OnGuard database changes.

There are two ways to receive events from DataConduIT: via temporary event consumers and via permanent event consumers. A temporary event consumer registers to receive events when it starts, receives those events while running, and then ends its registration when it terminates. A permanent event consumer submits an event registration to WMI and binds that registration to a particular COM component. Whenever WMI receives an event that matches the registration, that component is created and passed the event. This occurs until the event registration is deleted or unbound from the component.

Note: In order to receive DataConduIT source events, add at least one online panel to the same monitor zone as the source.

Here is an example of a simple temporary event consumer:

```
var wbemServices = GetObject("winmgmts://./root/OnGuard" );
var sink = WScript.CreateObject( "WbemScripting.SWbemSink", "SINK_" );
wbemServices.ExecNotificationQueryAsync(
    sink, "SELECT * FROM Ln1_AccessEvent" );
var wshShell = WScript.CreateObject( "WScript.Shell" );
wshShell.Popup( "Click OK to stop listening for events..." );
sink.Cancel();

function SINK_OnObjectReady( wbemObject, asyncContext ) {
WScript.Echo("Hardware event received: " + wbemObject.Description);
}
```

This sample demonstrates the three steps necessary for a temporary event consumer to receive events from DataConduIT. First, the client creates an event sink object. Second, the client registers an event query describing which events the client would like to receive. Like data queries, this event query is

written in WQL. Unlike data queries, this query does not return events from WMI immediately. Instead, it tells WMI which events it wants to receive when WMI gets them in the future.

The **ExecNotificationQueryAsync()** method also takes in the event sink object. This ensures that the function **SINK_OnObjectReady()** is called whenever WMI receives an event that matches the event query. The first argument to this function is the event object itself. The properties of the event object can then be read. (Setting the properties of an event object has no effect.)

The final step is to unregister the event query, which is accomplished by calling the event sink's **Cancel()** method.

The following sections describe particular features of registering for and receiving hardware and software events, as well as how to use permanent event consumers with DataConduIT.

Registering to Receive Hardware Events

DataConduIT provides access to all OnGuard events in the system. These events are accessed using the general WMI class `Lnl_SecurityEvent`. This class has several subclasses that can be used to simplify filtering specific types of events. For example, the `Lnl_AccessEvent` subclass can be used to receive access granted and access denied events in the system. The `Lnl_IntercomEvent` subclass can be used to only receive intercom related events. Objects retrieved using `Lnl_SecurityEvent` must be set to the specific subclass object in order to retrieve specific properties. You can use the `__CLASS` property to identify which subclass object to use in order to retrieve all of the event's properties.

The sample code in the previous section showed how to receive all access events. If the system is not segmented, this event query will always succeed for users that are permitted to use DataConduIT. In a segmented system, users can only receive events from hardware in segments to which they have access. Due to the implementation of WMI, when you register an event query, you must be able to receive all possible instances of that event. Therefore, you need to make sure that your query explicitly specifies the segments, readers, and/or panels to which you have access. If you register for events that you don't have access to, you will receive an access denied error when you try to register your event query. Here are a few sample hardware event queries:

```
Receive all access events at all readers in the segment with ID 1:
    select * from Lnl_AccessEvent where SegmentID=1
Receive all access events at all readers in segments with ID 1 or ID 2:
    select * from Lnl_AccessEvent where SegmentID=1 or SegmentID=2
Receive all access events at all readers on the panel with ID 8:
    select * from Lnl_AccessEvent where PanelID=8
Receive all access events for the reader with ID 5 on the panel with ID 8:
    select * from Lnl_AccessEvent where DeviceID=5 and PanelID=8
Receive all events
    select * from Lnl_SecurityEvent
Receive only intercom related events
    select * from Lnl_IntercomEvent
```

DataConduIT will obtain the set of segments specified by the event query, and it will make sure that the user has permissions to receive events from all of these segments.

Receiving Hardware Events

Once the event query is registered, DataConduIT will begin sending hardware events to the client. The `Lnl_AccessEvent` class has a number of properties that can be accessed by the client. These properties are generally self-explanatory. For details, see the description qualifier on the `Lnl_Event`, `Lnl_SecurityEvent`, and `Lnl_AccessEvent` class definitions.

Registering to Receive Software Events

Software events are instances of the standard WMI intrinsic event classes, namely `__InstanceOperationEvent` and its subclasses. The `__InstanceOperationEvent` class has one property, `TargetInstance`, which contains the instance that was added, modified, or deleted. If the instance was modified, the `__InstanceModificationEvent` event subclass also contains the previous version of the instance in its `PreviousInstance` property. Both the `TargetInstance` and `PreviousInstance` properties are of type object, meaning that they contain the embedded WMI instances of the affected class.

As was the case with hardware events, you must only register to receive those software events for which you have permission to receive. In general, you can view a software event for an object if you could view that object normally. For instance, if you do not have permission to view visitors, then you cannot receive software events indicating that a visitor was created, modified, or deleted. If you don't have access to segment A, then you can't receive software events for objects in segment A. Furthermore, if you do not have view permissions for each property of a class, then you can't receive software events for instances of that class. For example, if you can't view the visitor address field (set through the field/page permission groups in System Administration), you can't view visitor software events.

The following classes are supported by DataConduIT for software event registration: `Lnl_Cardholder`, `Lnl_Visitor`, `Lnl_Badge`, and `Lnl_Account`.

Common software event queries that you might use include:

```

Receive an event whenever a cardholder is added, modified, or deleted:
    select * from __InstanceOperationEvent where TargetInstance ISA
    "Lnl_Cardholder"
Receive an event whenever a badge is printed:
    select * from __InstanceModificationEvent where TargetInstance ISA
    "Lnl_Badge" and TargetInstance.Prints > PreviousInstance.Prints
Receive an event whenever a badge changes from active to inactive:
    select * from __InstanceModificationEvent where TargetInstance ISA
    "Lnl_Badge" and TargetInstance.Status!=1 and PreviousInstance.Status=1
Receive an event whenever a cardholder, visitor, or badge is created:
    select * from __InstanceCreationEvent where TargetInstance ISA
    "Lnl_Person" or TargetInstance ISA "Lnl_Badge"

```

The first example demonstrates how the `__InstanceOperationEvent` class can be used to receive events for all add, modify, and delete operations. It is also the first example of the ISA operator. As you might guess, the ISA operator is used to query for an object only when its class name equals the specified name. To successfully register this query, the user must be an All Segments user with the View Cardholder permission and the view permission to all cardholder fields.

The second and third examples show how properties of the `TargetInstance` and `PreviousInstance` objects can be used as part of an event query. To successfully register these two queries, the user must be an All Segments user with the View Badge permission and view permission for the appropriate badge field (prints or status). The last example demonstrates that the ISA operator can be used with a superclass (`Lnl_Person`) to indicate that it and all of its subclasses are included in the query. It also demonstrates that the ISA operator can be used with the regular boolean (“and”, “or”) operators. A user registering this query must be an All Segments user with the View Cardholder, View Visitor, and View Badge permissions, and must be able to view all properties of those classes.

Receiving Software Events

As mentioned above, the `TargetInstance` and `PreviousInstance` contain all the data in the current and previous instances. This data can then be used in the event handler to perform other actions. For instance, when a cardholder is created, a script could use the cardholder’s name and department to create an LDAP account for that cardholder and link it back to the cardholder by creating an instance of the `Lnl_Account` class.

Assuming the software event feature is enabled (refer to [Receiving Events](#) on page 14), software events are generated whenever changes are made to particular tables in the database. This includes changes made by all OnGuard applications, and even changes made by directly editing data in the database.

There are two situations, however, in which software events will not be generated. The first is when a truncate table SQL command is issued on a table. In this case, no cardholder and visitor deletion events will be fired. The second case is in a full download from an enterprise master to an enterprise region. Software events will be fired on the region in an incremental download.

Using Permanent Event Consumers with DataConduIT

The [Using DataConduIT to Receive Events](#) on page 25 gave an example of how to use a temporary event consumer to receive events from DataConduIT. Temporary event consumers only run when the consumer is running. Therefore, unless this consumer is running in a service, a user must be physically logged on to a machine and running that consumer. A permanent event consumer can be setup once, and WMI will invoke it whenever a matching event is fired. Therefore, no one needs to be logged onto the machine where DataConduIT is located for a permanent event consumer to work.

Microsoft provides a permanent event consumer called the Active Script Event Consumer (ASEC). The ASEC runs a script (JScript or VBScript) when an event is received. This is exactly the functionality that many customers want. Please refer to the Microsoft WMI documentation to see how to use the ASEC.

There are a couple things to note when using the ASEC with DataConduIT. First, the ASEC is not installed by default in the `root\onguard` namespace. To install it, find the `DataConduIT\Samples\ASEC\asec-onguard.mof` file provided with this documentation and run “`mofcomp asec-onguard.mof`.” This will install the ASEC in the `root\onguard` namespace. Also provided with this documentation is a utility, `regpermscript.exe`, that will help you install scripts for use with the ASEC. The utility takes as parameters an event query to register and the script file containing the script to run when an event is received. Note that the currently logged on user must be authorized to register this event query according to the rules in [Registering to Receive Hardware Events](#) on page 26 and [Registering to Receive Software Events](#) on page 27 of this user guide.

A second note on ASEC and DataConduIT is that when the ASEC runs your script, it will be running under the security context of the WMI service. If your script tries to connect back to DataConduIT, DataConduIT will try to use single sign-on to log on, looking for the user that is linked to the LocalSystem account on the local machine. If it doesn't find such an account, your call will fail.

Unfortunately, WMI does not allow you to connect to it with an alternate username and password on the local machine. Therefore, the best way to resolve this situation is to link the LocalSystem account to a user in the OnGuard software. To do this, you first create a Windows Local Accounts directory for the machine on which DataConduIT is running. Then, link the LocalSystem account in this directory to a user. Your script will now execute under the user to which you linked the account.

Scripts run by the ASEC cannot interact with the desktop, so they cannot write to the console (e.g. using `Echo()`) or show UI components (e.g. using `MsgBox()`). If an error occurs in the script, the error will not be displayed to the screen. Instead, it will be written to one of the standard WMI error logs.

Using DataConduIT to Send Alarms to OnGuard

DataConduIT provides the capability of sending alarms to the Alarm Monitoring application. These alarms are also logged to the OnGuard database just like other alarms.

It is necessary to first setup a DataConduIT Source using System Administration before using this capability of DataConduIT. DataConduIT will use this source as the device to display alarms for in Alarm Monitoring. For more information, refer to [Add a DataConduIT Source](#) on page 56.

After configuring the DataConduIT Source, you should also add any DataConduIT Device and DataConduIT Sub-Device downstream devices in System Administration. Use of devices and sub-devices is optional. OnGuard uses devices and sub-devices to report alarms for DataConduIT Source child and sub-child devices in Alarm Monitoring. For more information, refer to [Add a DataConduIT Device](#) on page 59 and [Add a DataConduIT Sub-Device](#) on page 61.

Sending alarms to Alarm Monitoring is very simple. Here is an example using JavaScript:

```
var wbemServices = GetObject("winmgmts://./root/onguard");
oReg = wbemServices.Get("Lnl_IncomingEvent");
oMethod = oReg.Methods_.Item("SendIncomingEvent");
oInParam = oMethod.InParameters.SpawnInstance_();
oInParam.Source = "DataConduIT Source 6";
oInParam.Description = "Test Event From DataConduIT";
wbemServices.ExecMethod("Lnl_IncomingEvent", "SendIncomingEvent",
oInParam);
```

The above sample will display and log an alarm with the description “Test Event From DataConduIT” from controller name “DataConduIT Source 6”. This sample assumes System Administration was used to create a DataConduIT Source called “DataConduIT Source 6” and demonstrates the five steps necessary for sending an alarm to Alarm Monitoring. First, the client gets an instance of Lnl_IncomingEvent object. Second, the client gets the “SendIncomingEvent” method referred to by oMethod. Third, the method is used to retrieve a parameter object oInParam. The fourth step is simply set the Source and Description properties of the oInParam parameter. The Source refers to the DataConduIT source setup in System Administration. The Description property is the actual text of the alarm that will be displayed in Alarm Monitoring and logged into the OnGuard database. The fifth and final step is simply execute the method using ExecMethod.

The `LnI_IncomingEvent` object has no properties and currently supports the methods “`SendIncomingEvent`” and “`AcknowledgeAlarm`”. For more information, refer to [LnI_IncomingEvent](#) on page 87.

The DataConduIT `SendIncomingEvent` method allows the ability to generate Access Granted and Access Denied events for a DataConduIT Source, Device and SubDevice. This is made possible via the following additional optional parameters that may be specified to the `SendIncomingEvent` method: `IsAccessGrant`, `IsAccessDeny`, `BadgeID`, and `ExtendedID`.

If ‘`IsAccessGrant`’ is set to true, the ‘Granted Access’ event will be reported for the DataConduIT Source, Device or SubDevice specified in the script. Similarly, if ‘`IsAccessDeny`’ is set to true, the ‘Access Denied’ event will be reported. If both of these are set to true, the method will fail since only one of these can be set to true at a given time (i.e., they are mutually exclusive). For more information, refer to [Generating Access Granted and Access Denied Events](#) on page 89.

The process is similar if the name of the Source and Device parameters correspond to the name of an access panel and reader respectively. OnGuard checks to see if the DataConduIT Source name provided matches a DataConduIT Source. If not, then a check is made to see if it matches the name of a Lenel access panel. If so, OnGuard checks the Device parameter and see if it matches the name of a reader assigned to the access panel. If these conditions are met, the ‘Granted Access’ or ‘Access Denied’ events are reported based on how ‘`IsAccessGrant`’ and ‘`IsAccessDeny`’ are set.

The `BadgeID` or `ExtendedID` parameter can be specified when either ‘`IsAccessGrant`’ or ‘`IsAccessDeny`’ are set to true to report an event for a specific OnGuard cardholder. `BadgeID` is not required when using ‘`IsAccessGrant`’ or ‘`IsAccessDeny`’.

MobileVerify is a feature that allows the cardholder view in OnGuard to make grant and deny decisions similar to a reader. DataConduIT has an Lnl_MobileVerify object that provides the ability to determine the configuration settings of MobileVerify and also help make grant or deny decisions. Here is an example using Lnl_MobileVerify:

```
var wbemServices = GetObject("winmgmts://./root/onguard");

var mvClass = wbemServices.Get( "Lnl_MobileVerify" );
var mv = mvClass.SpawnInstance_();
oMethod = mv.Methods_.Item("RecommendProperties");
oOutParam = oMethod.OutParameters.SpawnInstance_();
var mvResult = wbemServices.ExecMethod("Lnl_MobileVerify",
"RecommendProperties", oOutParam);
WScript.Echo("LogicalName: " + mvResult.LogicalName + "\n" +
"AssociatedDropdown: " + mvResult.AssociatedDropdown + "\n" +
"DenyText: " + mvResult.DenyText + "\n" +
"DenyColor: " + mvResult.DenyColor + "\n" +
"GrantText: " + mvResult.GrantText + "\n" +
"GrantColor: " + mvResult.GrantColor + "\n"
);
```

The sample above will retrieve important configuration settings about the MobileVerify feature and display them. The other supporting methods for the MobileVerify feature are IsGrant, LogGrant, LogDeny, and SystemSetting. Programmers can use these methods to simulate the MobileVerify feature in other applications. For more information about the methods related to MobileVerify, refer to [Chapter 13: Reference](#) on page 71.

Receiving Error Information from DataConduIT

DataConduIT performs authorization and validation checks on all incoming queries and requests to write data to the OnGuard software. If these checks do not pass or some other type of error occurs, an error code and message will be returned to the client. To retrieve the error message, you need to use the **SWbemLastError** object. Here's an example that demonstrates its use:

```
try {
    // do something that would cause an error...
}
catch (e) {
    var extStatus = new ActiveXObject("WbemScripting.SWbemLastError");
    if (extStatus != null && extStatus.Description != null) {
        WScript.Echo("Error: " + extStatus.Description);
    }
    else { throw e; }
}
```

The code sample above catches an error that occurs in DataConduIT. Next, it creates an instance of the **SWbemLastError** object and tries to retrieve a detailed error message from it, stored in its Description property. (Note that the extStatus object is actually an instance of the Lnl_Error WMI class, if such an error was returned by DataConduIT.) If a detailed description exists, it is printed out. Otherwise, if some other type of error occurred, such as a scripting error, that error will be printed out to the command line when the error is re-thrown.

DataConduIT reports the correct WMI error codes from all its functions. WMI error codes and their meanings can be found in the Microsoft WMI documentation reference.

Before Calling Technical Support

DataConduIT relies on several configuration options and environment settings of both OnGuard and the operating system used. If you are experiencing problems, please be sure to do the following BEFORE contacting technical support:

1. Consult the list of common DataConduIT problems. For more information, refer to [Appendix C: Common DataConduIT Problems](#) on page 143.
2. If the list of common problems did not provide a solution, perform all steps in the pre-call checklist. For more information, refer to [Appendix D: Technical Support Pre-Call Checklist](#) on page 145.

This will help technical support more accurately identify the problem and provide some quick potential solutions to the problem you are experiencing.

Error Logging

DataConduIT maintains an error log in the standard WMI logging directory. This directory is located at:

- **<windows directory>\system32\wbem\logs**

or, if using a 64-bit operating system at:

- **<windows directory>\SysWOW64\wbem\Logs**

The log file is named **DataConduIT.log**. Any errors that occur in DataConduIT are logged to this file, along with the data and time that they occurred. This includes errors that can be retrieved from DataConduIT using the **SWbemLastError** object described in the previous section.

OnGuard allows you to configure the filename of the DataConduIT log file as well as how verbose the logging is. Both of these parameters are configured in the registry on the machine where DataConduIT is running. Both registry values are located in the registry at:

- HKEY_LOCAL_MACHINE\Software\Lenel\OnGuard\DataConduIT

or, if using a 64-bit operating system at:

- HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Lenel\OnGuard\DataConduIT

(Note that the DataConduIT key does not exist by default - you will have to create it.)

The log file filename is stored in the value “DebugFile” in this key. This is the full path to the log file, such as **c:\program files\OnGuard\DataConduIT.log**. The logging level is stored in the value “DebugLevel” in this key. Possible values are 0 (normal/default), 1 (verbose), 2 (extra verbose). Note that when DebugLevel is set to 1 or 2, the log file can become large very quickly. Therefore, the DebugLevel should only be set above 0 when trying to debug an error. Note that the debug logging level must be a DWORD for the process to work correctly.

If you need to call tech support regarding a DataConduIT issue, you should first reproduce your particular error while the DebugLevel is set to 2. Next, create a ZIP archive containing the contents of the WMI log folder, the DataConduIT log file, and the Lenel error log. This will be very helpful to tech support as they help you with your issue.

Changing the Database Connection Pool Time

DataConduIT uses a database connection pool. A connection in the pool is closed after a certain period of inactivity. This period of time is specified in the value “DATABASETIMEOUT” in the HKEY_LOCAL_MACHINE\Software\Lenel\OnGuard\DataConduIT registry key. (Note that the DataConduIT key does not exist by default - you will have to create it.)

By default, the DATABASETIMEOUT value is not specified in the registry, and the timeout value is 5 minutes. If the DATABASETIMEOUT value is specified in the registry, the time specified (in seconds) will be used for the timeout value.

Tuning Parameters

DataConduIT allows administrators to tune some parameters that it uses for general operation and for communication with other servers. All of these parameters are stored in the database in the LNLCONFIG table. These parameters are described the table below. ID refers to the value in the LNLCONFIG.LNLCONFIGID column. The default value is the value assigned to the parameter if the LNLCONFIG table does not contain a record with this ID.

ID	Default Value	Description	Used By
33	60	Number of seconds for which DataConduIT caches user logon credentials. After this time, the cached credentials are refreshed from the database.	DataConduIT
34	60	Number of seconds for which DataConduIT caches its panel/segment ID map.	DataConduIT
35	15	Number of seconds in polling interval for software events by the Linkage Server.	Linkage Server
36	3	Number of seconds between which changes to tables for the same object are considered part of the same software event.	Linkage Server
37	10	Number of seconds between attempts by DataConduIT to contact the Linkage Server to notify it of WMI event registrations.	DataConduIT
38	30	Number of seconds after startup that the Linkage Server waits to receive event registrations from DataConduIT servers.	Linkage Server
39	3600	Number of seconds for which DataConduIT caches class definitions for dynamically generated classes.	Linkage Server

IMPORTANT: These configuration parameters should only be set by system administrators when trying to correct a problem.

Setting or changing any of these tuning parameters requires a restart of the appropriate server to take effect.

Stopping and Restarting the DataConduIT Service

Stopping and restarting the DataConduIT service is generally unnecessary. During normal operation, the server should be left running at all times. Although DataConduIT is installed as a manually-run service, WMI will start it automatically whenever it has a request to make. This includes a request for data as well as event query registrations.

In a few limited circumstances, however, you will need to stop and restart DataConduIT to allow it to retrieve new configuration information. DataConduIT needs to be stopped and restarted after any of the following changes are made:

- You change the data source DSN in your **ACS.INI** file. For more information, refer to the *Configuration Editor* appendix in the *Installation Guide*.
- You modify a cardholder, visitor, or badge layout in FormsDesigner.
- You change any of the tuning parameters (discussed above) that DataConduIT uses.
- You install a new license.

Note: If you have any event consumers running and you stop DataConduIT, WMI will automatically restart DataConduIT after a couple seconds.

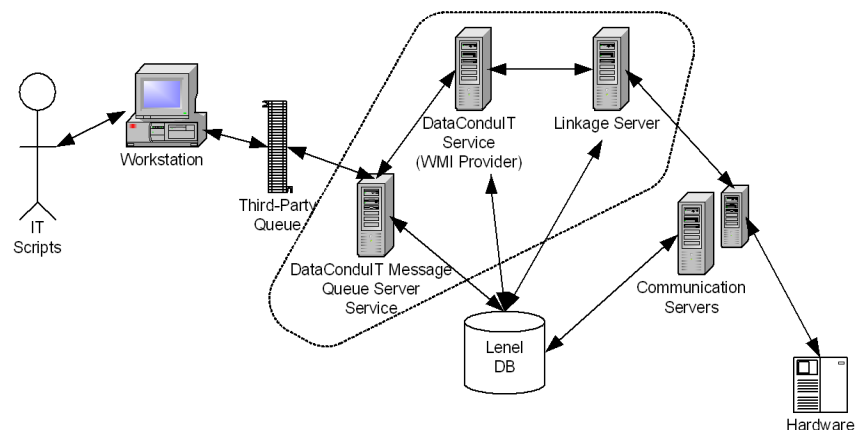
Getting Started with DataConduIT Message Queues

DataConduIT can be used alone, or it can be used in combination with message queues. Message queues are used to store DataConduIT software events. When message queues are used, the LS DataConduIT Message Queue Server service is used to package DataConduIT events into XML and send them across the queue. The service also receives XML and packages it up into DataConduIT requests.

The LS DataConduIT Message Queue Server service uses Windows Management Instrumentation or WMI for short to talk to DataConduIT. WMI is a Windows service that allows providers to expose application data and events to consumers. Microsoft uses WMI to expose information about the local machine's installed hardware and software, performance statistics, registry entries, Active Directory data, and much more. WMI ships with Windows, and it may be installed on Windows as a separate software package. Data exposed through WMI can be accessed by any COM-capable language, such as C++, Visual Basic, and VB Script.

Since the DataConduIT service is implemented as a WMI provider, it allows access to OnGuard cardholders, badges, photos, and linked accounts, through a queue. OnGuard hardware events are also exposed.

The picture below gives a high-level overview of how the DataConduIT and DataConduIT Message Queue Server services are related to the other major parts of the OnGuard software.



DataConduIT Message Queue runs as a Windows service on a machine. The service registers with DataConduIT who in turn registers with the Linkage Server to receive events. The Linkage Server receives hardware events by contacting all the communication servers in its region. The Linkage Server receives software events directly from the OnGuard database.

Note that while the Linkage Server, the DataConduIT Service, the DataConduIT Message Queue Service and the client workstation are pictured as residing on different machines, the setup has all four running on the same computer. Also, note that DataConduIT Message Queue requires/depends on a third party queue system running on the machine that is to receive XML packages from the DataConduIT Message Queue service. The workstation and the third party queue system are shown separately but actually reside on the same machine.

Overview of DataConduIT Message Queue Functions

The DataConduIT Message Queue service includes most of the capabilities of DataConduIT, including:

- Send/receive Cardholder data
- Send/receive Visitor data
- Send Cardholder/Visitor Photos
- Send/receive Badge data
- Send/receive Linked Account data
- Send hardware events

Data will be sent/received when any of the above are created, modified, or deleted. However, the following DataConduIT capabilities are not supported:

- View directory definitions
- View information about readers, anti-passback areas, and the relationships between them
- View information about segments and segment groups

All photos are sent with the cardholder/visitor data, and are sent with base-64 encoding. Photos are not sent when a cardholder is deleted, because when a cardholder is deleted, the photo is also deleted.

Each message sent across the queue contains either a hardware event or all information for one cardholder. So, if a badge is modified, the message contains the cardholder, the badge, the pictures (if configured to do so), and the linked accounts along with what has changed.

Directories, readers and segments cannot be viewed via the queue. However, these are available as enumerations inside the XML Schema just like the user-defined field dropdowns are, and the XML message contains the ID of the referencing object.

Supported Queue Types

OnGuard supports IBM WebSphere® MQ, formerly known as MQSeries. You must purchase the IBM WebSphere MQ software to setup DataConduIT Message Queue. IBM WebSphere MQ supports two types of message queues: incoming and outgoing. A queue must be designated as incoming or outgoing; it cannot be both.

Incoming queues allow you to send a request to the OnGuard software. Incoming queues are used to receive cardholder data, visitor data, cardholder/visitor photos, badge data, and linked account data from the user.

Outgoing queues allow OnGuard to send messages to you. Outgoing queues are used to send cardholder data, visitor data, cardholder/visitor photos, badge data, linked account data, and hardware events to the user.

Outgoing Queue Overview

DataConduIT Message Queue takes DataConduIT events, packages them up in XML, and sends them out. If a cardholder is added, DataConduIT will take the add cardholder object and send it out to whomever wants to know about it. You might tell DataConduIT to give you all the cardholder adds. If you're just using DataConduIT, you must install scripts that say what you want to register for, what you want to know about, and you're sent limited information. For example, if you're sent a badge add, you're sent only badge properties. You wouldn't know what the cardholder properties are. You then have to know that the EMP ID, which is something that is only used by OnGuard, is paired to this cardholder. This puts the responsibility on you to look up the EMP ID and then ask DataConduIT to give you the cardholder object. Basically, you have to talk back and forth to DataConduIT.

DataConduIT Message Queue is designed to take away that layer. You do not need to talk to DataConduIT, and you do not need to run scripts; everything is automated through the user interface. When you tell the user interface what kind of information you want, it will automatically set you up to get that information, and all the information will be put on these queues, instead of just being sent across to you. You can let them build up for days if you want, they'll just be sitting in the queue.

Because OnGuard does not allow you to talk back and forth to these queues, the queues are designed to be one-way. When they receive events, there's a one-way traffic. When a cardholder's properties, badges, or accounts are changed, DataConduIT Queue will look up all cardholder information and send it across the queue as XML. Photos are optional because they are large. All of this information is sent as XML. This way you can store the data you want without having to look up anything extra. Any data in the XML packet can be ignored if you wish.

Schema Overview

The schema shows the structure of the OnGuard events going across the queue and the format for making requests to the OnGuard software. The schema is available through the OnGuard user interface by clicking the [Generate Schema] button on the DataConduIT Message Queues form. This will detect all UDF drop-down values as well as custom cardholder forms. The schema is saved as a separate file with a .XSD extension. The schema is not sent across the queue and there is no way to request it other than to generate it in System Administration. If FormsDesigner changes are made, DataConduIT and DataConduIT Message Queue must be restarted to pick up the new layout, and the schema must be regenerated.

In order to have the schema dynamically generated via System Administration, you must be logged in using single sign-on. This is required for WMI, which is needed to build the schema.

When you're generating the schema, you have to make sure you're generating the schema on a machine that DataConduIT is running on. This is because to generate the schema, OnGuard needs to communicate with DataConduIT to get database information. If it's not, an error will be generated that tells you to check the log.

When the DataConduIT Message Queue service starts up, it will also generate its own copy of the schema. If you receive a message with a format or value that is not in the schema, you must generate a new one to use. If the DataConduIT Message Queue service cannot validate the DataConduIT request that it was sent with the XML schema that it has, an error will occur. You may need to restart both services so that a new schema is generated.

How DataConduit Message Queue Handles Database Layout Changes

It is strongly recommended that all necessary changes to FormsDesigner be made before using DataConduit or DataConduit Message Queue. After using FormsDesigner to make changes to the database, you must restart the LS DataConduit Service, LS Linkage Server, and LS DataConduit Message Queue Server services in order to pick up the new database layouts. You must also regenerate the schema.

Updating the Database with Queue Changes

Consider the scenario where you have five queues configured and the DataConduit Message Queue service is running. If you decide that you don't want badge events or notifications on one queue, you can modify the queue and tell it not to send badge events. DataConduit Message Queue will check and see if anything has changed about these queues, and will pickup the change. How often does DataConduit Message Queue check the queues?

DataConduit Message Queue periodically looks at the database to see if anything has changed. This period of time is specified in the value "DATABASEUPDATE" in HKEY_LOCAL_MACHINE\Software\Lenel\OnGuard\DataConduitQueue registry key.

Note: The DataConduitQueue key does not exist by default - you will have to create it.

By default, the DATABASEUPDATE value is not specified in the registry, and the update value is one minute. If the DATABASEUPDATE value is specified in the registry, the time specified (in seconds) will be used for the update value.

Error Logging

DataConduit Message Queue errors are written to the **DataConduitQueue.log** file in the OnGuard logs directory (located in **C:\Program Files\OnGuard\logs** by default). Here are a few of the most common situations where an error message would be written to the **DataConduitQueue.log** file.

If connection to the queue/queue manager is lost. Constant connection to the queue/queue manager is critical. If at any time the connection is lost, the LS DataConduit Message Queue Server service will try to reconnect. Incoming queues try to reconnect every 15 seconds and outgoing queues will try to reconnect every time a message is about to be sent.

The service will not write errors to the log if it is shut down. An error will be written only when the queue manager or queue is unreachable or if the queue was not configured correctly in System Administration.

Can't log in.

Can't connect to DataConduit.

Can't send request to DataConduit for validation reasons.

A request that does not match the schema. If a message with an invalid schema is received, an error will be written to the **DataConduitQueue.log** file.

If the **DataConduITQueue.log** does not provide enough information or if you are directed to do so, refer to the **DataConduIT.log** file. For more information, refer to [Error Logging](#) on page 36.

Installing DataConduIT Message Queue

DataConduIT Message Queue is installed as part of a standard server installation. Note that DataConduIT must be on the same machine that the Linkage Server is running on if you want to receive events. Therefore, DataConduIT Message Queue is required to be on the same machine as the Linkage Server and DataConduIT is configured to run on; it cannot be run on a separate machine.

DataConduIT Message Queue runs as a Windows service under the same account that single sign-on is enabled for. DataConduIT Message Queue is installed with the login as LocalSystem, as all the other OnGuard services are, but it will not work under the LocalSystem account. You must change the account that DataConduIT Message Queue runs under by following [Change the Account the DataConduIT Message Service is Run With](#) on page 45.

The Linkage Server does not need to be running if you are using incoming queues. The Linkage Server is only used to receive events. Since you can only set up one instance of the Linkage Server per system, you can only setup one instance of DataConduIT Message Queue per system. If you set up DataConduIT Message Queue on machine A and you want to get events on machine B, all you have to do is setup your queue software to have client tools to B. You can still receive events at any machine that you want; it's where your queues reside that is key. Your queue doesn't have to physically reside on the machine you're setting up, but you must have the IBM WebSphere client tools on the machine you're setting up the queue on.

License for DataConduIT Message Queue

DataConduIT and DataConduIT Message Queue are separately licensed features. You can have a license for only DataConduIT, or a license for DataConduIT and DataConduIT Message Queue.

The DataConduIT Message Queue license is count-based; you are licensed to use a certain number of queues. Every time you click [Add] on the DataConduIT Message Queues form in System Administration, this counts as another queue.

The number of queues you are licensed to use is displayed in the "Maximum Number of Message Queues" setting in the General section of the license. To view this setting, open License Administration. For more information, refer to "Using OnGuard in the Supported Operating Systems" in the Installation Guide.

Setting Permissions to Use DataConduIT

Configure the System Options

1. In the *System Administration* application, select *Administration > System Options > General System Options* form.
2. In the **Linkage Server host** field, ensure that the correct host computer that runs the Linkage Server is identified. If not, click **Browse** and select the correct host computer.
3. Select the **Generate software events** checkbox.

Note: Selecting this checkbox ensures that events are generated for DataConduIT Message Queue to package up into XML and send over the queue.

4. When done, click **OK** at the bottom of the *General System Options* form to save the changes made.

Configure the User Permissions

In order for a user to use the DataConduIT Message Queue, the user must have the **DataConduIT message queues** and **DataConduIT Service** user permissions.

1. In the *System Administration* application, select *Administration > Users > System Permission Groups* form.
2. From the **Permission Group** list, select the desired user.
3. From the **Listing** window, scroll to the **Software options** entry and expand the entry.
4. Ensure that the selected user has at least **View/Access** permission for **DataConduIT message queues**. If not, click the key icon to enable **View/Access** permission for the selected user.

Note: If desired, the selected user can also have **Add**, **Modify**, and/or **Delete** permissions for **DataConduIT message queues**.

5. Scroll to the **Software Options - Applications** entry and expand it.
6. Ensure that the selected user has at least **View/Access** permission for **DataConduIT Service**. If not, click the key icon to enable **View/Access** permission for the selected user.
7. When done, click **OK** at the bottom of the *System Permission Groups* form to save the changes made.

All functionality available through DataConduIT is controlled by the same permissions that already used to manage data in ID CredentialCenter. For example, to add a cardholder through DataConduIT, the user adding the cardholder must have the **Add Cardholder** user permission. To view readers through DataConduIT, the user viewing readers must have the **View Reader** user permission.

Existing permissions also control who can receive hardware and software events. For hardware events, the client should only be able to receive events on its segment. For software events, the client should only be able to receive events for objects that the client can view on its segment. This means that the object must be in one of the client's segments, and the client must have permission to view the object and all of its properties (for objects with view/access permissions).

Configuring DataConduIT Message Queue

Configure the DataConduIT Message Queue

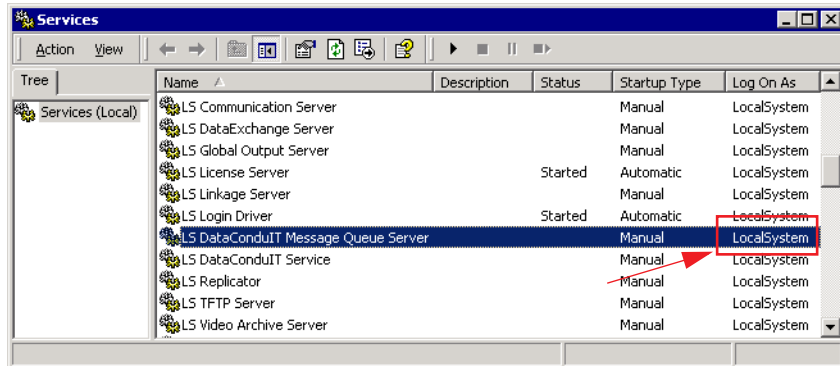
1. Install IBM WebSphere MQ software.
2. In IBM WebSphere MQ, configure the queues that you want for use with the OnGuard software.
3. Set up DataConduIT as you normally would. This includes:
 - a. Set up the Linkage Server.
 - b. Check the software events.
 - c. Select the **Generate software events** checkbox on the General System Options form in the System Options folder.
 - d. Set up single sign-on for DataConduIT.

4. Change the account the DataConduIT Message Queue Server service is run with.

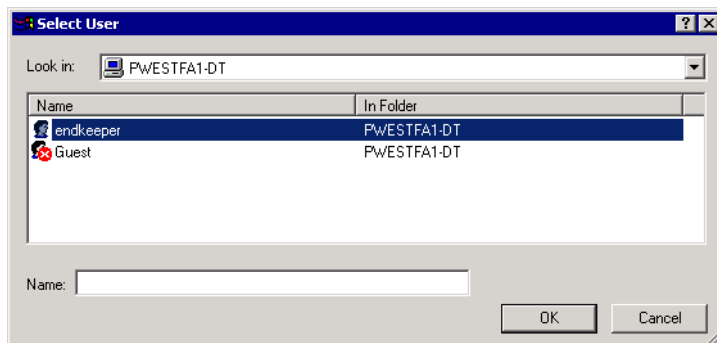
Change the Account the DataConduIT Message Service is Run With

DataConduIT Message Queue Server is installed with the login as LocalSystem, as all the other services do. However, it will not work under the LocalSystem account. You must change the LS DataConduIT Message Queue Server service to logon under the account that single sign-on is enabled for. To do this:

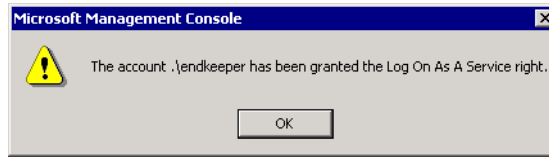
1. In Windows, open the **Control Panel**.
For more information, refer to “Using OnGuard in the Supported Operating Systems” in the Installation Guide.
2. Double-click “Administrative Tools”.
3. Double-click “Services”.
4. Select the “LS DataConduIT Message Queue Server” service, as shown.



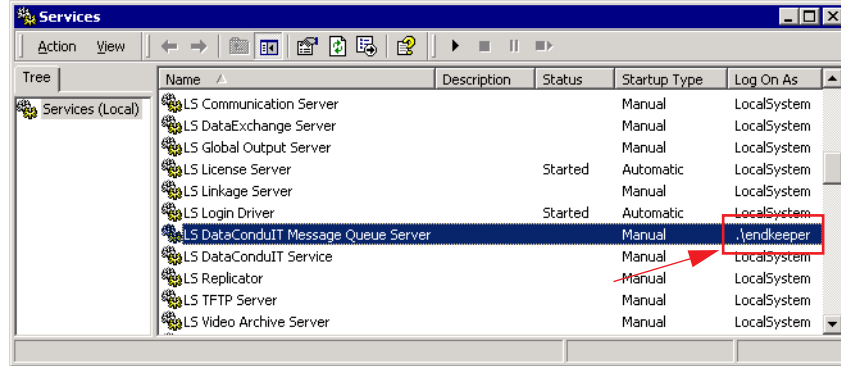
5. Right-click on the “LS DataConduIT Message Queue Server” service and select **Properties** from the right-click menu.
6. Click the Log On tab.
 - a. Select the **This account** radio button.
 - b. Click [Browse...].
 - c. In the Select User window, select the user account that single sign-on is enabled for, then click [OK].



- d. In the **Password** field, type the Windows password for the user account that you selected.
- e. In the **Confirm Password** field, retype the password.
- f. Click [OK]. A confirmation message similar to the following will be displayed:



- g. Click [OK].
- h. In the Services window, the user account you selected will be displayed in the Log On As column, as shown.



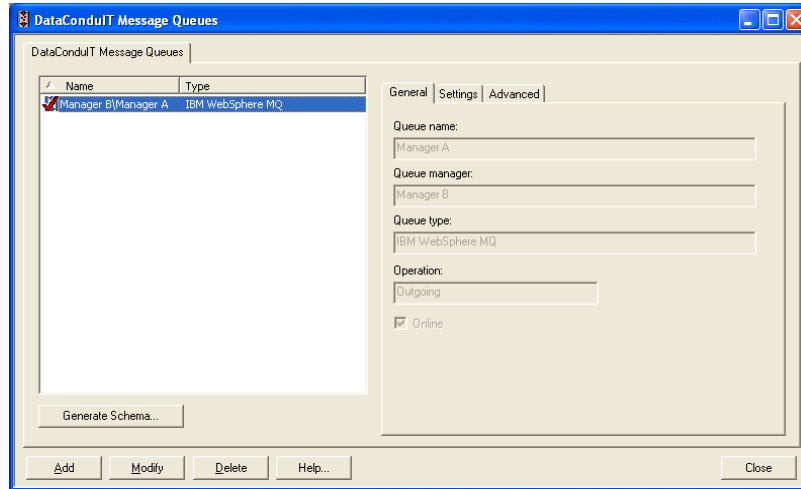
The DataConduIT Message Queues folder is found in System Administration and ID CredentialCenter, and contains forms with which you can:

- Add, modify, or delete DataConduIT message queues.
- Generate a schema for the user to reference.
- Configure whether photo and signature information is included in messages.
- Configure when messages are sent.
- Add, modify, or delete a custom object event WMI query, custom access and security event WMI query.

The DataConduIT Message Queues folder contains one form: the DataConduIT Message Queues form. The DataConduIT Message Queues form contains three sub-tabs: General, Settings, and Advanced.

This folder is displayed by selecting *DataConduIT Message Queues* from the *Administration* menu in System Administration or ID CredentialCenter.

DataConduIT Message Queues Form (General Sub-tab)



Listing window

Lists currently defined DataConduIT message queues. Each entry contains the queue's name and type.

Generate Schema

Generates a schema for you to reference. If clicked, the Save As window is displayed, and you must select where to save the schema.

After any changes to the database have been made using FormsDesigner, you must regenerate the schema so that the updated database is reflected in the schema file.

DataConduIT uses the Windows account of the person who is logged on to the machine at the time of schema creation. Because of this, it is probably more preferable for a system administrator to handle all schema generation.

Add

Click this button to add a DataConduIT message queue.

Modify

Click this button to change a selected DataConduIT message queue.

Delete

Click this button to delete a selected DataConduIT message queue.

Help

Displays online help for this form.

Close

Closes the DataConduIT Message Queues folder.

Queue name

Enter the queue's name. This field is case-sensitive.

Queue/SNMP manager

This field does not pertain to Microsoft Message Queues. If adding an IBM WebSphere MQ queue, enter the queue manager's name. This field is case-sensitive. If adding an SNMP Trap

Messages queue, enter the SNMP manager's IP address. Depending on the network configuration, a fully qualified NetBios name may be required.

Queue type

OnGuard supports the following types of queues: IBM WebSphere MQ, Microsoft Message Queue, and SNMP Trap Messages. The queue type is selected when a queue is added, and it cannot be modified after the queue has been added.

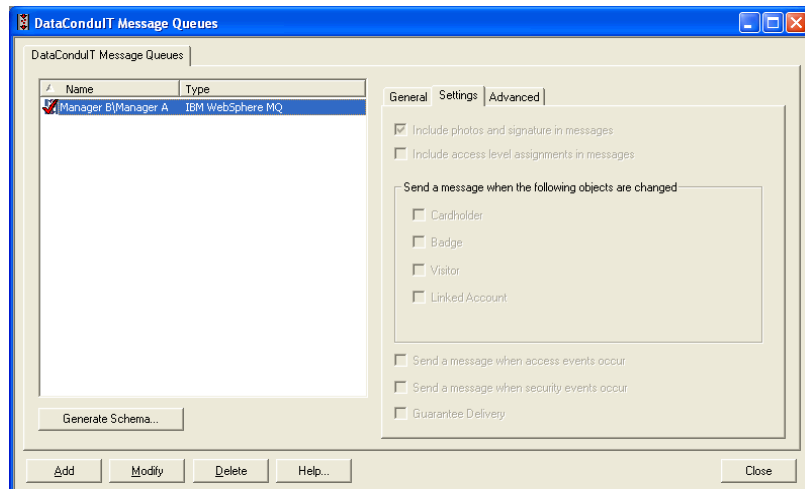
Operation

The IBM WebSphere MQ queue type supports two operations: incoming and outgoing. A queue is designated as either incoming or outgoing when it is added. The SNMP Trap Messages queue type only supports outgoing queues. The operation cannot be modified after a queue has been added.

Online

Shows whether the queue is online or offline. While checked the queue is online and will function normally. Unchecked makes the queue become offline. Being offline means no events are sent or received from the queue.

DataConduIT Message Queues Form (Settings Sub-tab)



Note: This sub-tab is only displayed for outgoing queues.

Include photos and signature in messages

Specifies whether photos, signatures, and fingerprints are included in messages. If this option is selected, the size of the messages sent is much larger.

Include access level assignments in messages

Check this box to include access level assignments in the outgoing messages.

Cardholder

If selected, a message will be sent whenever a cardholder record is added, modified, or deleted.

Badge

If selected, a message will be sent whenever a badge record is added, modified, or deleted.

Visitor

If selected, a message will be sent whenever a visitor record is added, modified, or deleted.

Linked Account

If selected, a message will be sent whenever a linked account record is added, modified, or deleted.

Send a message when access events occur

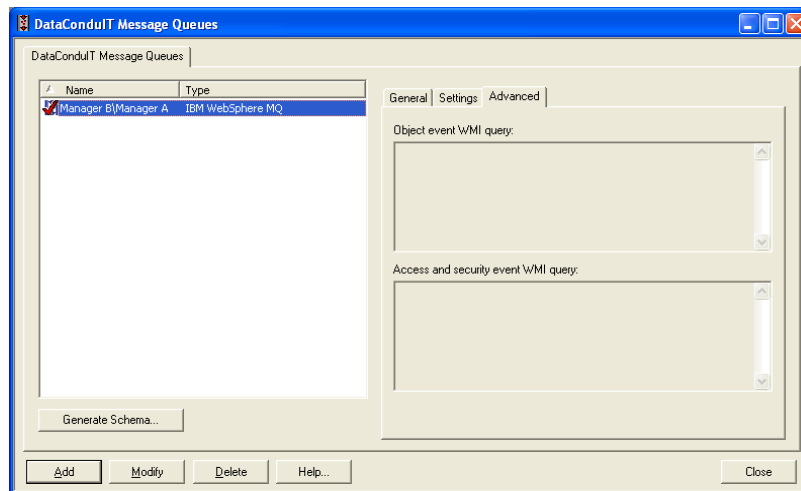
If selected, a message will be sent every time an access event occurs. Two examples of access events are access granted and access denied events.

Send a message when security events occur

If selected, a message will be sent every time a security event occurs. Two examples of security events are door forced open and alarm restored events.

Guarantee Delivery

Check this box to guarantee delivery of hardware events. This works by first sending the events to a table where the DataConduITQueue will then retrieve them. The guarantee is assured because the table is used as a preliminary queue and the events are not deleted until picked up by the DataConduITQueue. The DataConduITQueue will not mark the event as processed until it is written on the designated message queue. There is a mathematically small possibility that you could receive a duplicate event, but the chances are negligible.

DataConduIT Message Queues Form (Advanced Sub-tab)

Note: This sub-tab is only displayed for outgoing queues.

Object event WMI query

You can type an object event WMI query in directly. Objects include cardholders, linked accounts, badges, and visitors.

Access and security event WMI query

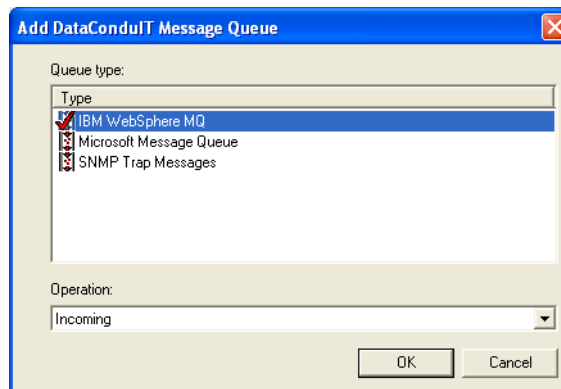
You can type an access and security event WMI query in directly. Access events are events such as access granted and access denied. Security events are events such as door forced open and alarm restored.

DataConduIT Message Queues Form Procedures

Use the following procedures on this form.

Add DataConduIT Message Queue

1. From the *Administration* menu, select *DataConduIT Message Queues*.
2. On the DataConduIT Message Queues form, click the [Add] button.
3. The Add DataConduIT Message Queue window opens.
 - a. Select the queue **Type**.
 - b. Select the queue **Operation**. The operation cannot be modified after a queue has been added.
 - The Microsoft Message Queue and IBM WebSphere MQ queue types support two operations: incoming and outgoing.
 - The SNMP Trap Messages queue type supports only the outgoing operation.



- c. Click [OK].
4. On the General sub-tab:
 - a. In the **Queue name** field, type the name of the queue. The name is case-sensitive. For IBM WebSphere MQ queues, this name must be exactly the same name that you used when setting up the queue in the IBM WebSphere MQ software.
 - b. In the **Queue manager or SNMP manager** field, enter the manager's name. If adding an IBM WebSphere MQ queue, enter the queue manager's name. If adding an SNMP Trap Messages queue, enter the SNMP manager's IP address. Depending on the network configuration, a fully qualified NetBios name may be required. If adding a Microsoft Message Queue this field is not present.
 - c. Note that the **Queue type** and **Operation** that you selected are displayed, but cannot be modified.
5. If you added an incoming queue, click [OK] and the queue will be added. If you added an outgoing queue, continue on to step 6.
6. On the Settings sub-tab:

- a. If you wish to have photo, signature, and fingerprint information sent in messages, select the **Include photos and signature in messages** check box.

Note: Including photo information in the messages makes the size of the message sent much larger.

- b. Select whether a message will be sent when cardholder, badge, visitor, and linked accounts are added, modified, or deleted.
 - c. If you wish to have a message sent when an access event occurs, select the **Send a message when access events occur** check box.
 - d. If you wish to have a message sent when a security event occurs, select the **Send a message when security events occur** check box.
7. Using the Advanced sub-tab is optional and for advanced users. On the Advanced sub-tab you may:
 - a. Type an object event WMI query directly into the **Object event WMI query** textbox.
 - b. Type an access and security event WMI query directly into the **Access and security event WMI query** textbox.
 8. Click the [OK] button.

Note: If you configured an SNMP Trap Messages queue, load the **lenel.mib** file into the SNMP Manager so that it knows how to handle and display the variables it receives. The Lenel MIB file is located in the **Support Center/SNMP** folder on the Supplemental Materials disc.

Modify a DataConduIT Message Queue

1. From the *Administration* menu, select *DataConduIT Message Queues*.
2. In the listing window of the DataConduIT Message Queues form, select the queue record you wish to modify.
3. Click the [Modify] button.
4. Make the changes you want to the fields. Changes can be made on any sub-tab.
5. Click the [OK] button to save the changes, or the [Cancel] button to revert to the previously saved values.

Delete a DataConduIT Message Queue

1. From the *Administration* menu, select *DataConduIT Message Queues*.
2. In the listing window of the DataConduIT Message Queues form, select the queue record you wish to delete.
3. Click the [Delete] button.
4. Click the [OK] button.
5. Click the [Yes] button to confirm the deletion.

DataConduIT is an advanced application integration service that allows real time, bidirectional integration between OnGuard and third party IT sources. DataConduIT allows System Administrators to develop scripts and/or applications that allow events in one domain (security or IT) to cause appropriate actions in the other.

DataConduIT Sources Folder

Note: In order to receive DataConduIT source events, add at least one online panel to the same monitor zone as the source.

The DataConduIT Sources folder is found in System Administration and allows System Administrators to add, modify and delete third-party DataConduIT Sources, Devices, and Sub-Devices. After third-party sources are added, users can send the incoming events to OnGuard via DataConduIT and view third party events in Alarm Monitoring.

To send an event to OnGuard via DataConduIT, System Administrators must:

- Define the incoming source in the DataConduIT Sources folder
- Use the `Lnl_IncomingEvent::SendIncomingEvent` method

Note: The DataConduIT method has four parameters: the source, description, device (optional), and subdevice (optional). The source of the DataConduIT method must match the source name on the DataConduIT Sources form. If the optional parameters are used, the device of the DataConduIT method must match the device name on the DataConduIT Devices form, and the subdevice must match the sub-device name on the DataConduIT Sub-Devices form.

- Have at least one panel (non-system DataConduIT Source) configured and marked online so that the Communications Server will work properly with DataConduIT Sources. The panel does not need to exist or actually be online in Alarm Monitoring, it simply needs to exist and show up in the System Status view. Once this is set up, events can be successfully received by Alarm Monitoring from DataConduIT Sources.

This folder is displayed by selecting *DataConduIT Sources* from the *Additional Hardware* menu, or by selecting the DataConduIT Sources toolbar button in System Administration or ID CredentialCenter.

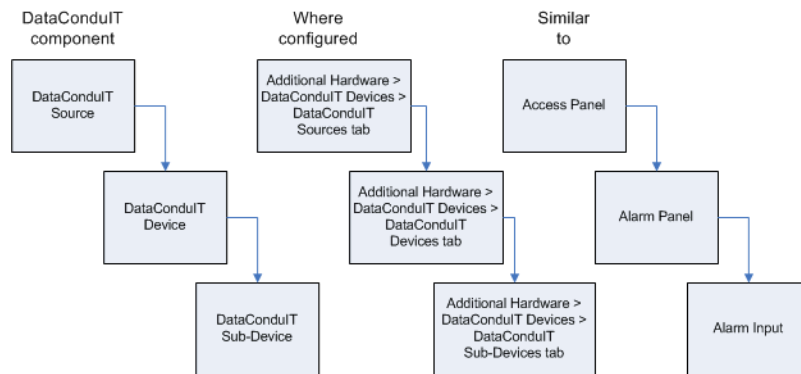
Toolbar Shortcut



DataConduIT Source Downstream Devices

A DataConduIT Source may have DataConduIT Device or DataConduIT Sub-Device downstream devices. A DataConduIT Device is a child of a DataConduIT Source, similar to how an alarm panel is a child of an access panel. A DataConduIT Sub-Device is a sub-child device of a DataConduIT Device, similar to how an alarm input is a sub-child of an alarm panel. The diagram that follows illustrates this hierarchy.

DataConduIT Downstream Device Hierarchy



DataConduIT Devices and DataConduIT Sub-Devices also display in Alarm Monitoring in the System Status Tree. For example, a DataConduIT Device named “Tivoli” with a DataConduIT Device named “Tivoli device” and a DataConduIT Sub-Device named “Tivoli sub-device” would display in Alarm Monitoring in the following manner:



Licenses Required

No additional license is required to use the DataConduIT Sources folder other than the “Maximum Number of DataConduIT Clients” license to use DataConduIT in general.

User Permissions Required

DataConduIT Service Permission

The permission required to use DataConduIT in general is the DataConduIT service user permission. This permission is located in Administration > Users > System Permission Groups tab > Software Options sub-tab in System Administration or ID CredentialCenter.

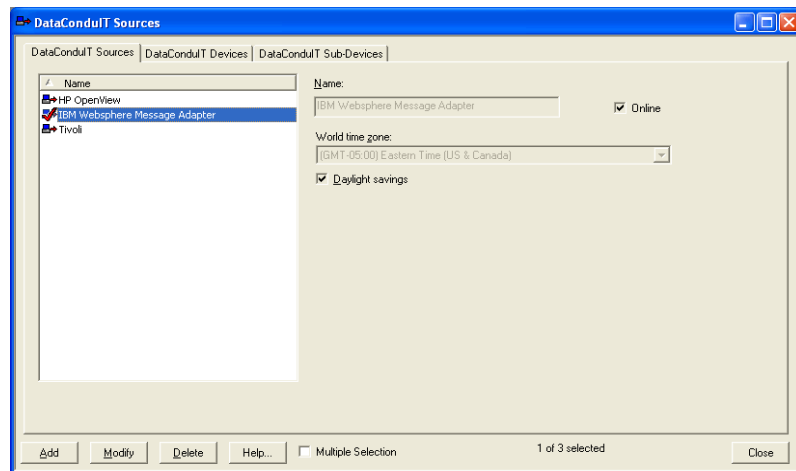
Add, Modify, and Delete DataConduIT Sources, Devices, and Sub-Devices

The add, modify, and/or delete DataConduIT Sources permissions determine what functions a user can perform on DataConduIT Sources, DataConduIT Devices, and DataConduIT Sub-Devices in the DataConduIT Sources folder. These permissions are located in Administration > Users > System Permission Groups tab > Additional Data Sources sub-tab in System Administration or ID CredentialCenter.

Trace DataConduIT Sources, Devices, and Sub-Devices

In addition, user permissions are required to trace DataConduIT Sources, DataConduIT Devices, and DataConduIT Sub-devices in Alarm Monitoring. These permissions are located in Administration > Users > Monitor Permission Groups tab > Monitor sub-tab in System Administration or ID CredentialCenter.

DataConduIT Sources Form



Listing window

Lists DataConduIT Source names.

Name

Identifies the name of the DataConduIT Source. This is a “friendly” name assigned to each DataConduIT Source to make it easy to identify.

Online

If selected, the DataConduIT Source is online and ready for use. To suspend the DataConduIT Source deselect this box.

World time zone

Select the world time zone for the selected access panel's geographical location. The selections in the drop-down list are listed sequentially, and each includes:

- The world time zone's clock time relative to Greenwich Mean Time. For example, (GMT+05:00) indicates that the clock time in the selected world time zone is 5 hours ahead of the clock time in Greenwich, England.
- The name of one or more countries or cities that are located in that world time zone.

Daylight savings

Select this check box if Daylight Savings Time is enforced in the selected access panel's geographical location.

Add

Click this button to add a DataConduIT Source.

Modify

Click this button to modify a DataConduIT Source.

Delete

Click this button to delete a DataConduIT Source.

Help

Click this button to display online help for this form.

Multiple Selection

If selected, more than one entry in the listing window can be selected simultaneously. The changes made on this form will apply to all selected DataConduIT Sources.

Close

Click this button to close the DataConduIT Sources folder.

DataConduIT Sources Form Procedures

Use the following procedures on this form.

Add a DataConduIT Source

1. From the *Additional Hardware* menu, select **DataConduIT Sources**. The DataConduIT Sources folder opens.
2. On the DataConduIT Sources tab, click [Add].
3. If segmentation is not enabled, skip this step. If segmentation is enabled:
 - a. The Segment Membership window opens. Select the segment that this DataConduIT Source will be assigned to.
 - b. Click [OK].
4. In the **Name** field, type a name for the DataConduIT Source.

5. Select whether the DataConduIT Source will be online.
6. Select the world time zone and daylight savings options as you see fit.
7. Click [OK].

IMPORTANT: In addition to having a DataConduIT Source configured, there must be at least one panel (non-system DataConduIT Source) configured and marked online so that the Communications Server will work properly with DataConduIT Sources. The panel does not need to exist or actually be online in Alarm Monitoring, it simply needs to exist and show up in the System Status view. Once this is set up, events can be successfully received by Alarm Monitoring from DataConduIT Sources.

Modify a DataConduIT Source

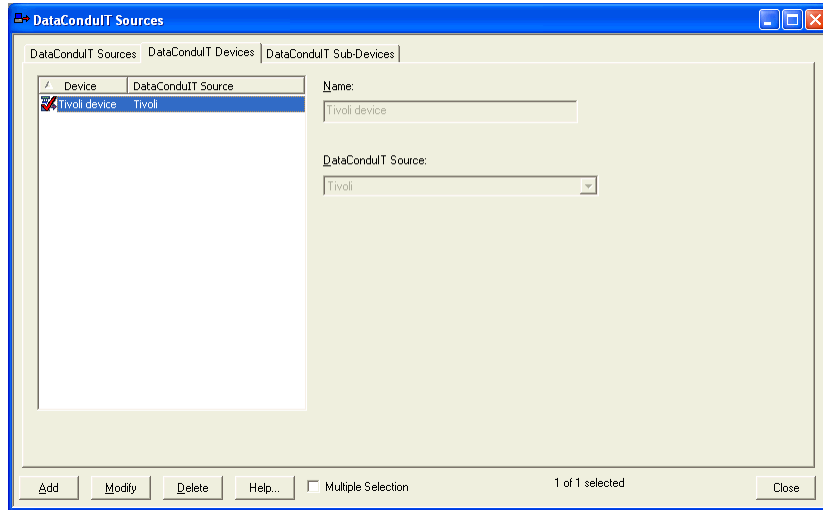
1. From the *Additional Hardware* menu, select *DataConduIT Sources*.
2. On the DataConduIT Sources tab, select the entry you want to modify from the listing window.
3. Click [Modify].
4. Make any changes.
5. Click [OK].
6. A prompt to confirm that you want to make the modification displays. Click [OK].

Delete a DataConduIT Source

To suspend a DataConduIT Source without deleting it, take it offline.

1. From the *Additional Hardware* menu, select *DataConduIT Sources*.
2. On the DataConduIT Sources tab, select the entry you want to delete from the listing window.
3. Click [Delete].
4. Click [OK].
5. A prompt to confirm that you want to make the deletion will be displayed. Click [OK].

DataConduIT Devices Form



Listing window

Lists DataConduIT Device names.

Name

Identifies the name of the DataConduIT Device. This is a “friendly” name assigned to each DataConduIT Device to make it easy to identify.

DataConduIT Source

Select the DataConduIT Source that is the parent of the child device being configured. DataConduIT Sources are configured on the DataConduIT Sources tab (Additional Hardware > DataConduIT Sources > DataConduIT Sources tab).

Add

Click this button to add a DataConduIT Device.

Modify

Click this button to modify a DataConduIT Device.

Delete

Click this button to delete a DataConduIT Device.

Help

Click this button to display online help for this form.

Multiple Selection

If selected, more than one entry in the listing window can be selected simultaneously. The changes made on this form will apply to all selected DataConduIT Devices.

Close

Click this button to close the DataConduIT Sources folder.

DataConduIT Devices Form Procedures

Use the following procedures on this form.

Add a DataConduIT Device

Prerequisite: Before a DataConduIT Device can be configured, its parent DataConduIT Source must first be configured.

Note: If segmentation is enabled, the segment of the DataConduIT Source will be used as the segment for the DataConduIT Device.

1. From the *Additional Hardware* menu, select *DataConduIT Sources*. The DataConduIT Sources folder opens.
2. Click the DataConduIT Devices tab.
3. Click [Add].
4. In the **Name** field, type a name for the DataConduIT Device.
5. Select the DataConduIT Source that is the parent of the DataConduIT Device.

Note: The DataConduIT Source must be configured on the DataConduIT Sources tab.

6. Click [OK].

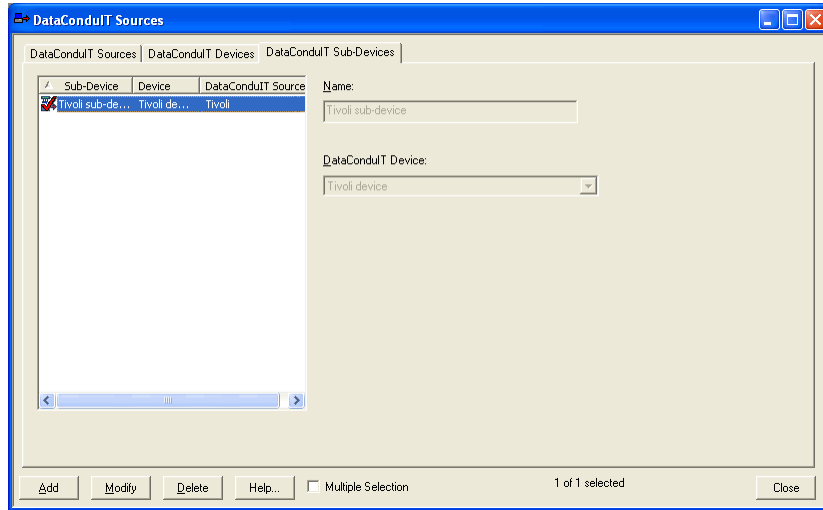
Modify a DataConduIT Device

1. From the *Additional Hardware* menu, select *DataConduIT Sources*.
2. Click the DataConduIT Devices tab.
3. Select the entry you want to modify from the listing window.
4. Click [Modify].
5. Make any changes.
6. Click [OK].
7. A prompt to confirm that you want to make the modification displays. Click [OK].

Delete a DataConduIT Device

1. From the *Additional Hardware* menu, select *DataConduIT Sources*.
2. Click the DataConduIT Devices tab.
3. Select the entry you want to delete from the listing window.
4. Click [Delete].
5. Click [OK].
6. A prompt to confirm that you want to make the deletion will be displayed. Click [OK].

DataConduIT Sub-Devices Form



Listing window

Lists DataConduIT Sub-Device names, along with the parent DataConduIT Device and DataConduIT Source.

Name

Identifies the name of the DataConduIT Sub-Device. This is a “friendly” name assigned to each DataConduIT Sub-Device to make it easy to identify.

DataConduIT Device

Select the DataConduIT Device that is the parent of the child Sub-Device being configured. DataConduIT Devices are configured on the DataConduIT Devices tab (*Additional Hardware* > *DataConduIT Sources* > DataConduIT Devices tab).

Add

Click this button to add a DataConduIT Sub-Device.

Modify

Click this button to modify a DataConduIT Sub-Device.

Delete

Click this button to delete a DataConduIT Sub-Device.

Help

Click this button to display online help for this form.

Multiple Selection

If selected, more than one entry in the listing window can be selected simultaneously. The changes made on this form will apply to all selected DataConduIT Sub-Devices.

Close

Click this button to close the DataConduIT Sources folder.

DataConduIT Sub-Devices Form Procedures

Use the following procedures on this form.

Add a DataConduIT Sub-Device

Prerequisite: Before a DataConduIT Sub-Device can be configured, its parent DataConduIT Source and DataConduIT Device must be configured.

Note: If segmentation is enabled, the segment of the DataConduIT Source will be used as the segment for the DataConduIT Sub-Device.

1. From the *Additional Hardware* menu, select *DataConduIT Sources*. The DataConduIT Sources folder opens.
2. Click the DataConduIT Sub-Devices tab.
3. Click [Add].
4. In the **Name** field, type a name for the DataConduIT Sub-Device.
5. Select the DataConduIT Device that is the parent of the DataConduIT Sub-Device.

Note: The DataConduIT Device must be configured on the DataConduIT Devices tab.

6. Click [OK].

Modify a DataConduIT Sub-Device

1. From the *Additional Hardware* menu, select *DataConduIT Sources*.
2. Click the DataConduIT Sub-Devices tab.
3. Select the entry you want to modify from the listing window.
4. Click [Modify].
5. Make any changes.
6. Click [OK].
7. A prompt to confirm that you want to make the modification displays. Click [OK].

Delete a DataConduIT Sub-Device

1. From the *Additional Hardware* menu, select *DataConduIT Sources*.
2. Click the DataConduIT Sub-Devices tab.
3. Select the entry you want to delete from the listing window.
4. Click [Delete].
5. Click [OK].
6. A prompt to confirm that you want to make the deletion will be displayed. Click [OK].

The OnGuard OPC Client is a solution for integrating OnGuard with existing third party OPC Servers. The OnGuard OPC Client is an OPC-Alarms and Events client that can connect to any OPC Alarms and Events server. The purpose of the OnGuard OPC Client is to allow OPC Servers to send event and alarm notifications to OnGuard using the OLE for Process Control (OPC) industry standard format.

The OnGuard OPC Client consists of an user interface component to configure OPC Connections and a service component that subscribes to specified OPC Servers to receive event and alarm notifications.

OPC Client Functions

The purpose of the OnGuard OPC Client is to:

- Provide real time communication with any compatible OPC source
- Monitor events and alarms shared by the OnGuard OPC Client and compatible OPC sources

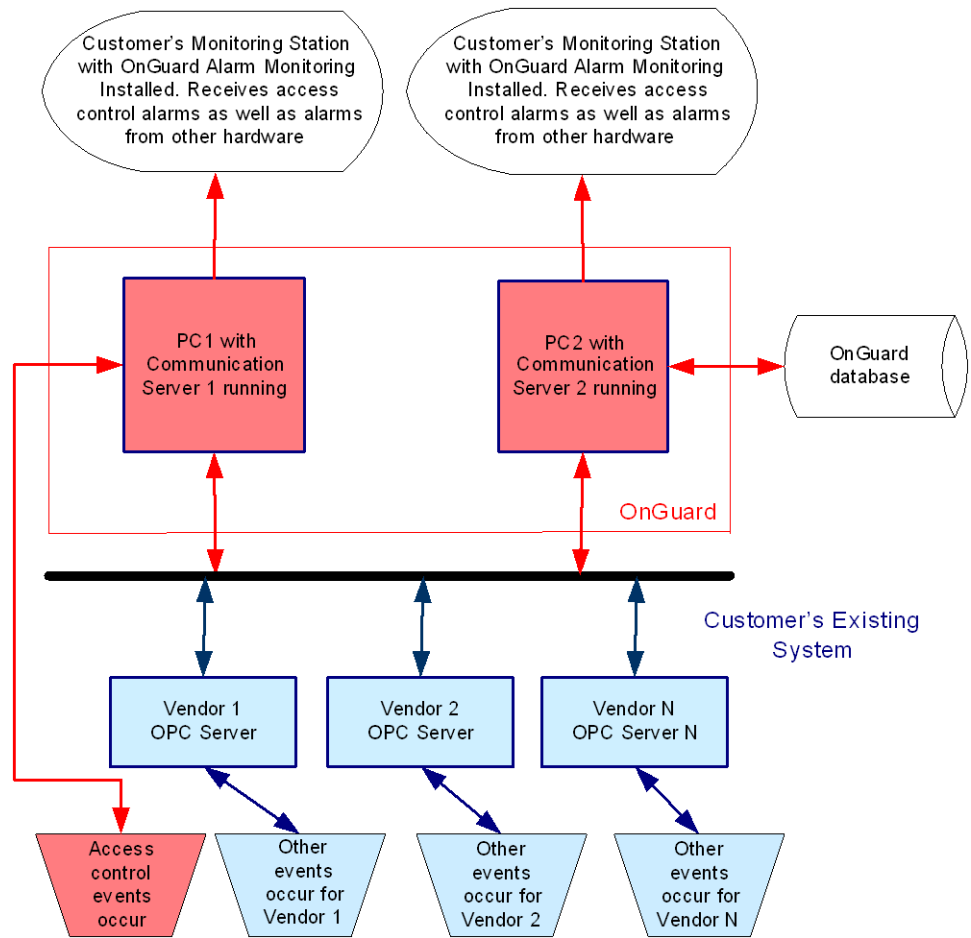
Note: Events and alarms sent by an OPC Server can be viewed, logged and even used to trigger specific actions.

OnGuard OPC Client Scenario

Let's look at a hypothetical customer in the airline industry. This customer has an existing central control room with several OPC compliant servers monitoring every flight and traveler information.

New high security access control card readers, cameras and motion detectors have been installed and the customer wants to integrate these devices with their existing systems and monitor access control alarms and events from the same control room.

How does the customer monitor the access control alarms and events using the existing OPC Servers?



By making OnGuard an OPC Client, the customer can use OnGuard to communicate directly with their existing OPC Servers. To make OnGuard an OPC Client the OPC support license must be purchased.

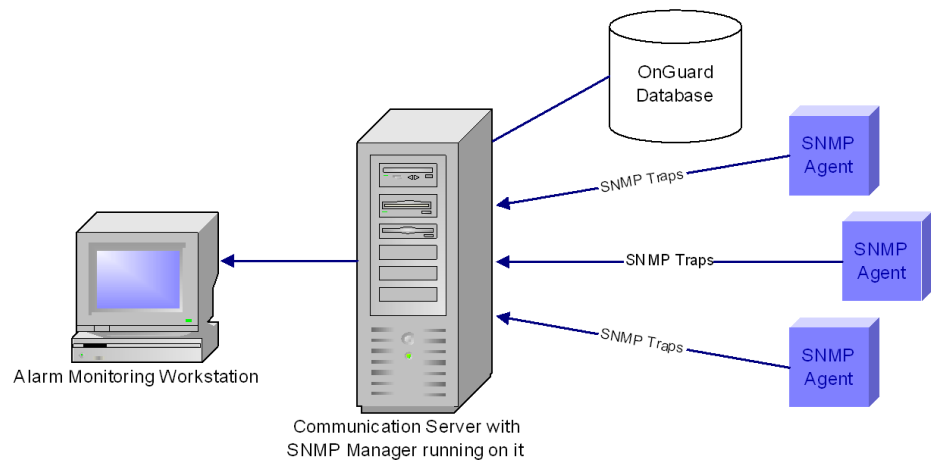
The OnGuard OPC Client receives and translates alarms and events from the OPC Server and outputs them in the Alarm Monitoring application along with the alarm and events received from the newly installed access control system.

In OnGuard 7.0 and later, the OPC client can receive and translate status events from the OPC server, and display the appropriate status icon for the OPC server in Alarm Monitoring. The messages that must be sent from the OPC server to indicate status are:

- LNL OPC PANEL ONLINE, which indicates online status, and
- LNL OPC PANEL OFFLINE, which indicates offline status.

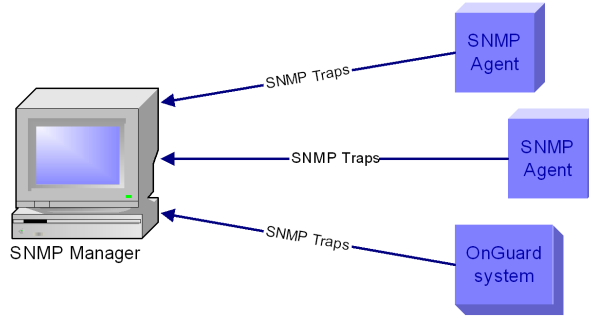
SNMP (Simple Network Management Protocol) is used primarily for managing and monitoring devices on a network. This is achieved through the use of get and set requests which access and modify variables on a given device, as well as SNMP traps which are used to notify Managers of changes as they occur. The device which is being managed or monitored is called the *Agent*. The application that is doing the managing or monitoring is called the *Manager*. You can think of a Manager as the coach of a team, and Agents as all the players on the team. The following diagram illustrates how OnGuard can be used as an SNMP Manager:

OnGuard as an SNMP Manager



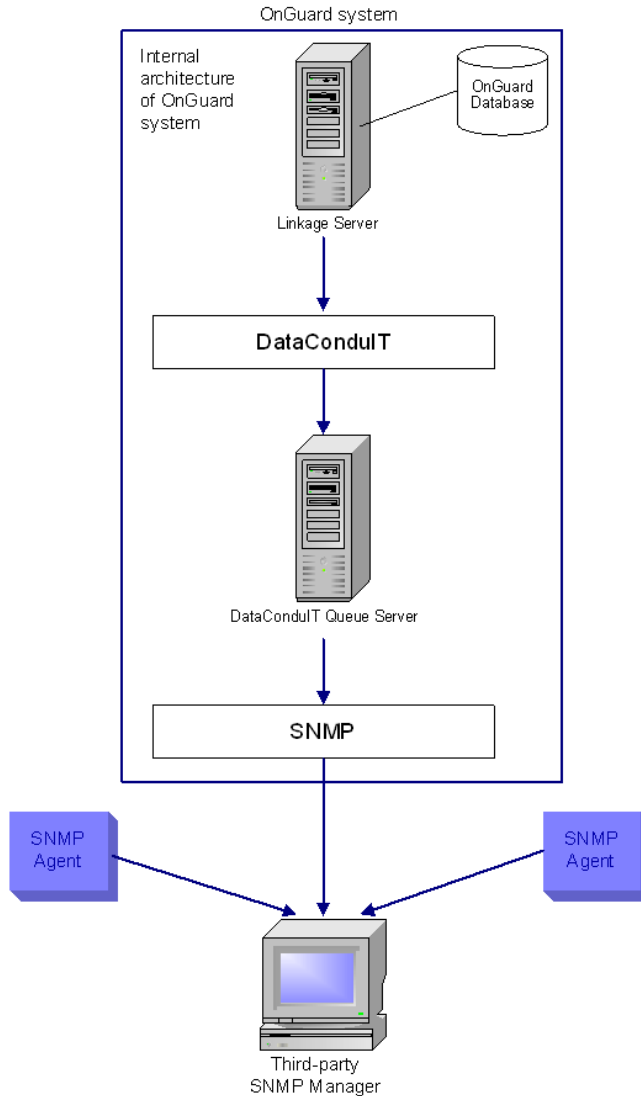
Agents generate *trap messages*, which are sent to a Manager to indicate that something has changed. Trap messages generally contain the system uptime, the trap type, and the enterprise number. OnGuard uses Enterprise specific trap messages to send alarms to SNMP Managers. OnGuard generates trap messages, but does not listen for messages from SNMP Managers. The following diagram illustrates how OnGuard can be used as an SNMP Agent:

OnGuard as an SNMP Agent



Configuring OnGuard as an SNMP Agent requires the use of DataConduIT and the DataConduIT Queue Server, as shown in the diagram that follows.

OnGuard as an SNMP Agent (Internal Architecture)



Why use SNMP with OnGuard? This depends on whether you are using OnGuard as an SNMP Manager or as an SNMP Agent.

OnGuard as an SNMP Manager

When OnGuard is used as an SNMP Manager:

- You can monitor hardware or software applications in OnGuard that you couldn't monitor before without a specific integration.
- If you already have OnGuard installed and are using a third-party application to monitor SNMP traps, you can now move that functionality over to OnGuard and monitor everything in a central location.
- By loading into OnGuard the MIB file for the SNMP Agents you are monitoring, you can customize how the information from the SNMP Agent is displayed in Alarm Monitoring
- Based on the information received and displayed in OnGuard, you can create custom alarm and Global I/O linkages for the trap, as well as take advantage of other existing OnGuard functionality.

To set up OnGuard to function as an SNMP Manager, you must configure an SNMP Manager on a workstation. This is done through System Administration. In addition to configuring the SNMP Manager, you can also load up third party MIB files into OnGuard, which will allow you to customize how SNMP Traps are handled and displayed in the OnGuard software. For more information, refer to the SNMP Managers Folder chapter in the System Administration User Guide.

OnGuard as an SNMP Agent

OnGuard hardware and software events can be reported as SNMP traps to third-party applications with SNMP trap support.

To configure OnGuard as an SNMP Agent, you must configure an SNMP Trap Message queue within the DataConduIT Message Queue configuration in System Administration. You can specify what events you want sent out through this queue (as SNMP Traps) and where you want them sent. For more information, refer to the DataConduIT Message Queues Folder chapter in the System Administration User Guide.

After setting this up, you must load the Lenel MIB file (located in the **SNMP** folder on the OnGuard Supplemental Materials disc) into your SNMP Manager application. For more information, refer to the SNMP Managers Folder chapter in the System Administration User Guide.

SNMP Manager Copyright Information

---- Part 1: CMU/UCD copyright notice: (BSD like) ----

Copyright 1989, 1991, 1992 by Carnegie Mellon University

Derivative Work - 1996, 1998-2000

Copyright 1996, 1998-2000 The Regents of the University of California

All Rights Reserved

Permission to use, copy, modify and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appears in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of CMU and The Regents of the University of California not be used in advertising or publicity pertaining to distribution of the software without specific written permission.

CMU AND THE REGENTS OF THE UNIVERSITY OF CALIFORNIA DISCLAIM ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL CMU OR THE REGENTS OF THE UNIVERSITY OF CALIFORNIA BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM THE LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

---- Part 2: Networks Associates Technology, Inc copyright notice (BSD) ----

Copyright (c) 2001-2002, Networks Associates Technology, Inc

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Networks Associates Technology, Inc nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---- Part 3: Cambridge Broadband Ltd. copyright notice (BSD) ----

Portions of this code are copyright (c) 2001-2002, Cambridge Broadband Ltd.

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- The name of Cambridge Broadband Ltd. may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Data Classes

Note: All class and property access is subject to OnGuard user permissions.

Lnl_AccessGroup

Description: An access group defined in the security system.

Abstract: No

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	SEGMENTID	Lnl_Segment.ID - ID of the segment the access level belongs to	View
string	NAME	Display name	View

Methods:

```
void AssignGroup([in]sint32 badgeKey);
```

Assigns all the access levels in the group to a specific badge.

Parameters:

badgeKey - Internal ID of the badge to assign the access levels to

LnI_AccessLevel

Description: An access level defined in the security system.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	SegmentID	LnI_Segment.ID - ID of the segment the access level belongs to	View/Edit
string	Name	Display name	View/Edit
boolean	HasCommandAuthority	Command authority is enabled for the access level	View
boolean	DownloadToIntelligentReaders	Level is download to Intelligent Readers	View
boolean	FirstCardUnlock	First Card Unlocks the reader	View

LnI_AccessLevelAssignment

Description: An access level assignment defined in the security system.

Abstract: No

Access: View/Add/Delete

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ACCESSLEVELID	LnI_AccessLevel.ID - ID of the access level. Key field.	View/Edit
sint32	BADGEKEY	LnI_Badge.BADGEKEY - BadgeKey of the badge. Key field.	View/Edit
datetime	ACTIVATE	Date when this assignment will become active.	View/Edit

Type	Name	Description	Access
datetime	DEACTIVATE	Date when this assignment will become inactive.	View/Edit

LnI_AccessLevelReaderAssignment

Description: An access level reader assignment defined in the security system.

Abstract: No

Access: View

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	AccessLevelID	Access level that the link belongs to	View
sint32	PanelID	LnI_Panel which is linked to this level	View
sint32	ReaderID	LnI_Reader ID which is linked to this level	View
sint32	TimezoneID	LnI_Timezone in which this level is active	View

LnI_Account

Description: A directory account belonging to a person in the security system.

Abstract: No

Access: View/Add /Delete

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View

Type	Name	Description	Access
string	ACCOUNTID	ID of the entry in the external directory. The ID is the value of the attribute specified in the Lnl_Directory.AccountIDAttr property. For example, for Microsoft directories, this property would contain the account's security identifier (SID)	View/Edit
sint32	DIRECTORYID	Internal ID of the directory to which this account belongs. See Lnl_Directory.ID.	View/Edit
sint32	PERSONID	Internal ID of the person who owns this account. See Lnl_Person.ID.	View/Edit

Lnl_AlarmDefinition

Description: Defines how the alarm that is received from the panel is displayed. Lnl_AlarmDefinition instances are queried by an end user in order to establish configuration details. This contrasts Lnl_Alarm instances, which come in with all security events that come through the Communication Server.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	Priority	Alarm priority (0-255)	View
string	Description	Parameter description	View
sint32	SegmentID	Segment ID	View
string	TextInstructionName	Text instruction name	View
string	TextInstructionData	Text instruction	View

Lnl_Area

Description: An APB area defined in the security system.

Abstract: No

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	AREATYPE	Type of APB area. Possible values: 0: Other 1: Unknown 2: Local Area 3: Global Area 4: Hazardous Location 5: Safe Location	View
string	NAME	Display name.	View

Methods:

```
void MoveBadge();
```

Moves a badge from one area into another.

```
sint32 MoveBadge([in] sint32 areaID, [in] sint64 badgeID, [in] sint32 panelID, [in] sint32 readerID,  
[in] sint32 segmentID, [in] datetime UTCTime);
```

Parameters:

- *areaID* - This is ID of the area to move the badge to.
- *badgeID* - This is the badge ID of the badge you want to move.
- *panelID* - This is the ID of the panel of the reader responsible for moving the badge to the new area.
- *readerID* - This is the ID of the reader responsible for moving the badge.
- *segmentID* - This is the segment associated with the *panelID*, *readerID*.
- **UTCTime** - The time when the badge was moved to the area.

Lnl_AuthenticationMode

Description: Authentication modes for pivCLASS authenticated readers. Authentication modes specify the authentication mechanism used by the reader to authenticate a cardholder. These modes are configured as assurance profiles in the pivCLASS Validation Server. Use the ID of a retrieved authentication mode when setting reader modes with the Lnl_Reader associated class. For more information, refer to [Lnl_Reader](#) on page 121.

Abstract: No

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID	View
string	Name	Name of the authentication mode	View

Lnl_Badge

Description: A badge in the security system.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	BADGEKEY	Internal database ID. Key field.	View
datetime	ACTIVATE	Badge activate date	View/Edit
boolean	APBEXEMPT	Whether the badge is APB exempt	View/Edit
datetime	DEACTIVATE	Badge deactivate date	View/Edit
sint32	EMBOSSSED	Embossed	View/Edit
boolean	EXTEND_STRIKE_HELD	Use extended strike/held times	View/Edit
string	EXTENDED_ID	Extended ID	View/Edit
sint64	ID	Badge ID	View/Edit
sint32	ISSUECODE	Issue code	View/Edit
datetime	LASTCHANGED	Badge last changed	View
datetime	LASTPRINT	Badge last printed	View
sint32	PERSONID	Internal ID of the person who owns this badge. See Lnl_Person.ID.	View/Edit
string	PIN	PIN code	View/Edit
sint32	PRINTS	Number of times badge has been printed	View

Type	Name	Description	Access
sint32	STATUS	Badge status. "Active" is 1. Other values are user-defined	View/Edit
sint32	TYPE	Badge type ID	View/Edit
sint32	USELIMIT	Use limit	View/Edit

Methods:

- void AddBadge([in] object *BadgeIn*, [out] object *BadgeOut*);
Adds badge to the system.
Parameters:
 - *BadgeIn* - The badge to be added to the system.
 - *BadgeOut* - The badge that was just added to the system with the new badge ID.
- void AssignAccessLevel([in] Uint32[] *LevelIn*);
Assigns the access level(s) of a badge.
Parameters:
 - *LevelIn* - Array that includes all the access level IDs the badge needs to be assigned with.

Lnl_BadgeFIPS201

Description: Holds the data imported from FIPS 201 credentials.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	BADGEKEY	Internal database ID of the associated badge record.	View/Edit
unit8[25]	FASCN	Federal Agency Smart Credential Number.	View/Edit
unit8[32]	TWICPrivacyKey	TWIC Privacy Key. The key used to encrypt/decrypt the fingerprints on TWICs.	View/Edit
sint32	TPKAlgorithmId	TWIC Privacy Key algorithm identifier. The algorithm used for encrypting/decrypting the fingerprints on TWICs. Paired with the TWIC Privacy Key.	View/Edit

Type	Name	Description	Access
unit8[16]	GUID	Cardholder's globally unique identifier.	View/Edit
sint32	CredentialType	The type of FIP 201 credential. 0 = Unknown 1 = PIV 2 = TWIC 3 = CAC with PIV Endpoint or Next Generation (NG) applet 4 = CAC without PIV applet 5 = PIV-I or CIV	View/Edit

Lnl_BadgeLastLocation

Description: Defines at what reader the badge was presented last.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint64	BadgeID	Badge ID	View
sint32	AccessFlag	Shows whether the access was granted	View
sint32	PanelID	Panel ID where access event occurred	View
sint32	ReaderID	Reader ID at which access occurred	View
datetime	EventTime	Time at which access occurred	View
sint32	EventID	ID of the event associated with the access.	View
sint32	EventType	Type of the event associate with access	View
sint32	PersonID	Lnl_Person for which access occurred	View
sint32	IsFromReplication	Shows whether badge last location came over for other region in the system.	View
sint32	DatabaseID	Database ID in an Enterprise system that identifies the reader to which the badge was presented.	View

Lnl_BadgeProperties

Description: Additional properties for the badge.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	BADGEKEY	Internal database ID of the associated Badge record.	View/Edit
sint32	CardInterface	Defines the contact or contactless interface of the badge. 0 = contact interface 1 = contactless interface	View/Edit
sint32	CardTechnology	Defines the technology of the card. 0 = contact 1 = iCLASS 2 = MiFare 3 = DESFire 4 = Proximity	View/Edit
string	SerialNumber	The serial number of the card. This may be different for each card interface.	View/Edit
string	DeviceType	The device type of the badge specific to the ActivIdentity CMS 3.8 integration.	View/Edit
string	ATR	The Answer to Reset of the badge specific to the ActivIdentity CMS integration.	View/Edit
boolean	IsRegisteredWithActivIdentity	Determines whether or not this badge is registered for logical access with ActivIdentity CMS. A badge is registered if it has been bound or issued to a user.	View/Edit
sint32	IssuingCmsID	If the badge is registered with CMS, then this specifies the ID of the Card Management System that issued the badge. This ID can be found in the ActivIdentity CMS server configuration screen in System Administration.	View/Edit

LnI_BadgeType

Description: A badge type in the security system.

Abstract: No

Access: View only

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	NAME	Name of the badgetype	View
sint32	BADGECLASS	Class of the badgetype Possible values: 1: Standard 2: Temporary 3: Visitor 4: Guest	View

LnI_Camera

IMPORTANT: The CameraType property for LnI_Camera should not be used. Instead use the CameraTypeName property.

Description: A camera defined in the system.

Abstract: No

Access: View

Superclass: LnI_Element

Platforms: OnGuard

Methods:

void GetHardwareStatus([out] uint32 Status): retrieves the camera status (0-offline, 1-online)

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	PanelID	Lenel NVR ID. See LnI_Panel.ID.	View
string	Name	Camera Name	View
sint32	CameraType	Camera Type	View
string	CameraTypeName	Camera Type Name	View
sint32	Channel	Lenel NVR Channel	View

Type	Name	Description	Access
string	VideoStandard	Video Standard (Ex.: NTSC)	View
sint32	IPAddress	IP address of the camera	View
sint32	Port	Port of the camera	View
sint32	HorizontalResolution	Horizontal resolution	View
sint32	VerticalResolution	Vertical Resolution	View
sint32	MotionBitRate	Motion Bit Rate	View
sint32	NonMotionBitRate	Non-motion Bit Rate	View
sint32	FrameRate	Frame rate	View
string	Workstation	Workstation of the host Lenel NVR	View

Lnl_CameraGroup

Description: Camera group definition.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Group name	View
sint32	SegmentID	Segment ID	View

Lnl_CameraGroupCameraLink

Description: An association between a camera and camera group.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	CameraGroupID	Camera group for this link. Lnl_CameraGroup.ID. Key field.	View
sint32	PanelID	Panel ID for the camera. See Lnl_Camera.PanelID. Key field.	View
sint32	CameraID	Camera ID. Key field. See Lnl_Camera.ID.	View

Lnl_Cardholder

Description: A cardholder in the security system.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: Lnl_Person

Platforms: OnGuard

Properties: The class has all the properties of the Lnl_Person class, plus any custom fields defined by the end user. In addition, the class has the following properties:

Type	Name	Description	Access
boolean	ALLOWEDVISITORS	Whether this cardholder is allowed to have visitors	View/Edit

Lnl_DataConduitManager

Description: Used for non-object related methods.

Abstract: No

Access: View only

Superclass: None

Platforms: OnGuard

Methods:

[static]void RefreshCache();

Refreshes all of the objects, reading in the UDF layout and list values.

[static]string GetCurrentUser();

Returns the user currently logged into DataConduit using the format: LastName, FirstName (UserID).

Properties: None

Lnl_Directory

Description: A directory defined in the security system.

Abstract: No

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	ACCOUNTCATEGORY	Account category	View
string	ACCOUNTCLASS	Account class	View
string	ACCOUNTDISPLAYNAME ATTR	Account display name attribute	View
string	ACCOUNTIDATTR	Account ID attribute	View
string	HOSTNAME	Host name or domain	View
string	NAME	Display name	View
sint32	PORT	Port	View
string	STARTNODE	Start node	View
sint32	TYPE	Directory type. Possible values: 0: LDAP 1: Microsoft Active Directory 2: Microsoft Windows NT 4 Domain 3: Microsoft Local Accounts	View
boolean	USESSL	Use SSL.	View

See the ID CredentialCenter User Guide for more information about directory properties.

Lnl_Element

Description: The base class for many data classes.

Abstract: Yes

Access: View only

Superclass: None

Platforms: OnGuard

Properties: None

LnI_EventAlarmDefinitionLink

Description: The link between the event type and alarm for a particular device.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	PanelID	Panel ID (ex.: ISC). Key field.	View
sint32	DeviceID	Device ID (ex.: Alarm panel, Reader). Key field.	View
sint32	SecondaryDeviceID	Secondary device ID (ex.: Input, Output). Key field.	View
sint32	EventTypeID	Event Type. Key field. See LnI_EventType.ID.	View
sint32	EventSubtypeDefinitionID	Event Subtype. Key field. See LnI_EventSubtypeDefinition.ID.	View
sint32	AlarmDefinitionID	Alarm Definition. See LnI_AlarmDefinition SubtypeID.	View
sint32	EventParameterID	Event parameter ID. Key field. See LnI_EventParameter.ID.	View

LnI_EventParameter

Description: An event parameter.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Description	Parameter description	View
sint32	Value	Parameter value	View

LnI_EventSubtypeDefinition

Description: An event subtype defined in the system.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	TypeID	Event Type ID, see Lnl_EventType.ID.	View
sint32	SubtypeID	ID within the subtype.	View
string	Description	Sub type description	View
sint32	SupportParameters	Supporting Parameter ID	View
sint32	Category	Event subtype category	View

Lnl_EventSubtypeParameterLink

Description: An association between an event subtype and event parameter.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	EventSubtypeDefinitionID	Key field. See Lnl_EventSubtypeDefinition.ID.	View
sint32	EventParameterID	Key field. See Lnl_EventParameter.ID.	View

Lnl_EventType

Description: An event type defined in the system.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Description	Event type description	View

Lnl_Holiday

Description: A holiday that is defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	SegmentID	Segment to which the holiday belongs to.	View
sint32	ExtentDays	How many days the holiday lasts	View
datetime	StartDate	Date the holiday starts	View
string	Name	Holiday name	View

Lnl_HolidayType

Description: A holiday that is defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	SegmentID	Segment to which the holiday belongs to.	View
string	Name	Holiday name	View

Lnl_HolidayTypeLink

Description: Defines what holiday type that is associated with a given holiday

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	HolidayID	Holiday	View
sint32	HolidayTypeID	Holiday type	View

Lnl_IncomingEvent

Description: An event that supports sending incoming events via DataConduIT.

Abstract: No

Superclass: None

Platforms: OnGuard

Methods:

- **sint32** SendIncomingEvent([in] **string** Source, [in] **string** Device, [in] **string** SubDevice, [in] **string** Description, [in] **datetime** Time, [in] **boolean** IsAccessGrant, [in] **boolean** IsAccessDeny, [in] **sint64** BadgeID, [in] **string** ExtendedID);

Parameters:

- Source - text representation of the object/device that generated the event
Variable-length Unicode string with a maximum length of 80 Unicode characters. This parameter is required. The source must be defined in the DataConduIT Sources folder (in the System Administration application) prior to using the Lnl_IncomingEvent::SendIncomingEvent method. For more information, refer to [Add a DataConduIT Source](#) on page 56.
- Device (Optional; available in OnGuard 2006 or later, only) - text representation of a device associated with a DataConduIT Source that generated the event
Variable-length Unicode string with a maximum length of 80 Unicode characters. This parameter is optional. The device must be defined in the DataConduIT Sources folder > DataConduIT Devices tab (in System Administration) prior to using the Lnl_IncomingEvent::SendIncomingEvent method.
- SubDevice (Optional; available in OnGuard 2006 or later, only) - text representation of a sub device associated with a DataConduIT Device that generated the event.
Variable-length Unicode string with a maximum length of 80 Unicode characters. This parameter is optional. The device must be defined in the DataConduIT Sources folder > DataConduIT Sub-Devices tab (in System Administration) prior to using the Lnl_IncomingEvent::SendIncomingEvent method.
- Description - text that describes the event
Variable-length Unicode string with a maximum length of 2000 Unicode characters.

- Time - The time when this event occurred. If this is empty, the current time will be used.
- IsAccessGrant - boolean value that specifies whether the event reported for the DataConduIT Source, Device or Sub-Device will be the “Granted Access” event. This parameter is optional. However, if this parameter is set to true, BadgeID or ExtendedID can be specified to report an “Granted Access” event for a specific OnGuard cardholder. The DataConduIT Source, Device or Sub-Device must be defined in the DataConduIT Sources folder > DataConduIT Devices tab (in the System Administration application) prior to using the Lnl_IncomingEvent::SendIncomingEvent method with the IsAccessGrant parameter set to true. For more information, refer to [Generating Access Granted and Access Denied Events](#) on page 89.
- IsAccessDeny - boolean value that specifies whether the event reported for the DataConduIT Source, Device or Sub-Device will be the “Access Denied” event. This parameter is optional. However, if this parameter is set, then BadgeID or ExtendedID can be specified to report an “Access Denied” event for a specific OnGuard cardholder. The DataConduIT Source, Device or SubDevice must be defined in the DataConduIT Sources folder > DataConduIT Devices tab (in the System Administration application) prior to using the Lnl_IncomingEvent::SendIncomingEvent method with the IsAccessDeny parameter set to true. For more information, refer to [Generating Access Granted and Access Denied Events](#) on page 89.
- BadgeID - Numeric identifier that refers to a badge in the OnGuard database that generated the event. This parameter is optional and is used in association with all badge related events.
- ExtendedID - Extended length string identifier that refers to a PIV-based badge in the OnGuard database that generated the event. Specifies the 128-bit GUID or 200-bit FASC-N. This parameter is optional and is used in association with all badge-related events.

Note: BadgeID is always given precedence over ExtendedID during the search for the badge information to be displayed in Alarm Monitoring.

- **sint32** AcknowledgeAlarm([in] **sint32** CurrentAckStatus, [in] **sint32** SerialNumber, [in] **string** CommServerHostName, [in] **sint32** PanelID, [in] **sint32** AlarmID, [in] **datetime** AlarmTime, [in] **sint32** AckStatus, [in] **string** AckNotes, [out] **sint32** SimultaneousAckStatus);

Description:

Allows acknowledgment of alarms received from the system. Most of the parameters can be extracted from the Lnl_SecurityEvent.

Return:

0 - If acknowledgment fails. Examine the SimultaneousAckStatus value to see if the conflict occurred when processing the request.

1 - If acknowledgment succeeds.

Parameters:

- CurrentAckStatus - current acknowledgment status of the alarm to ensure that simultaneous acknowledgment by other means does not interfere with user’s intent. Possible values are:
 - 0 - No. Initial status for an unacknowledged event.
 - 1 - Yes. Acknowledge.
 - 2 - Note. Acknowledge with note.
 - 3 - In-Progress. Mark event as “in-progress”
- SerialNumber - serial number of the event to acknowledge
- CommServerHostName - host name of the Communication server through which the event arrived

- PanelID - Panel ID associated with the event to ensure the integrity of the acknowledgment request
- AlarmID - Event type ID associated with the event to ensure the integrity of the acknowledgment request
- AlarmTime - Time the event occurred to ensure the integrity of the acknowledgment request
- AckStatus - Acknowledgment status to set. See the CurrentAckStatus parameter description for possible values.
- AckNotes - Acknowledgment notes to set. AckStatus must be 2.
- SimultaneousAckStatus - Value greater than 0 if alarm had been acknowledged by other means. Contains the new acknowledgment status if that was the case. See the CurrentAckStatus parameter description for possible values.

Note: Return value of 4 indicates that no simultaneous acknowledgment occurred.

Properties: None

Generating Access Granted and Access Denied Events

The IsAccessGrant, IsAccessDeny, Badge ID and ExtendedID parameters can be used to generate access granted and access denied events as follows:

- IsAccessGrant and IsAccessDeny are mutually exclusive (i.e., either one or the other can be set to true but not both).
- If IsAccessGrant or IsAccessDeny is set to true, any text that may be specified for the Description parameter will be ignored.

Notes: When a user writes a script that invokes the Lnl_IncomingEvent::SendIncomingEvent method, he or she may optionally specify the IsAccessGrant or IsAccessDeny parameters to generate “Granted Access” or “Access Denied” events respectively.

The above functionality will work similarly if the name of the Source and Device parameters correspond to an Access panel and Reader configured in the system. If these conditions are met then the “Granted Access” or “Access Denied” events will be reported for the specified Access panel and Reader based on how the IsAccessGrant and IsAccessDeny parameters are set.

Using Device and SubDevice in Scripts

A script that invokes the Lnl_IncomingEvent::SendIncomingEvent method may optionally include the Device and SubDevice name. These parameters are reported (to Alarm Monitoring) in the following manner:

- If the Device name is empty, the event will only be reported for the DataConduIT Source
- If the Device name exists and is found in the OnGuard database, the event will be reported for the DataConduIT Device (i.e., Controller and Device columns respectively show the DataConduIT Source and DataConduIT Device that generated the alarm).
- If the SubDevice name exists and is found in the OnGuard database, the event will be reported for the DataConduIT Sub-Device (i.e., Controller, Device, and Input/Output columns respectively show the DataConduIT Source, DataConduIT Device, and DataConduIT Sub-Device that generated the alarm).

Note: The DataConduIT Source, Device, and SubDevice names must all match what has been configured in the OnGuard database in order for the event to be reported in Alarm Monitoring.

LnI_LoggedEvent

Description: Represents an event that has been logged to the database.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	SerialNumber	Serial number of the event. Key field.	View
sint32	PanelID	Panel at which the event occurred. Key field.	View
datetime	Time	Time when event had occurred	View
string	Description	Description of the event	View
sint32	DeviceID	Device ID at which event occurred (LnI_Reader, LnI_AlarmPanel, etc.)	View
string	ExtendedID	Extended length identifier of the card (where available) which caused the event	View
sint32	SecondaryDeviceID	Secondary device ID at which event occurred (ex. LnI_Input)	View
sint32	SegmentID	Segment where event occurred	View
sint32	Type	Event type i.e., "duress", "system", etc. Corresponds to LnI_EventSubtypeDefinition.TypeID and LnI_EventType.ID.	View
sint32	SubType	Event sub-type i.e., "granted", "door forced open", etc. Corresponds to LnI_EventSubtypeDefinition.SubTypeID.	View
string	EventText	Text associated with event	View
sint64	CardNumber	Card (where available) which caused the event	View
sint32	IssueCode	Issue code of the card	View
sint32	AssetID	Asset (where available) which caused the event	View

Type	Name	Description	Access
sint32	AccessResult	The level of access that was granted that resulted from reading the card. Possible values: 0: Other 1: Unknown 2: Granted 3: Denied 4: Not Applicable	View
boolean	CardholderEntered	Whether entry was made by the cardholder	View
boolean	Duress	Indicates whether this card access indicates an under duress/emergency state	View
sint32	PersonID	Internal ID of the person who is assigned the badge at the time of the access event. See Lnl_Person.ID.	View

Lnl_LogicalSystemAccount

Description: An account in a logical system that is associated with a given person.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	PersonID	Reference to the associated Lnl_Person	View/Edit
sint32	LogicalSystemType	Identifies the Card or Identity Management System. 1 = ActivIdentity CMS	View/Edit
sint32	LogicalSystemID	The identifier for the Card or Identity Management System. ActivIdentity CMS logical system ID's are identified by their ID as seen in the CMS server configuration in System Administration.	View/Edit

LnI_MobileVerify

Description: Specifically designed for OnGuard MobileVerify software application. The class currently contains two static methods that allow to log an access grant or deny transaction based on input parameters.

Abstract: No

Superclass: None

Platforms: OnGuard

Methods:

- [static] void RecommendProperties([out] **string** LogicalName, [out] **string** AssociatedDropdown, [out] **string** DenyText, [out] **sint32** DenyColor, [out] **boolean** DenyOverride, [out] **string** GrantText, [out] **sint32** GrantColor, [out] **boolean** GrantOverride, [out] **boolean** OverridePrompt, [out] **boolean** NotifyUserOfOperation);

Retrieves configuration information of how the MobileVerify feature is setup.

Note: This should be called prior to using all other methods of this object. Use the value returned in the *AssociatedDropdown* parameter as the name of the property in *LnI_Cardholder* to retrieve the enumerated values.

Parameters:

- CurrentLevel - This is the ID of the value of the cardholder's force protection level.
- SystemLevel - This is the ID of the value of the current system's force protection level.
- CardholderName - name of cardholder
- SSNo - social security of cardholder
- ReaderName - Name of reader being opened (can be null)
- GateName - Name of gate or building associated with this reader or mobile unit
- [static] void LogGrantTransaction([in] **sint32** CurrentLevel, [in] **sint32** SystemLevel, [in] **string** CardholderName, [in] **string** SSNo, [in] **string** ReaderName, [in] **string** GateName);

Logs an access grant transaction based on the input parameters. This method is used to specify that the operator has granted the user access.

Note: This should be called when the operator clicks a grant button. It should not reflect whether or not the cardholder's force protection level was actually grant or deny. This routine will appropriately log the correct transaction. For example, if the operator clicks Grant on a cardholder whose force protection level is LESS than the system setting (deny access), this routine will log a grant-override transaction.

Parameters:

- CurrentLevel - This is the index of the combo box from the cardholder's force protection level.
- SystemLevel - This is the index of the combo box from the current system's force protection level.
- CardholderName - name of cardholder
- SSNo - social security of cardholder
- ReaderName - Name of reader being opened (can be null)
- GateName - Name of gate or building associated with this reader or mobile unit
- [static] void LogDenyTransaction([in] **sint32** CurrentLevel, [in] **sint32** SystemLevel, [in] **string** CardholderName, [in] **string** SSNo, [in] **string** ReaderName, [in] **string** GateName);

Logs an access deny transaction based on the input parameters.

Parameters:

- CurrentLevel - This is the index of the combo box from the cardholder's force protection level.
- SystemLevel - This is the index of the combo box from the current system's force protection level.
- CardholderName - name of cardholder
- SSNo - social security of cardholder
- ReaderName - Name of reader being opened (can be null)
- GateName - Name of gate or building associated with this reader or mobile unit

Access: In order to use the class users need to have a Mobile Sentry license.

Properties: None

LnI_MonitoringZone

Description: A Monitoring zone defined in the system.

Abstract: No

Access: View

Superclass: LnI_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Monitoring zone name	View
sint32	SegmentID	Monitoring zone's Segment ID	View

LnI_MonitoringZoneCameraLink

Description: Defines what cameras are associated with a given monitoring zone.

Abstract: No

Access: View

Superclass: LnI_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	MonitoringZoneID	Monitoring Zone ID. Key field. See LnI_MonitoringZone.ID.	View
sint32	PanelID	Panel ID for the camera. Key field. See LnI_Camera.PanelID.	View

Type	Name	Description	Access
sint32	CameraID	Camera ID. Key field. See LnI_Camera.ID.	View

LnI_MultimediaObject

Description: An image belonging to a person in the security system.

Abstract: No

Access: View/Add/Delete

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	DATATYPE	Data type. Key field. Possible values: 0: Normal Image 1: Normal Image with Chromakey Mask 2: Thumbnail Image	View/Edit
sint32	OBJECTTYPE	Object type. Key field. Possible values: 1: Photo 8: Signature 10: Hand Geometry	View/Edit
sint32	PERSONID	Internal ID of the person who owns this object. See LnI_Person.ID.	View/Edit
uint8[]	DATA	Array of image data.	View/Edit
datetime	LASTCHANGED	Image last changed	View

LnI_Panel

Description: A panel defined in the security system.

Abstract: No

Access: View only

Superclass: LnI_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	NAME	Display name	View
string	PANELTYPE	Panel type name	View
sint32	SEGMENTID	Lnl_Segment.ID - ID of the segment	View
string	WORKSTATION	Panel workstation name	View

Lnl_Person

Description: A cardholder or visitor in the security system.

Abstract: Yes

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	FIRSTNAME	First name	View/Edit
datetime	LASTCHANGED	Person last changed	View
string	LASTNAME	Last name	View/Edit
string	MIDNAME	Middle name	View/Edit
string	SSNO	Person's identification number	View/Edit

Lnl_Reader

Description: A reader defined in the security system.

Abstract: No

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	PANELID	ID of the panel to which this reader belongs. Key field.	View
sint32	READERID	Internal database ID. Key field.	View
string	NAME	Display name.	View
sint32	TimeAttendanceType	The time and attendance reader configuration. not used = 0 (or <empty>) Entrance Reader = 1 Exit Reader = 2	View

Lnl_Segment

Description: A segment or segment group defined in the security system. Present in segmented systems only.

Abstract: Yes

Access: View only

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	NAME	Display name	View

Lnl_SegmentGroup

Description: A segment group in the security system. Present in segmented systems only.

Abstract: No

Access: View only

Superclass: Lnl_Segment

Platforms: OnGuard

Properties: Same properties as in Lnl_Segment.

Lnl_SegmentUnit

Description: A segment in the security system. Present in segmented systems only.

Abstract: No

Access: View only

Superclass: Lnl_Segment

Platforms: OnGuard

Properties: Same properties as in Lnl_Segment.

Lnl_Timezone

Description: A time zone defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	SegmentID	Segment ID to which the time zone belongs.	View
string	Name	Name of the timezone	View

Lnl_TimezoneInterval

Description: A time zone defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
sint32	TimezoneID	Lnl_Timezone of which this interval is a part of.	View
datetime	StartTime	Time of day when interval becomes active	View
datetime	EndTime	Time of day when interval stops being active	View
boolean	Monday - Sunday	Day of the week when interval is active	View
boolean	HolidayType1 - HolidayType8	Holiday type during which the interval is active	View

LnI_User

Description: A user defined in the system.

Abstract: No

Access: View/Add /Modify/Delete

Superclass: LnI_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	LogonID	Internal Account User name	View/Edit
string	Password	Internal Account Password	Edit
string	FirstName	First Name	View/Edit
string	LastName	Last Name	View/Edit
boolean	AllowManualLogon	Allow user to manually log-in	View/Edit
boolean	AllowUnifiedLogon	Allow single-sign-on	View/Edit
boolean	Enabled	Determines whether user is enabled	View/Edit
sint32	SystemPermissionGroupID	System User Permission Group. See LnI_UserPermissionGroup.ID.	View/Edit
sint32	MonitoringPermissionGroupID	Monitor User Permission Group. See LnI_UserPermissionGroup.ID.	View/Edit
sint32	CardPermissionGroupID	Cardholder User Permission Group. See LnI_UserPermissionGroup.ID.	View/Edit
sint32	ReportPermissionGroupID	Indicates the Report Permission Group ID. This is a required field, but defaults to 0 which provides no report permissions.	View/Edit
sint32	FieldPermissionID	Field/Page Access Group. See LnI_UserFieldPermissionGroup.ID.	View/Edit
sint32	PrimarySegmentID	User's Primary Segment ID	View/Edit
sint32	MonitoringZoneID	Monitoring Zone ID. See LnI_MonitoringZone.ID.	View/Edit
datetime	Created	Date user was created	View
datetime	LastChanged	Date user was modified	View
string	Notes	Notes associated with the user	View

Type	Name	Description	Access
boolean	AutomaticallyCreated	An automatic user is one that has been created in "bulk" using the Bulk User Tool. This property is set to false for all users except those created using the Bulk User Tool. It is included in the application programming interface (API) for filtering only.	View
sint32	DatabaseID	Stores the replication setting for the User; applies to Enterprise systems only. The value has a default value of 'Local System Only' which matches the default through the OnGuard software.	View/Edit

LnI_UserAccount

Description: An association between a user and its directory account.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: None

Platforms: OnGuard

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	AccountID	ID of the entry in the external directory. The ID is the value of the attribute specified in the LnI_Directory.AccountIDAttr property. For example, for Microsoft directories, this property would contain the account's security identifier (SID).	View/Edit
sin32	DirectoryID	Internal ID of the directory to which this account belongs. See LnI_Directory.ID.	View/Edit
sint32	UserID	Internal ID of the user who owns this account. See LnI_User.ID.	View/Edit

LnI_UserPermissionGroup

Description: A permission group defined in the system.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Permission Group name	View
sint32	Type	Permission Group Type: System = 1 Cardholder = 2 Monitor = 3	View
sint32	SegmentID	Group's Segment ID	View
sint32	PTZPriority	PTZ Priority for the users belonging to this group	View
boolean	CanLoginToDataConduIT	Shows if the user in this group can login to DataConduIT	View
boolean	CanViewLiveVideo	Shows if the user in this group can view live video	View
boolean	CanViewRecordedVideo	Shows if the user in this group can view recorded video	View
boolean	CanSearchVideo	Shows if the user in this group can search video	View
boolean	DevicesExcluded	Shows if the devices in the associated group are excluded	View

LnI_UserFieldPermissionGroup

Description: The permission group assigned to the user.

Abstract: No

Access: View

Superclass: LnI_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Permission Group name	View
sint32	SegmentID	Group's Segment ID	View

LnI_UserPermissionDeviceGroupLink

Description: Describes a link between a device group and a permission.

Abstract: No

Access: View

Superclass: LnI_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	UserPermissionGroupID	User permission group. See LnI_UserPermissionGroup.ID.	View
sint32	DeviceGroupID	Device Group ID. See LnI_CameraGroup.ID.	View

LnI_UserReportPermissionGroup

Description: A report permission group defined in the system.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Permission Group name	View
sint32	SegmentID	Group's Segment ID	View

Type	Name	Description	Access
sint32	DatabaseID	Stores the replication setting for the group. Applies to Enterprise systems only. The value has a default value of 'Local System Only' which matches the default through the OnGuard software.	View

Lnl_UserSecondarySegment

Description: An association between a user and its assigned secondary segments.

Abstract: No

Superclass: Lnl_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	UserID	Internal ID of the user Lnl_User.ID.	View/Edit
sint32	SegmentID	User's Segment ID	View/Edit

Lnl_Visit

Description: A visit in the security system.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: Lnl_Element

Platforms: OnGuard

Type	Name	Description	Access
sint32	CARDHOLDERID	LNL_CARDHOLDER.ID - the host	View/Edit
boolean	EMAIL_INCLUDE_DEF_RECIPENTS	Whether the default recipients are notified	Edit
boolean	EMAIL_INCLUDE_HOST	Whether the host is notified	Edit
boolean	EMAIL_INCLUDE_VISITOR	Whether the visitor is notified	Edit
string	EMAIL_LIST	A list of semi-colon separated e-mail recipients (other than the visitor, host or defaults) Ex: abc@123.com;xyz@123.com	Edit

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
datetime	LASTCHANGED	Visit last changed	View
string	PURPOSE	Visit purpose	View/Edit
datetime	SCHEDULED_TIMEIN	Scheduled start time	View/Edit
datetime	SCHEDULED_TIMEOUT	Scheduled end time	View/Edit
datetime	TIMEIN	Actual start time	View
datetime	TIMEOUT	Actual end time	View
sint32	TYPE	Visit type, values are user-defined	View/Edit
sint32	VISITORID	Lnl_Visitor.ID - the visitor	View/Edit

Methods:

`void SignVisitOut();`

Signs a visit out, modifying the visit and setting TIMEOUT to current date/time. Any associated badge with the visitor is deactivated and set to the status as configured in the OnGuard software.

`void SignVisitIn([in]sint64 BadgeTypeID, [in]string PrinterName, [in]sint64 AssignedBadgeID);`

Signs a visit in, modifying the visit and setting TIMEIN to current date/time. If AssignedBadgeID is set to a valid ID, the badge is automatically assigned to the visitor and made active.

Parameters:

- badgeTypeID - This is the badge type you want to assign the visitor.
- AssignedBadgeID - This is the badge ID you want to assign the visitor, a badge already in the system.
- printerName - The name of the printer you want to use to print out the disposable badge

Note: If badgeTypeID is provided so must the printerName (unless there is a default printer set up for the badgeTypeID specified) and AssignedBadgeID will be ignored. If AssignedBadgeID is specified, badgeTypeID and printerName are ignored. See the Visitor Management User Guide for more detailed documentation on visits and signing them in.

Lnl_VisitEmailRecipient

Description: A visit e-mail recipient in the security system.

Abstract: No

Access: View only

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	ACCOUNTID	ID of the entry in the external directory. The ID is the value of the attribute specified in the Lnl_Directory.AccountIDAttr property. For example, for Microsoft directories, this property would contain the account's security identifier (SID)	View
sint32	DIRECTORYID	Internal ID of the directory to which this account belongs. See Lnl_Directory.ID.	View
string	EMAILADDRESS	Recipient e-mail address	View
boolean	INCLUDEDEFAULTRECIPIENTS	Whether the default recipients are notified	View
boolean	INCLUDEHOST	Whether the visit host is notified	View
boolean	INCLUDEVISITOR	Whether the visitor is notified	View
sint32	PERSONID	Lnl_Person.ID - ID of the person receiving the e-mail	View
sint32	RECIPIENTNUMBER	Internal database ID. Key field.	View
sint32	SEGMENTID	Segment for default recipients	View
sint32	VISITID	Lnl_Visit.ID - ID of the visit. Key field.	View

Lnl_Visitor

Description: A visitor in the security system.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: Lnl_Person

Platforms: OnGuard

Properties: The class has all the properties of the Lnl_Person class, plus any custom fields defined by the end user.

User-Defined Value Lists (LNL_BadgeStatus, Lnl_Title, ...)

Description: Any user-defined list in the system, populated via List Builder.

Abstract: No

Access: Full (View/Add/Modify/Delete)

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	NAME	Name of the list value	View/Edit
sint32	SEGMENTID	Segment the list belongs to	View/Edit

Association Classes

Lnl_AccessLevelGroupAssignment

Description: An association between an access level and the group in which it belongs.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_AccessLevel	ACCESSLEVEL	Reference to the access level
ref:Lnl_AccessGroup	ACCESSGROUP	Reference to the access group

Lnl_BadgeOwner

Description: An association between a badge and the person who owns it.

Abstract: Yes

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Badge	BADGE	Reference to the badge
ref:Lnl_Person	PERSON	Reference to the person

LnI_CardholderAccount

Description: An association between an account and the cardholder with which it is associated.

Abstract: No

Superclass: LnI_PersonAccount

Platforms: OnGuard

Properties:

Type	Name	Description
ref:LnI_Account	ACCOUNT	Reference to the account
ref:LnI_Cardholder	PERSON	Reference to the cardholder.

LnI_CardholderBadge

Description: An association between a badge and the cardholder who owns it.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:LnI_Badge	BADGE	Reference to the badge
ref:LnI_Visitor	PERSON	Reference to the visitor

LnI_CardholderMultimediaObject

Description: An association between a multimedia object and the cardholder who owns it.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:LnI_MultimediaObject	MULTIMEDIAOBJECT	Reference to the multimedia object
ref:LnI_Cardholder	PERSON	Reference to the cardholder

LnI_DirectoryAccount

Description: An association between an account and the directory in which it is stored.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Account	ACCOUNT	Reference to the account
ref:Lnl_Directory	DIRECTORY	Reference to the directory

Lnl_MultimediaObjectOwner

Description: An association between a multimedia object and the person who owns it.

Abstract: Yes

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_MultimediaObject	MULTIMEDIAOBJECT	Reference to the multimedia object
ref:Lnl_Person	PERSON	Reference to the person

Lnl_PersonAccount

Description: An association between an account and the person with which it is associated.

Abstract: Yes

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Account	ACCOUNT	Reference to the account
ref:Lnl_Person	PERSON	Reference to the person

Lnl_ReaderEntersArea

Description: An association between a reader and the APB area to which it allows entry.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Area	AREA	Reference to the APB area
ref:Lnl_Reader	READER	Reference to the reader

Lnl_ReaderExitsArea

Description: An association between a reader and the APB area to which it allows departure from.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Area	AREA	Reference to the APB area
ref:Lnl_Reader	READER	Reference to the reader

Lnl_SegmentGroupMember

Description: An association between a segment unit and the segment group of which the unit is a member. Present in segmented systems only.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_SegmentGroup	GROUP	Reference to the segment group
ref:Lnl_SegmentUnit	MEMBER	Reference to the segment unit

Lnl_VisitorAccount

Description: An association between an account and the visitor with which it is associated.

Abstract: No

Superclass: Lnl_PersonAccount

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Account	ACCOUNT	Reference to the account
ref:Lnl_Visitor	PERSON	Reference to the visitor

Lnl_VisitorMultimediaObject

Description: An association between a multimedia object and the visitor who owns it.

Abstract: No

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_MultimediaObject	MULTIMEDIAOBJECT	Reference to the multimedia object
ref:Lnl_Visitor	PERSON	Reference to the visitor

Event Classes

All event classes are view only and are not abstract.

Lnl_AccessEvent

Description: An event occurring due to the presentation of credentials at a reader. Credentials here are represented as being stored on a card, but the “card” could be any form factor. Similarly, the “reader” represents any system that can read the credentials on the card. This class includes information read from the card (card number, biometric information) in addition to what access was granted (granted/denied and under duress).

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
sint32	ACCESSRESULT	The level of access that was granted that resulted from reading the card. Possible values: 0: Other 1: Unknown 2: Granted 3: Denied 4: Not Applicable
sint32	AREAENTEREDID	The ID of the area that was entered, if any.
sint32	AREAEXITEDID	The ID of the area that was exited, if any.
string	ASSETID	The ID of the asset related to this event, if any.
boolean	CARDHOLDERENTERED	Whether entry was made by the cardholder.
sint32	CARDNUMBER	The badge ID for the card that was read, if available.
boolean	DURESS	Indicates whether this card access indicates an under duress/emergency state.
sint32	ELEVATORFLOOR	The elevator floor on which the access event was generated, if any.
string	ExtendedID	The extended length identifier for the card that was read, if available.
sint32	FACILITYCODE	The facility code for the card that was read, if available.
boolean	ISREADABLECARD	Whether the card could be read. If it could not be read (due to an invalid card format or damage to the card), the other properties of this class relating to card information will be null.
sint32	ISSUECODE	The issue code for the card that was read, if available.

Lnl_Alarm

Description: An alarm in the system. The Lnl_Alarm class is embedded directly into the Lnl_SecurityEvent class, because an alarm cannot happen without an event and an event can be mapped to one and only one alarm definition. Since this is an embedded object, you cannot query for it.

Abstract: No

Access: View only

Superclass: None

Platforms: OnGuard

Properties: These priorities are based off the Alarm Configuration folder in System Administration.:

Type	Name	Description
string	DESCRIPTION	A human readable of the event parameter
string	EVENTPARAMDESCRIPTION	A human readable brief description of the event parameter
boolean	ISACTIVE	Whether the alarm is active
boolean	MUSTACKNOWLEDGE	Whether the alarm has to be acknowledged
sint32	PRIORITY	The alarm's priority

Lnl_Event

Description: An event occurring in the OnGuard system.

Superclass: __ExtrinsicEvent

Platforms: OnGuard

Properties:

Type	Name	Description
string	DESCRIPTION	A human readable, brief description of this event.
datetime	TIME	The time when this event occurred.

Lnl_FireEvent

Description: An event that relates to a fire hazard and/or fire hardware.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
sint32	TroubleCode	A trouble code associated with the fire event.

Lnl_FunctionExecEvent

Description: An event that consists of a function that is executed when a given event occurs. Input arguments may also be included.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
sint32	FunctionID	The ID of the function that was executed.
sint32	InitiatingEventID	The ID of the event that caused the function to be executed.
sint32	FunctionInputArguments	Any input arguments to the function that was executed.

Lnl_IntercomEvent

Description: An event occurring on intercom hardware such as an intercom exchange or an intercom station.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
sint32	IntercomData	Additional data for the intercom event that occurred.
sint32	LineNumber	The line number involved in the intercom event.

Lnl_OtherSecurityEvent

Description: An event that is not card related and not access-related, such as door forced open and alarm restored. The Lnl_OtherSecurity event class supports all event types that were not included in the other event sub-classes. A combination of all of the above classes yields all security events and alarms available in the system.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties: All properties belong to the superclass.

Lnl_SecurityEvent

Description: An event occurring in the physical security system.

Superclass: Lnl_Event

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	DEVICEID	The ID of the device where this event originated	
sint32	ID	The ID that uniquely identifies the type of this event	
sint32	PANELID	The ID of the panel where this event originated	
sint32	SECONDARYDEVICEID	The ID of the secondary device where this event originated	
sint32	SEGMENTID	The ID of the segment that the panel is in	
sint32	SERIALNUMBER	A number that uniquely identifies the instance of the event for a particular panel	
object:Lnl_Alarm	ALARM	The alarm associated with the event, if there is one	
sint32	Type	Event type i.e., "duress", "system", etc. Corresponds to Lnl_EventSubtypeDefinition.TypeID and Lnl_EventType.ID.	View
sint32	SubType	Event sub-type i.e., "granted", "door forced open", etc. Corresponds to Lnl_EventSubtypeDefinition.SubTypeID.	View

Lnl_StatusChangeEvent

Description: An event that indicates a change of status for the device specified.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
sint32	CommunicationStatus	The status for the communication link with the device specified in the event.
sint8	NewStatus	The new status of the device specified in the event.
sint8	OldStatus	The old status for the device specified in the event.

Lnl_TransmitterEvent

Description: A personal safety event involving a transmitter.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
sint32	TransmitterBaseID	The base ID of the transmitter associated with the event.
sint32	TransmitterID	The ID of the transmitter associated with the event.
sint32	TransmitterInputID	The ID of the input on the transmitter associated with the event.
boolean	VerifiedAlarm	Boolean value indicating whether the transmitter message is known to be verified.

Lnl_VideoEvent

Description: An event associated with video equipment such as video recorders and cameras.

Superclass: Lnl_SecurityEvent

Platforms: OnGuard

Properties:

Type	Name	Description
datetime	StartTime	The time the video event started.
datetime	EndTime	The time the video event ended.
sint32	Channel	The physical channel the camera is connected to that is creating this event.

Lnl_VisitEvent

Description: An event associated with a visit.

Superclass: __InstanceOperationEvent

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	The internal database ID	View
string	Name	The user-friendly name of this object.	View

Type	Name	Description	Access
sint32	CardholderID	The host of the visit event.	View
sint32	DelegateID	The person who schedules or maintains the event instead of the host.	View
sint32	DatabaseID	The database identifier in an Enterprise system that identifies the system that owns the event.	View
datetime	Scheduled_TimeIn	The time the visit event is scheduled to start.	View
datetime	Scheduled_TimeOut	The time the visit event is scheduled to complete.	View
datetime	LastChanged	The last time the properties of the visit event changed.	View
sint32	SignInLocationID	The ID of the visitor sign in location.	View

Command and Control: Classes and Methods

Lnl_AlarmInput

Description: Inherits from Lnl_Input. Implements the input control methods and represents an alarm input found on an input control module.

Lnl_AlarmOutput

Description: Inherits from Lnl_Output. Implements the relay control methods and represents an alarm relay found on an input or output control module.

Notes: The Activate(), Deactivate(), and Pulse() methods are not supported on Mercury, NGP, or Casi alarm panels when those panels are designated as elevator hardware.

Access panels with a dual reader that are designated as elevator hardware will not generate instances of this class.

Lnl_AlarmPanel

Description: This class represents the Alarm input or output control module.

Methods:

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC

uint32 Status – device status:		
uint32 Status	Description	Device status
ONLINE_STATUS	Online	0x01
OPTIONS_MISMATCH_STATUS	Options Mismatch	0x02
CABINET_TAMPER	Cabinet Tamper	0x04
POWER_FAIL	Power Failure	0x8

Lnl_Input

Description: Abstract class that represents any kind of alarm input. It declares methods for controlling such output.

Methods:

void Mask();

Sends a command to mask a specific alarm input.

void Unmask();

Sends a command to unmask a specific alarm input.

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the ISC.

uint32 Status – device status:	
ALRM_STATUS_SECURE	0x00
ALRM_STATUS_ACTIVE	0x01
ALRM_STATUS_GND_FLT	0x02
ALRM_STATUS_SHRT_FLT	0x03
ALRM_STATUS_OPEN_FLT	0x04
ALRM_STATUS_GEN_FLT	0x05

Lnl_IntrusionArea

Description: Implements the control methods for the Intrusion Area.

Methods:

void Arm([in] sint32 armState);

armState - the desired arm state of the area. Values include:

Value	Name	Description
1	PerimeterArm	Sends a command to perform a perimeter arm.
2	EntirePartitionArm	Sends a command to perform an entire partition arm.
3	MasterDelayArm	Sends a command to perform a delayed master arm.
4	MasterInstantArm	Sends a command to perform an instant master arm.
5	PerimeterDelayArm	Sends a command to perform a delayed perimeter arm.
6	PerimeterInstantArm	Sends a command to perform an instant perimeter arm.
7	PartialArm	Sends a command to perform a partial arm.
9	AwayArm	Sends a command to perform an away arm.
10	AwayForcedArm	Sends a command to perform an away forced arm.
11	StayArm	Sends a command to perform a stay arm.
12	StayForcedArm	Sends a command to perform a stay forced arm.

void Disarm()

Sends a command to disarm the area.

void SilenceAlarms ()

Sends a command to silence area alarms.

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:	
OFFLINE_STATUS	0x00
ONLINE_STATUS	0x01

Lnl_IntrusionDoor

Description: Implements the control methods for the Intrusion Door.

Methods:

void Open()

Sends a command to open the intrusion door.

void SetMode([in] sint32 Mode);

Sends a command to change the door mode.

Mode – door mode:	
DoorLock	0x0
DoorUnlock	0x1
SetDoorSecure	0x2

Lnl_IntrusionOutput

Description: Abstract class that inherits from Lnl_Output. Declares the relay control methods and represents an output device of the Intrusion Panel.

Note: This class does not support the Pulse() method.

Lnl_IntrusionZone

Description: Implements the control methods for the Intrusion Zone.

Methods:

void Bypass()

Sends a command to open by pass the alarm zone.

void UnBypass();

Sends a command to un bypass the alarm zone.

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:	
OFFLINE_STATUS	0x00
ONLINE_STATUS	0x01

Lnl_IntrusionZoneOutput

Description: Inherits from Lnl_Output. Implements the relay control methods and represents an Output Zone defined on the Intrusion Panel.

Note: This class does not support the Pulse() method.

Lnl_OffBoardRelay

Description: Inherits from Lnl_Output. Implements the relay control methods and represents an Off-Board relay connected to the Intrusion Panel.

Methods:

void Toggle();

Toggles the state of the specific alarm relay.

Note: This class does not support the Pulse() method.

Lnl_OnBoardRelay

Description: Inherits from Lnl_Output. Implements the relay control methods and represents an On-Board relay of the Intrusion Panel.

Note: This class does not support the Pulse() method.

Lnl_Output

Description: Abstract class that represents any kind relay output. It declares methods for controlling such output.

Methods:

void Activate()

Sends a command to activate a specific alarm relay.

void Deactivate()

Sends a command to deactivate a specific alarm relay.

void Pulse()

Sends a momentary pulse command to a specific alarm relay.

Example (VB Script):

```
Set wbemServices = GetObject("winmgmts://./root/onguard")
` run the query. this call returns a SWbemObjectSet that contains a
` list of all outputs in the system.
Set outputSet = wbemServices.ExecQuery("select * from Lnl Output")
Dim Counter
Counter = 0
` for each output - pulse three times
While Counter < 3 ` Test value of Counter.
for each output in outputSet
` Pulse the output
output.Pulse()
WScript.Sleep 1000
next
Counter = Counter + 1 ` Increment Counter.
Wend
```

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:		
uint32 Status	Description	Device status
ALRM_STATUS_SECURE	Output Secure	0
ALRM_STATUS_ACTIVE	Output Active	1

Lnl_Panel

Description: This class represents the Intelligent System Controller.

Methods:

void DownloadFirmware()

Sends a download firmware command to the ISC.

void DownloadDatabase()

Sends a command to the ISC to download the cardholder database.

void ResetUseLimit()

Sends a command to reset the use limit of all cardholders within the ISC.

void UpdateHardwareStatus()

Sends a command to retrieve the status of the Intelligent System controller and all downstream hardware connected to the specific system controller.

void Connect()

Used for dial-up only. This command instructs the host to connect to the ISC via dial-up.

void Disconnect()

Used for dial-up only. This command instructs the host to send a disconnect command to the ISC.

void SetClock()

Sends the current time down to the ISC.

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the ISC.

uint32 Status – device status:		
uint32 Status	Description	Device status
ONLINE_STATUS	Online	0x01
OPTIONS_MISMATCH_STATUS	Options Mismatch	0x02
CABINET_TAMPER	Cabinet Tamper	0x04
POWER_FAIL	Power Failure	0x8

uint32 Status – device status:		
uint32 Status	Description	Device status
DOWNLOADING_FIRMWARE	Downloading Firmware	0x10

Lnl_Reader

Description:

Methods:

void OpenDoor()

Sends a command to open the door for a specific reader.

void SetMode([in] sint32 Mode)

Sends a command to set the current operating mode of a reader.

void GetMode ([out] sint32 Mode)

Retrieves current mode of the reader. Mode is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

Parameters:

sint32 Mode: Reader mode to be set. Allowed values are:	
MODE_LOCKED	0x0
MODE_CARDONLY	0x1
MODE_PIN_OR_CARD	0x2
MODE_PIN_AND_CARD	0x3
MODE_UNLOCKED	0x4
MODE_FACCODE_ONLY	0x5
MODE_CYPHERLOCK	0x6
MODE_AUTOMATIC	0x7

You can set the current mode of the reader to an authentication mode using the ID retrieved with the Lnl_AuthenticationMode class. Authentication mode IDs are not static like the system-defined reader modes in the table above.

void SetBiometricVerifyMode([in] boolean Value)

Sends a command to enable / disable the biometric mode of verification for a reader.

Parameters:

boolean Value: True – enable biometric mode of verification. False – disable biometric mode of verification.

void SetFirstCardUnlockMode([in] boolean Value)

Sends a command to enable/disable first card unlock mode for the reader.

Parameters:

boolean Value: True – enable first card unlock mode. False – first card unlock mode.

void DownloadFirmware()

Sends a download firmware command to the reader interface module.

void GetHardwareStatus([out] uint32 Status)

Retrieves the hardware status for the device. Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:		
uint32 Status	Description	Device status
RDRSTATUS_ONLINE	Online	0x1
RDRSTATUS_OPTION_MISMATCH	Options Mismatch	0x2
RDRSTATUS_CNTTAMPER	Cabinet Tamper	0x4
RDRSTATUS_PWR_FAIL	Power Failure	0x8
RDRSTATUS_TAMPER	Reader Tamper	0x10
RDRSTATUS_FORCED	Door Forced Open	0x20
RDRSTATUS_HELD	Door Held Open	0x40
RDRSTATUS_AUX	Auxiliary Input 1	0x80
RDRSTATUS_AUX2	Auxiliary Input 2	0x100
RDRSTATUS_AUX3	Auxiliary Input 3	0x400
RDRSTATUS_BIO_VERIFY	Bio Verify	0x800
RDRSTATUS_DC_GND_FLT	DC Ground Fault	0x1000
RDRSTATUS_DC_SHRT_FLT	DC Short Fault	0x2000
RDRSTATUS_DC_OPEN_FLT	DC Open Fault	0x4000
RDRSTATUS_DC_GEN_FLT	DC Generic Fault	0x8000
RDRSTATUS_RX_GND_FLT	RX Ground Fault	0x10000
RDRSTATUS_RX_SHRT_FLT	RX Short Fault	0x20000
RDRSTATUS_RX_OPEN_FLT	RX Open Fault	0x40000
RDRSTATUS_RX_GEN_FLT	RX Generic Fault	0x80000

uint32 Status – device status:		
uint32 Status	Description	Device status
RDRSTATUS_FIRST_CARD_UNLOCK	First Card Unlock Mode	0x100000
RDRSTATUS_EXTENDED_HELD_MODE	Extended Held Mode	0x200000
RDRSTATUS_CIPHER_MODE	Cipher Mode	0x400000
RDRSTATUS_LOW_BATTERY	Low Battery	0x800000
RDRSTATUS_MOTOR_STALLED	Motor Stalled	0x1000000
RDRSTATUS_READHEAD_OFFLINE	Read Head Offline	0x2000000
RDRSTATUS_MRDT_OFFLINE	MRDT Offline	0x4000000
RDRSTATUS_DOOR_CONTACT_OFFLINE	Door Contact Offline	0x8000000

Lnl_ReaderOutput

Description: Abstract class, inherits from Lnl_Output. Declares the relay control methods and represents an auxiliary relay found on a reader interface module.

Lnl_ReaderOutput1

Description: Inherits from Lnl_ReaderOutput. Implements the relay control methods and represents the first auxiliary relay found on a reader interface module.

Lnl_ReaderOutput2

Description: Inherits from Lnl_ReaderOutput. Implements the relay control methods and represents the second auxiliary relay found on a reader interface module.

Lnl_ReaderInput

Description: Abstract class, inherits from Lnl_Input. Declares the input control methods and represents an auxiliary input found on a reader interface module.

Lnl_ReaderInput1

Description: Inherits from Lnl_ReaderInput. Declares the input control methods and represents the first auxiliary input found on a reader interface module.

Lnl_ReaderInput2

Description: Inherits from Lnl_ReaderInput. Declares the input control methods and represents the second auxiliary input found on a reader interface module.

Appendices

Property Qualifiers Used In DataConduIT

The following property qualifiers are used for user-defined fields (UDFs) with the following settings. Some of these qualifiers are standard among WMI applications; others are DataConduIT specific.

UDF Setting	Property Qualifier Name	Property Qualifier Value
Required	not_null	true
Read-only	read noedit	true true
Can't view	noview	true
Maximum Length	maxlen	[maximum length value]
Display Name	DisplayName	[display name value]
Database Key	key	true
Database Foreign Key	propagated	[Foreign Class Name].[Foreign Property Name]
Default, Not Required	Optional	true
Default	DefaultValue	[field default value]
Unique	Unique	true

Event Generator

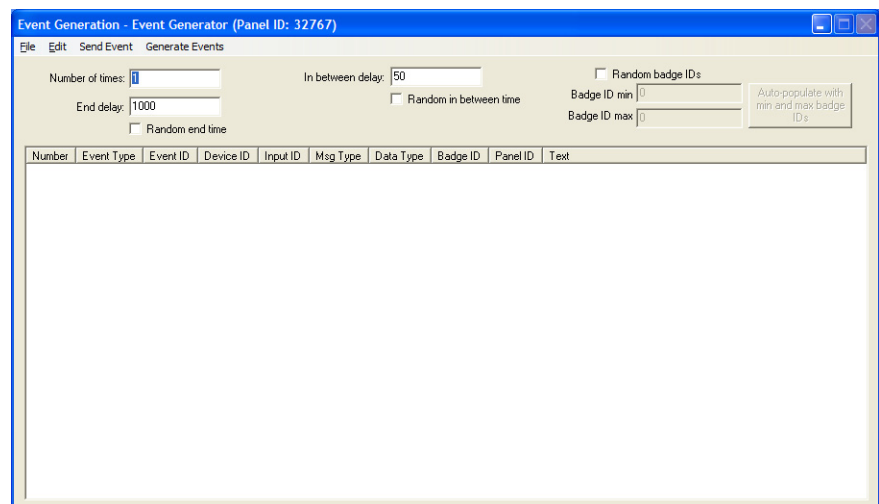
The Event Generator is a utility that is used to generate events without having “live” or online hardware connected to a system; it enables customers who wish to generate events without purchasing hardware to do so.

The Event Generator is available on the Lenel Web site: <http://www.lenel.com/support/downloads/onguard>. (You will need your Lenel login to gain access to this site.)

It is also available on the OnGuard Software Development Kit (SDK) installation disc.

Event Generator Main Window

The Event Generator Main Window displays automatically when the Communication Server is run as an application after the Event Generator is set up. To correctly set up the Event Generator, refer to [Required Event Generator Files](#) on page 137.



Number of times

Number of times each event in the listing window will be generated

End delay

Amount of time that will elapse after the last event is sent

Random end time

If selected, the **End delay** value specified will be ignored, and instead a random time will be used

In between delay

Amount of time that will elapse between events that are sent

Random in between time

If selected, the **In between delay** value specified will be ignored, and instead a random time will be used

Random badge IDs

If selected, badge ID numbers will be randomly generated. This check box must be selected for **Badge ID min**, **Badge ID max**, and [Auto-populate with min and max badge IDs] to be enabled and available for selection.

Badge ID min

The lowest badge ID that is allowed to be randomly selected. Badge IDs will be randomly determined, but will fall in the range between the specified badge ID min and max.

Badge ID max

The highest badge ID that is allowed to be randomly selected. Badge IDs will be randomly determined, but will fall in the range between the specified badge ID min and max.

Auto-populate with min and max badge IDs

Automatically populates the **Badge ID min** and **Badge ID max** fields with values appropriate for your particular database

Listing window

Lists events that have been added, along with the event type, event ID, device ID, input ID, message type, data type, badge ID, Panel ID, and text associated with each.

Edit Event (Simple) Window

The Edit Event (Simple) window is used to add new events or modify existing events using the minimum number of required parameters.

Only non-receiver/intrusion events in the OnGuard system are available in the Edit Event (Simple) window. For receiver/intrusion events, use the Edit Event (Advanced) window.

The Edit Event (Simple) window opens when you select either:

- **Edit > Create Event > Create Event (Simple)**, or
- **Edit > Modify Event > Modify Event (Simple)** when an event is selected

The screenshot shows a dialog box titled "Edit Event" with a blue border and a close button in the top right corner. The dialog contains the following fields:

- Event type:** A dropdown menu with "Access Granted" selected.
- Event sub-type:** A dropdown menu with "Access Granted" selected.
- Panel:** A dropdown menu with "Bldg 1 Access Panel" selected.
- Device:** A dropdown menu with "Bldg 1 South Exit Reader" selected.
- Input or output:** A dropdown menu that is currently empty.
- Badge ID to use for event:** A text input field containing the number "5".

In the top right corner of the dialog, there are two buttons: "OK" and "Cancel".

Event type

Lists all non-receiver/intrusion events in the OnGuard system. For receiver/intrusion events, use the Advanced user interface.

Event sub-type

Lists sub-categories of the selected event type.

Panel

Lists all available panels for the selected event type. The event will be generated for the selected panel.

Device

Lists all available readers for the selected event type (if applicable). The event will be generated for the selected reader.

Input or output

Lists all available inputs and outputs for the selected event type (if applicable). The event will be generated for the selected input or output.

Badge ID to use for event

The entered badge ID will be used in generating the event (if applicable).

OK

If adding a new event, the event will be added. If modifying an event, the modifications will be saved.

Cancel

Closes the Edit Event (Simple) window without adding or modifying any events.

Edit Event (Advanced) Window

The Edit Event (Advanced) window is used to add new events or modify existing events using advanced parameters.

In the Edit Event (Advanced) window, both non-receiver/intrusion and receiver/intrusion events are available. In the Edit Event (Simple) window, only non-receiver/intrusion events are available.

The Edit Event (Advanced) window opens when you select either:

- **Edit > Create Event > Create Event (Advanced)**, or
- **Edit > Modify Event > Modify Event (Advanced)** when an event is selected

The fields available on this window for the data type change depending on which data type is selected. For example, if the `EVENT_DATA_TYPE_STATUS` data type is selected, the **New status**, **Old status**, and **Comm status** fields are displayed and active.

There are six custom data fields: data1, data2, data3, data4, data5, and data6. If a data type uses custom fields, then the field names are displayed instead of data1, data2, data3, etc.

When a data type contains less than six custom data fields, the extra fields are disabled. For example:

- **New status** = data1
- **Old status** = data2
- **Comm status** = data3
- data4, data5 and data6 are not used and are disabled

The screenshot shows the 'Edit Event' dialog box with the following fields and values:

- Event type: System
- Message type: Event
- Event category: Default
- Data type: EVENT_DATA_TYPE_STATUS
- Events: AC Trouble
- New status: 0
- Old status: 0
- Comm status: 0
- data4: 0
- data5: 0
- data6: 0
- Device ID: 0
- Input ID: 0
- Panel ID: 0
- Account Number: [empty]
- Event Code Template: [empty]

Checkboxes and their states:

- Parameterized:
- Associated event text:
- Override Event Generator's panel ID:
- Generate Receiver Account event:

Event type

Lists all categories of events in the OnGuard system. This field is used in combination with the **Event category** drop-down to filter what events are listed in the **Events** drop-down.

Event category

Allows the events in the **Events** drop-down listbox to be filtered based on the category. Non-receiver/intrusion events and receiver/intrusion events are available in this drop-down; in the Simple user interface only non-receiver/intrusion events are available.

Events

Lists all events for the selected event type and event category.

Parameterized

Select this check box to generate an event that uses event parameters.

Note: Not all events support parameters. For more information on event parameters, refer to the OpenDevice Events Guide in the OnGuard Software Development Kit (*Program Files\OnGuard Software Development Kit\OpenDevice*).

Parameter

Enter the parameter value associated with the event to generate. For more information, refer to the OpenDevice Events Guide for events that have the **sb_EventParam** listed.

Message type

Indicates the message type of the event. The available choices are: Event, Status, Video. Most messages will be of the Event type. Status messages are for messages which pass back status information and will not display in Alarm Monitoring. Video events are special events used by video.

Data type

Indicates the type of additional data to be used with the message. For example, some messages can have a badge ID and a specific data type will be used for these so this information can be passed back.

The fields available on this window for the data type change depending on which data type is selected. For example, if the **EVENT_DATA_TYPE_STATUS** data type is selected, the **New status**, **Old status**, and **Comm status** fields are displayed and active.

There are six custom data fields: data1, data2, data3, data4, data5, and data6. If a data type uses custom fields, then the field names are displayed instead of data1, data2, data3, etc.

When a data type contains less than six custom data fields, the extra fields are disabled. For example:

- **New status** = data1
- **Old status** = data2
- **Comm status** = data3
- data4, data5 and data6 are not used and are disabled

If your event does not have additional data, use the **EVENT_DATA_TYPE_STATUS**.

For more information, refer to [Custom Data Fields Displayed for Each Data Type Setting](#) on page 134.

Associated event text

If selected, the text field will become enabled. Indicates if the message is to have associated text with it.

Text

Enter text to be associated with the event

Device ID

This is a downstream device ID that can be used to represent the event is from a downstream device instead of just from a panel. OnGuard uses a three tiered device ID in the format P-D-I; this is the second value.

Input ID

This is a downstream input ID that can be used to represent that the event is from a downstream device instead of just for a panel or its downstream device. OnGuard uses a three tiered device ID in the format P-D-I; this is the third value.

Override Event Generator's panel ID

This checkbox can be used to override the event generator's panel ID so that you can generate an event that is from a different panel.

Panel ID

If the Override Event Generator's panel ID option is being used, you will need to specify the panel ID that will be used for the event in replacement for the event generator's panel ID.

Generate Receiver Account event

Select this check box to generate an event that would be sent from a burglary/intrusion panel to a Central Station receiver connected to OnGuard.

This check box is only available when `EVENT_DATA_TYPE_RECEIVER` is selected from **Data type**. When this box is checked, the **Account Number** and **Event Code Template** fields become available.

Account Number

Enter the account number for the receiver. This number is then displayed in Alarm Monitoring under the Controller column.

Event Code Template

Select the event code format that is used to decode the receiver account event data. This is the same field in *System Administration > Additional Hardware > Receivers > Event Code Templates* tab.

Note: When using the **Event Code Template** drop-down list, the **Event type**, **Event category**, and **Events** drop-down lists are not used.

OK

If adding a new event, the event will be added. If modifying an event, the modifications will be saved.

Cancel

Closes the Edit Event (Advanced) window without adding or modifying any events

Custom Data Fields Displayed for Each Data Type Setting

Data type	Custom data fields and descriptions
EVENT_DATA_ASSET	Badge ID - Card number associated with the asset event.
EVENT_DATA_TYPE_AREAAPB	Area APB ID - Area anti-passback ID.
EVENT_DATA_TYPE_CA (Card Access)	Badge ID - Card number associated with the card event. Issue code - Issue code associated with the card. Bio score - Biometric score for biometric card events.

Custom Data Fields Displayed for Each Data Type Setting

Data type	Custom data fields and descriptions
EVENT_DATA_TYPE_CNA (Card No Access)	Badge ID - Card number associated with the event.
EVENT_DATA_TYPE_FC (Facility Code)	Facility code - Facility code associated with the event. Issue code - Issue code.
EVENT_DATA_TYPE_INTERCOM	Intercom data - Special intercom data associated with the event. Line number - Line number used by special intercom events.
EVENT_DATA_TYPE_INTRUSION	Area ID - Area ID for the intrusion event. User ID - User ID associated with the intrusion event.
EVENT_DATA_TYPE_RECEIVER	Receiver ID - ID of the receiver. Line number - Line number on the receiver. Area ID - Area ID for the event. User ID - User ID associated with the event. Event Code - Event code for the event. The Event Code depends on the selection made from the Event Code Template drop-down list. For example, if SIA is selected from the Event Code Template drop-down list, enter "BA" in the Event Code field for a Burglary Alarm event.
EVENT_DATA_TYPE_STATUS	New status - New status, which is dependent on the type of message. Old status - Old status, which is dependent on type of message. Comm status - Communication status, which is dependent on the type of message. If your event really does not have additional data, you can use the EVENT_DATA_TYPE_STATUS.
EVENT_DATA_TYPE_STATUSREQUEST	Status type - Type of status request. OnGuard has a number of pre-defined types. Status - Status associated with the status type. These values depend on the type of status.
EVENT_DATA_TYPE_TRANSMITTER	Transmitter ID - Transmitter ID associated with the transmitter event
EVENT_DATA_TYPE_VIDEO	Channel - Channel number associated with the video event

Event Generator Menus

File

Save Events

Saves the event list as a file with an EVT extension. This is generally done after the event configuration has been completed.

Load Events

Enables you to load a previously saved event configuration.

Edit

Create Event

Contains a sub-menu of options that are used to create events.

- **Create Event (Advanced):** Enables you to create an event using additional advanced parameters that are not available in the simple mode.
- **Create Event (Simple):** Enables you to create an event using the least number of parameters possible.

Modify Event

Contains a sub-menu of options that are used to modify events.

- **Modify Event (Advanced):** For a selected event, displays the basic parameters and enables you to change them.
- **Modify Event (Simple):** For a selected event, displays advanced parameters and enables you to change them.

Delete Event

Used to delete a selected event. A confirmation message is displayed before the actual deletion occurs.

Clear Events

Clears all events listed in the main window. Make sure to save the events before executing this command if you wish to use the events in the future; otherwise, you will need to recreate them.

Send Event

This option in the Edit menu performs the same function as **Send Event**. For more information, refer to [Send Event](#) on page 136.

Generate Events

This option in the Edit menu performs the same function as **Generate Events**. For more information, refer to [Generate Events](#) on page 137.

Send Event

Generates a single selected event, which is then sent to Alarm Monitoring.

Generate Events

Generates multiple events according to the configured frequency settings, and sends them to Alarm Monitoring.

Required Event Generator Files

To use the Event Generator, you will need the following files:

- **EventGeneratorSetupTool.exe**
- **LnlEventGeneratoru.dll**
- (Optional) **EventGenerator.chm**

These files are copied to the <**Windows Configured Program Files Location**>**OnGuard Software Development Kit** directory when the SDK software is installed. Typically, this directory is **C:\Program Files\OnGuard Software Development Kit\EventGenerator**.

You will need to manually copy the files listed above to the OnGuard installation directory, which is typically **C:\Program Files\OnGuard**. Although the **EventGenerator.chm** file is not required for the Event Generator to run, we recommend that you copy this as well, since this contains the online help for the Event Generator application. All of these files are also located on the OnGuard SDK disc in the **program files\OnGuard Software Development Kit\Event Generator** directory.

You must also manually register the **LnlEventGeneratoru.dll**. For more information, refer to [Registering the LnlEventGeneratoru.dll](#) on page 138.

Setting Up the Event Generator

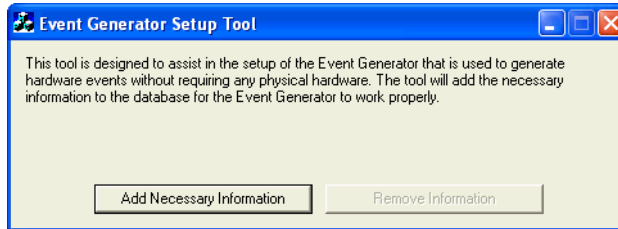
1. Install the OnGuard SDK software.
2. Copy the **EventGeneratorSetupTool.exe**, **LnlEventGeneratoru.dll**, **EventGenerator.chm** files from the Software Development Kit to your hard drive:
 - **For 32-bit operating systems:** Copy from **C:\Program Files\OnGuard Software Development Kit\EventGenerator** directory to **C:\Program files\OnGuard** directory
 - **For 64-bit operating systems:** Copy from **C:\Program Files (x86)\OnGuard Software Development Kit\EventGenerator** directory to **C:\Program Files (x86)\OnGuard** directory

Note: If you receive an information message stating that the **LnlEventGeneratoru.dll** already exists in the **C:\Program Files\OnGuard** directory (or **C:\Program Files (x86)\OnGuard** directory), replace the file.

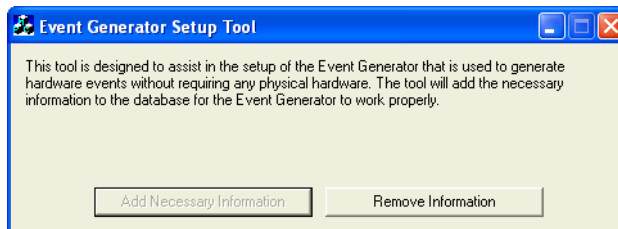
3. Register the **LnlEventGeneratoru.dll**. For more information, refer to [Registering the LnlEventGeneratoru.dll](#) on page 138.
4. In the OnGuard software, add hardware such as access panels, readers, etc. Keep in mind this hardware does not have to be “online”; it might even be hardware that doesn’t really exist.
5. Run the Event Generator Setup Tool. To do this, navigate to the **EventGeneratorSetupTool.exe** file in your OnGuard installation directory (**C:\Program Files\OnGuard** for 32-bit operating systems; **C:\Program Files (x86)\OnGuard** for 64-bit operating systems) and double-click it.


Note: If you receive an error saying that the **LnlFCDBu.dll** file could not be found in the specified path, register the **LnlEventGeneratoru.dll**. For more information, refer to [Registering the LnlEventGeneratoru.dll](#) on page 138.

6. Click [Add Necessary Information].



7. The [Add Necessary Information] button will then become grayed out. At this point, you can close the Event Generator Setup Tool.

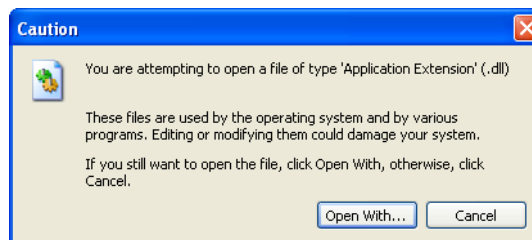


8. Run the Communication Server as an application. To do this:
 - a. Open the Communication Server.
For more information, refer to “Using OnGuard in the Supported Operating Systems” in the Installation Guide.
 - b. Right-click on the  icon in the system tray, and then select **Open Communication Server**. The Communication Server will open in one window, and the Event Generator will open in another window.

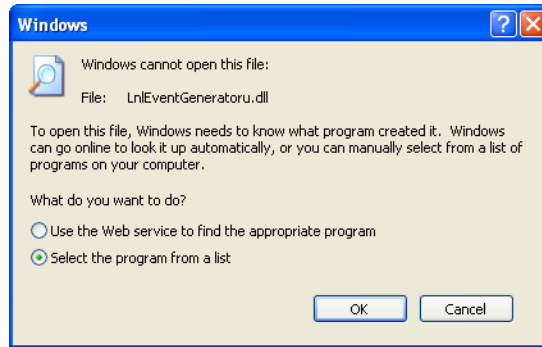
Registering the LnlEventGeneratoru.dll

One way to register the **LnlEventGeneratoru.dll** file is the following:

1. Navigate to the **LnlEventGeneratoru.dll** file in the OnGuard installation directory.
2. Right-click on the file, select **Open With > Choose Program**.
3. A warning message displays, indicating the potential danger of opening dll files. Click [OK].



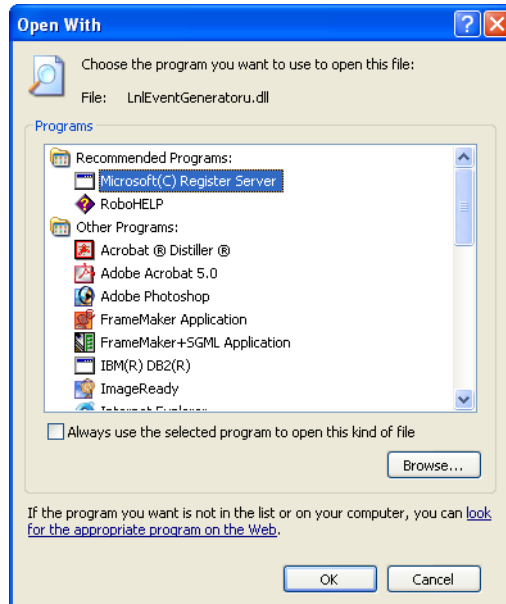
4. Click [Open With...].
5. Select the **Select the program from list** radio button, then click [OK].



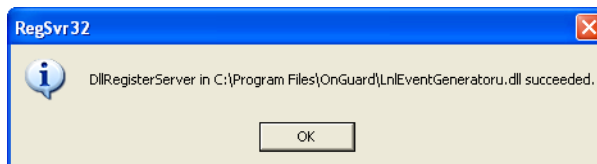
6. The Open With window opens. Click [Browse...], navigate to C:\Windows\system32, and then double-click on the **regsvr32.exe** file.

Note: Run the **regsvr32.exe** file as an administrator. Otherwise, an error message will appear.

7. In the Open With window, Microsoft Register Server will now be highlighted. Click [OK].



The following message is displayed, indicating that the file was successfully registered:



8. The **LnIEventGeneratoru.dll** file is now registered. If you were setting up Event Generator, return to [Setting Up the Event Generator](#) on page 137.

Adding an Event to the Event Generator

A Simple user interface and an Advanced user interface are available for adding events to the Event Generator. Only non-receiver/intrusion events are available in the Simple user interface; both non-receiver/intrusion events and receiver/intrusion events are available in the Advanced user interface.

Adding an Event Using the Simple User Interface

To add a new event to be generated using the Simple user interface:

1. From the **Edit** menu in the Event Generator main window, select **Create Event > Create Event (Simple)**.
2. When the Edit Event (Simple) window appears, select the desired **Event type**. Depending on your selection, the other drop-down lists will be enabled/disabled accordingly.
3. Once you've filled in all necessary items, click [OK].
4. Repeat these steps for all the events you wish to create.

Adding an Event Using the Advanced User Interface

To add a new event to be generated using the Advanced user interface:

1. From the **Edit** menu in the Event Generator main window, select **Create Event > Create Event (Advanced)**.
2. When the Edit Event (Simple) window appears, select the desired **Event type**. Depending on your selection, the other drop-down lists will be enabled/disabled accordingly.
3. Once you've filled in all necessary items, click [OK].
4. Repeat these steps for all the events you wish to create.

Generating Events

Events are generated differently depending on whether you are generating a single event or multiple events.

Generating a Single Event

Select the event you wish to generate from the list of events and then select **Edit > Send Event**. You should see that event in Alarm Monitoring.

Generating Multiple Events

1. In the Event Generator main window, enter a value in the **Number of times** field. This will be the number of times each event in the list is generated.
2. Either fill in the **End delay and In between delay** fields with new values, stay with defaults, or select to use a random time for one or both using the check boxes.
3. You can also select to use random cardholders along with these events, by clicking the **Random badge IDs** check box. To save time you can click [Auto-populate with min and max badge IDs], and then the fields will be automatically filled with the proper numbers from your database.
4. Click **Edit > Generate Events**.

Saving an Event List

After you have completed your event configuration, you can save the event list by doing the following:

1. From the **File** menu, select **Save Events**.
2. Navigate to the location where you wish to save the event list, enter a file name, and then click [Save]. The event list will be saved in a file with the extension EVT.

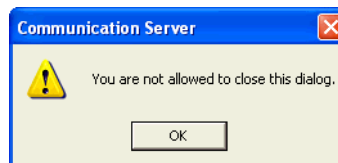
Loading an Event List

To load a previously saved list:

1. From the **File** menu, select **Load Events**.
2. Navigate to the event list that you wish to load, select the EVT file, and then click [Open].

Closing the Event Generator

To close the Event Generator, simply exit the Communication Server. After a short delay, the Event Generator window will close as well. You cannot close the Event Generator manually while the Communication Server is running; if you attempt to do so, the following error message will be displayed:



The following are common problems that you may encounter:

Can't receive cardholder events.

If you can't receive cardholder events, be sure the Linkage Server is running and that the System Options form has the correct setting for where LS Linkage Server is to be running. The **Generate software events** check box must also be selected.

Selecting the Generate software events check box on the System Options form and saving causes an unexpected error.

Try executing the query `new_sw_event.sql` in the `SoftwareEventsAlternate` directory. This can be found in the `DataConduIT TroubleShooting` directory of the DataConduIT documentation file structure.

Permanent consumer.

Permanent consumer only works with machines on a domain; it does not work with workgroup machines.

Multiple threads require multiple user login accounts.

Since DataConduIT uses database connection pooling, the same database connections will be used across multiple threads and will cause unexpected behavior and likely hang or crash at different times of execution. If DataConduIT is being used by separate applications or multiple threads, each application or thread must have its own OnGuard single sign-on account. Use impersonation if necessary to accomplish this.

Not receiving events or messages in a queue when using DataConduIT Message Queue.

Check and make sure the DataConduIT Message Queue service is not set to a local system account and that it has a valid NT account that is linked to OnGuard for single sign-on with the necessary permissions. We recommend testing with an administrator's NT account that is linked to the OnGuard SA account for testing.

Not receiving messages in a Microsoft Message Queue when using DataConduIT Message Queue

Make sure your Microsoft Message Queue is NOT configured to use transactional mode.

Receiving events may not work with Active directory.

Use **domain.exe** located in the **TroubleShooting** directory of the DataConduit documentation file structure to determine if this may be the problem. If the NT4Domain is different from the W2KDomain, then you will need to update the LNL_DIRECTORY.DIR_HOSTNAME to match the NT4Domain. In case this is Oracle, please use all upper case. A sample SQL query to do this is below; it assumes the NT4Domain name is "Lenel" from **domain.exe** and that the directory to be updated is LNL_DIRECTORYID = 1.

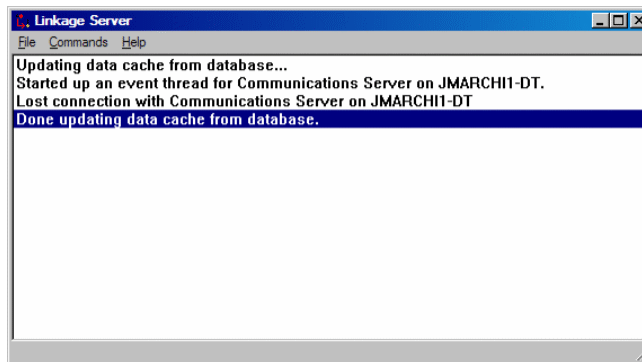
```
update lnl_directory set dir_hostname = 'LENEL' where lnl_directoryid=1
```


Technical Support Pre-Call Checklist

Before calling technical support for anything related to DataConduIT, please complete the following steps.

IMPORTANT: These steps must occur on the server running the DataConduIT and Linkage Server services!

1. Confirm that single sign-on is working. To do this, log into System Administration and confirm you are able to automatically log in.
2. Set the DEBUGLEVEL key. For more information, refer to [Error Logging](#) on page 36.
3. Stop the LS DataConduIT service if it is currently running.
4. Delete or rename the **LenelError.log** in the OnGuard directory.
5. Delete or rename the **DataConduIT.log** which is located in the **C:\WINDOWS\system32\wbem\Logs** directory by default.
6. Confirm that the Linkage Server is successfully running. You can do this by running it as an application - NOT A SERVICE. The application window should look something like this:



7. Start the LS DataConduIT service.
8. Start WMI CIM Studio and go to the **root\OnGuard** directory. WMI CIM studio can be found at <http://www.microsoft.com/downloads/details.aspx?familyid=6430f853-1120-48db-8cc5-f2abdc3ed314&displaylang=en>.
9. Connect to the OnGuard namespace "root\OnGuard" using WMI CIM Studio.

Note: Be sure to use the “Login as current user” option so the currently logged in user is the one being verified and used for single sign-on.

10. Expand the Lnl_Element class and confirm that the Lnl_Panel class exists.
11. Check the **DataConduIT.log**; it should look something like the **SampleDataConduIT.log** file. This file can be found in the **DataConduIT TroubleShooting** directory of the DataConduIT documentation file structure.
12. Confirm the Lnl_Person and Lnl_Badge classes are also under the Lnl_Element class. If you do not see them, DataConduIT cannot successfully connect to the OnGuard software. Please verify once again that single sign-on is working.

In the **DataConduIT Samples\VBDemo** directory of the DataConduIT documentation, there is a file named **SampleDataConduIT.exe**. This file is an application written in Visual Basic 6.0 that can be used to demonstrate some of the capabilities of DataConduIT. The source to this application is also included, but is not heavily commented and is not intended to be used for any kind of development training or sample.

This demo application, including its source code, is provided as is and is in no way supported by Lenel Systems International.

Installing the Visual Basic Demo

The three files necessary for the demo application to run are located in the **DataConduIT Samples\VBDemo** directory in the DataConduIT documentation and are called:

- **SampleDataConduIT.exe** - Visual Basic 6.0 demo program
- **Msflxgrd.ocx** - Visual Basic grid component used by demo program
- **Msvbvm60.dll** - Visual Basic 6.0 runtime dll (may not be necessary if already installed)

There is no installation or setup program. You will have to manually copy these files to a directory and run the **SampleDataConduIT.exe** file either by using Windows Explorer and clicking on the file, or by using the Run command from the Start menu.

Visual Basic Demo Configuration Prerequisites

You must have DataConduIT running and properly configured before using the demo application. This means that the DataConduIT service is running and single sign-on is configured and working with the NT account that will be used while running the demo.

Parts of the demo, such as receiving cardholder events, will require OnGuard to be configured to use the Linkage Server and to also enable “Generate software events” on the General System Options form in the System Options folder in the System Administration application.

You will also need to configure at least one DataConduIT Source from within System Administration. For more information, refer to [Add a DataConduIT Source](#) on page 56.

Using the Visual Basic Demo

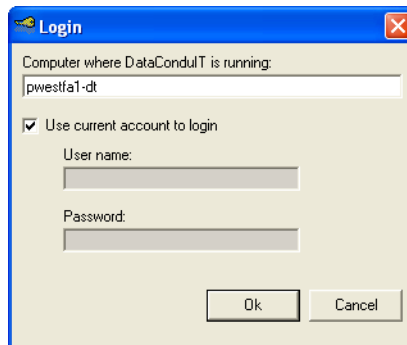
After you log into the Visual Basic demo, you can see demonstrations of the following:

- Sending alarms to the OnGuard software. For more information, refer to [Send Alarms to OnGuard](#) on page 149.
- Receiving alarms from the OnGuard software. For more information, refer to [Receive Alarms from OnGuard](#) on page 149.
- Working with cardholders, including searching for cardholders and detecting changes made to cardholder records. For more information, refer to [Working with Cardholders](#) on page 150.
- Controlling the Active Directory status with the OnGuard software. For more information, refer to [Integrating OnGuard with Active Directory](#) on page 151.

Logging In

To use the Visual Basic demo program you must first log in to it. To do this:

1. Double-click the **SampleDataConduIT.exe** file.
2. From the **File** menu, select **Login**. The Login dialog below appears.



- a. Enter the computer that DataConduIT is running on. This is typically the same machine that the Visual Basic demo program is running on, so the default is the current computer name.
- b. The **Use current account to login** checkbox determines what NT account to use to communicate with DataConduIT. The NT account specified must have a user account in OnGuard and be configured for single sign-on. It must also have user permissions to access DataConduIT.
 - If the checkbox is selected, the current NT account logged in will be used.
 - If the checkbox is deselected, then the **User name** and **Password** fields will be used.

Note: Be sure that the remote enabled permission in WMI Security is enabled for the NT account if you are using this Visual Basic demo from a computer that is not running the DataConduIT service. For more information, refer to [Using DataConduIT from a Remote Computer](#) on page 14.

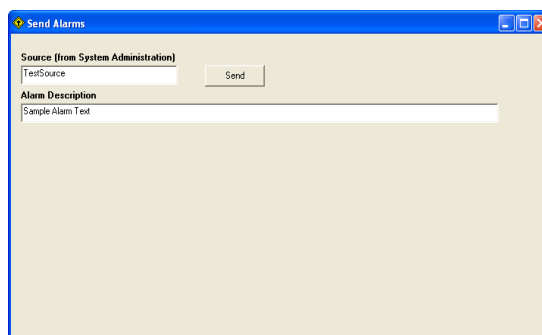
3. Once you have configured what login account to use to access DataConduIT, click [OK] to login.

Send Alarms to OnGuard

The Visual Basic demo program has a “Send Alarms” feature that demonstrates how you can send an alarm from a third party application and have it displayed in Alarm Monitoring. This feature is incredibly powerful because now third party applications can take advantage of all the functionality that OnGuard provides with an alarm, such as executing Global I/O, sending an e-mail, or bringing up video, etc.

To use this part of the demo:

1. Configure at least one DataConduIT Source from within System Administration. For more information, refer to [Add a DataConduIT Source](#) on page 56.
2. Log into the sample DataConduIT application. For more information, refer to [Logging In](#) on page 148.
3. From the **File** menu, select **Send Alarms**. The Send Alarms window opens, as shown.



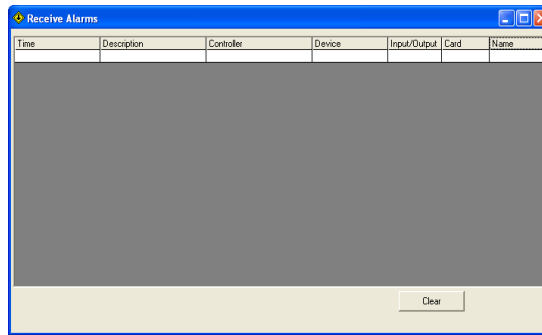
4. Verify that the **Source** field matches the DataConduIT Source in System Administration.
5. Click [Send]. The text specified in the **Alarm Description** field should appear as an alarm in Alarm Monitoring.

Receive Alarms from OnGuard

The Visual Basic demo program has a “Receive Alarms” feature that demonstrates how a third party application can receive alarms that are displayed in the Alarm Monitoring application in real time. This allows customers and third party developers to use alarms/events that occur in OnGuard in their own applications. Developers can use this capability to customize OnGuard even further and add their own custom business rules to the system.

1. Log into the Visual Basic demo application. For more information, refer to [Logging In](#) on page 148.
2. From the **File** menu, select **Receive Alarms**. The Receive Alarms window opens, as shown. When alarms come into Alarm Monitoring, the same alarm should appear in this window.

Note: The [Clear] button clears the events in the list.



Working with Cardholders

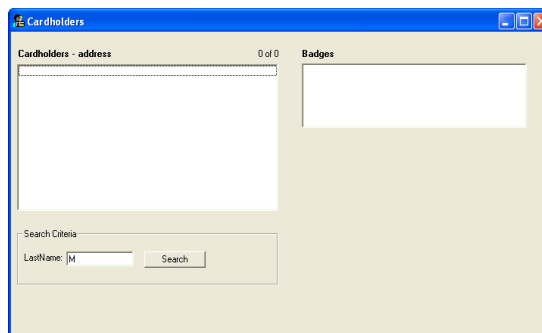
The Visual Basic demo program has a “Cardholders” feature that demonstrates two capabilities: searching cardholders, and receiving changes from OnGuard when changes happen to a cardholder. This capability is very important to third party developers for integrating cardholder information across multiple systems such as Active Directory or Human Resources Departments. Keep in mind that although the capability is not demonstrated, DataConduIT has the ability to change access levels and modify badges. This capability can be used to develop business rules to have an employee’s physical access controlled from another application such as Active Directory. This way you can combine physical and logical access as well as other types of information like vending, membership, library privileges, from other systems. Terminating an employee in one system can revoke all their privileges across several systems.

To work with cardholders:

1. From the **Administration** menu in System Administration, select **System Options**. Make sure that the following settings have been set:
 - In the **Linkage Server host** drop-down listbox, the workstation where the Linkage Server is running must be specified. The name specified must be the workstation’s NetBIOS name. (The NetBIOS name is configured when Windows networking is installed/configured.)
 - The **Generate software events** checkbox must be selected.

Note: If you change these settings, it is recommended that you restart the Linkage Server and the DataConduIT Server services. You will also then have to log back into the Visual Basic demo application.

2. Log into the Visual Basic demo application. For more information, refer to [Logging In](#) on page 148.
3. From the **File** menu, select **Cardholders**. The Cardholders window opens, as shown.



Searching for Cardholders

Use the search criteria data at the bottom to do some adhoc searches by Last Name. Clicking [Search] will find all cardholders whose last name begin with the letters typed into the **LastName** field. These cardholders' last name, first name and address (if any) will be displayed in the Cardholders - address listing window. Selecting a cardholder from the list will automatically find the cardholder's badges and display their badge ID, activate date, and deactivate date in the Badges listing window.

Modifying Cardholders

If you go into OnGuard and modify a cardholder's first name or address, these changes will be reflected automatically in the Cardholder - address listing window within a few seconds. Remember, this feature is only available if you have specified a Linkage Server and selected the **Generate software events** checkbox on the General System Options form in the System Options folder in System Administration.

Integrating OnGuard with Active Directory

A typical use of DataConduIT is integrating physical access to logical access by combining OnGuard access control with an IT department's Active Directory. This part of the demo program shows this integration by adding, automatically creating, or modifying an account for any cardholder that is created or modified in the OnGuard software. This demonstration is strictly a one way integration and shows OnGuard controlling the Active Directory status. Customers who want Active Directory to control OnGuard can use DataConduIT to make the changes to OnGuard data.

To see a demonstration of OnGuard controlling the Active directory status:

1. Log into the Visual Basic demo application. For more information, refer to [Logging In](#) on page 148.
2. From the **File** menu, select **Active Directory Integration**. The Active Directory Integration window opens, as shown.



3. Before continuing, verify that the account you used to log into the Visual Basic demo program has rights and privileges for adding directory/NT accounts.
4. In the **Directory Name** field, enter the name of the active directory or the NT based computer that you intend on adding accounts to.
5. The Operations History listing window will display a read-only information list of each operation made through this window. Before using OnGuard cardholders, we recommend that you test the ability to add/modify user accounts to the directory. To do this:
 - a. Type in a last name and a first name in the respective fields.
 - b. Be sure the **Account Disabled** checkbox is NOT selected.
 - c. Click [Create/Update User Account].

- d. Confirm an NT account with the username of Lastname combined with Firstname was indeed added to the system. Once this has indeed been confirmed, you should now be able to go to the Cardholders form in OnGuard and add cardholders to the system.
6. As cardholders are added in OnGuard, the demo program will detect a change in OnGuard, populate the **Lastname** and **Firstname** fields, and automatically execute the [Create/Update User Account] operation. The **Account Disabled** checkbox will be automatically set based on the operation performed in the OnGuard software. Deactivating an active badge, or deleting an active badge will disable the account. Adding an active badge will activate an account.

Note: There is no option to delete an NT account in this Visual Basic demo, so you will have to manually remove the accounts using Active Directory Users and Computers or using Computer Management depending on the type of system being controlled.

Index

A		B	
Abbreviations	10	Badges	23
Acronyms	10	C	
ACS.INI file	38	Caching user credentials	14
Active Script Event Consumer	28	Cancel() method	26
Add		Cardholders	23
DataConduIT Device	59	Changing the database connection pool time	37
DataConduIT message queue	51	Class definition	10
DataConduIT Source	56	Classes	
DataConduIT Sub-Device	61	association	105
Event to the Event Generator	140	data	71
Adding objects	21	event	109
ADsSecurity.dll	10	Class-specific features and limitations	23
Alarms		Client definition	10
Test Event From DataConduIT	31	Closing the Event Generator	141
using DataConduIT to send	31	Command and control classes and methods	
ASEC	28	LnI_AlarmInput	115
Association classes	105	LnI_AlarmOutput	115
LnI_AccessLevelGroupAssignment ..	105	LnI_AlarmPanel	115
LnI_BadgeOwner	105	LnI_Input	116
LnI_CardholderAccount	106	LnI_IntrusionArea	116
LnI_CardholderBadge	106	LnI_IntrusionDoor	117
LnI_CardholderMultimediaObject	106	LnI_IntrusionOutput	118
LnI_DirectoryAccount	106	LnI_IntrusionZone	118
LnI_MultimediaObjectOwner	107	LnI_IntrusionZoneOutput	118
LnI_PersonAccount	107	LnI_OffBoardRelay	118
LnI_ReaderEntersArea	107	LnI_OnBoardRelay	119
LnI_ReaderExitsArea	108	LnI_Output	119
LnI_SegmentGroupMember	108	LnI_Panel	120
LnI_VisitorAccount	108	LnI_Reader	121
LnI_VisitorMultimediaObject	109	LnI_ReaderInput	123
Authentication	13	LnI_ReaderInput1	123
Authorization	14	LnI_ReaderInput2	123
		LnI_ReaderOutput	123

- LnI_ReaderOutput1 123
- LnI_ReaderOutput2 123
- Connecting to DataConduit 19
- D**
- Data classes 71
 - LnI_AccessGroup 71
 - LnI_AccessLevel 72
 - LnI_AccessLevelAssignment 72
 - LnI_AccessLevelReaderAssignment 73
 - LnI_Account 73
 - LnI_AlarmDefinition 74
 - LnI_Area 74
 - LnI_Badge 76, 77
 - LnI_BadgeLastLocation 78
 - LnI_BadgeProperties 79
 - LnI_BadgeType 80
 - LnI_Camera 80
 - LnI_CameraGroup 81
 - LnI_CameraGroupCameraLink 81
 - LnI_Cardholder 82
 - LnI_DataConduitManager 82
 - LnI_Directory 83
 - LnI_Element 83
 - LnI_EventAlarmDefinitionLink 84
 - LnI_EventParameter 84
 - LnI_EventSubtypeDefinition 84
 - LnI_EventSubtypeParameterLink 85
 - LnI_EventType 85
 - LnI_Holiday 86
 - LnI_HolidayType 86
 - LnI_HolidayTypeLink 87
 - LnI_IncomingEvent 87
 - LnI_LoggedEvent 90
 - LnI_LogicalSystemAccount 91
 - LnI_MobileVerify 92
 - LnI_MonitoringZone 93
 - LnI_MonitoringZoneCameraLink 93
 - LnI_MultimediaObject 94
 - LnI_Panel 94
 - LnI_Person 95
 - LnI_Reader 95
 - LnI_Segment 96
 - LnI_SegmentGroup 96
 - LnI_SegmentUnit 96
 - LnI_Timezone 97
 - LnI_TimezoneInterval 97
 - LnI_User 98
 - LnI_UserAccount 99
 - LnI_UserPermissionDeviceGroupLink 101
 - LnI_UserPermissionGroup 99, 101
 - LnI_UserSecondarySegment 102
 - LnI_Visit 102
 - LnI_VisitEmailRecipient 103
 - LnI_Visitor 104
 - user-defined value lists 104
- DATABASESETIMEOUT registry setting 37
- DataConduit
 - connecting to 19
 - description 9
 - error log 36
 - installing 13
 - integration scenarios 9
 - overview of functions 16
 - Remote Enable permission 15
 - running on Linkage Server 13
 - samples 10
 - stopping and restarting the DataConduit service 38
 - user credential caching 14
 - using for data access 19
 - using from a remote computer 14
 - using permanent event consumers with 28
 - using to receiving events 25
 - viewing DataConduit classes with the Microsoft WMI SDK 15
- DataConduit Devices form
 - field table 59
 - procedures 59
- DataConduit Message Queues form
 - Advanced sub-tab 50
 - General sub-tab 48
 - procedures 51
 - Settings sub-tab 49
- DataConduit Sources
 - licenses required 54
 - user permissions required 55
- DataConduit Sources form
 - field table 56
 - procedures 56
- DataConduit Sub-Devices form
 - field table 61
 - procedures 61
- DataConduit.log file 36
- DebugFile registry setting 36
- DebugLevel registry setting 36
- Definitions 10
- Delete
 - DataConduit Device 59
 - DataConduit message queue 52
 - DataConduit Source 57
 - DataConduit Sub-Device 61
- Deleting objects 23
- Directory 14
- Directory accounts 24
- Documentation
 - contents 10
 - Microsoft Scripting Technologies 11
 - Microsoft WMI 11
 - prerequisites 10
- Documentation prerequisites
 - JScript 10
 - VBScript 10
 - Windows Management Instrumentation 10

E	
Error logging	36
Event classes	109
LnI_AccessEvent	109
LnI_Alarm	110
LnI_Event	111
LnI_FireEvent	111
LnI_FunctionExecEvent	111
LnI_IntercomEvent	112
LnI_OtherSecurityEvent	112
LnI_SecurityEvent	112
LnI_StatusChangeEvent	113
LnI_TransmitterEvent	113
LnI_VideoEvent	114
Event Generator	
add an event to the Event Generator ..	140
closing	141
generating a single event	140
generating events	140
generating multiple events	140
main window	129
menus	136
saving an event list	141
setting up	137
Events	
add an event to the Event Generator ..	140
event classes	109
Generate software events checkbox	14
generating	140
generating multiple	140
generating single	140
hardware	25
loading an event list	141
receiving	14, 27, 28
receiving software events	14
registering to receive	26, 27
saving an event list	141
software	25
using DataConduIT to receive	25
using permanent event consumers with DataConduIT	28
ExecNotificationQueryAsync() method	26
G	
Generate software events checkbox	14
Generating a single event	140
Generating Access Granted and Access Denied events	89
Generating events	140
Generating multiple events	140
Getting started	13
H	
Hardware event definition	11
Hardware events	25
I	
Install	13
Installing	
Visual Basic Demo	147
Integrating OnGuard with Active Directory ...	151
Introduction	9
J	
JScript	10
L	
Linkage Server	13
Lists	104
LnI_AccessEvent	109
LnI_AccessGroup	71
LnI_AccessLevel	72
LnI_AccessLevelAssignment	72
LnI_AccessLevelGroupAssignment	105
LnI_AccessLevelReaderAssignment	73
LnI_Account	73
LnI_Alarm	110
LnI_AlarmDefinition	74
LnI_AlarmInput	115
LnI_AlarmOutput	115
LnI_AlarmPanel	115
LnI_Area	74
LnI_AuthenticationMode	75
LnI_Badge	76
LnI_BadgeFIPS201	77
LnI_BadgeLastLocation	78
LnI_BadgeOwner	105
LnI_BadgeProperties	79
LnI_BadgeType	80
LnI_Camera	80
LnI_CameraGroup	81
LnI_CameraGroupCameraLink	81
LnI_Cardholder	82
LnI_CardholderAccount	106
LnI_CardholderBadge	106
LnI_CardholderMultimediaObject	106
LnI_DataConduITManager	82
LnI_Directory	83
LnI_DirectoryAccount	106
LnI_Element	83
LnI_Event	111
LnI_EventAlarmDefinitionLink	84
LnI_EventParameter	84
LnI_EventSubtypeDefinition	84
LnI_EventSubtypeParameterLink	85
LnI_EventType	85
LnI_FireEvent	111
LnI_FunctionExecEvent	111
LnI_Holiday	86
LnI_HolidayType	86
LnI_HolidayTypeLink	87
LnI_IncomingEvent	17, 31, 32, 87

Lnl_Input	116
Lnl_IntercomEvent	112
Lnl_IntrusionArea	116
Lnl_IntrusionDoor	117
Lnl_IntrusionOutput	118
Lnl_IntrusionZone	118
Lnl_IntrusionZoneOutput	118
Lnl_LoggedEvent	90
Lnl_LogicalSystemAccount	91
Lnl_MobileVerify	33, 92
Lnl_MonitoringZone	93
Lnl_MonitoringZoneCameraLink	93
Lnl_MultimediaObject	94
Lnl_MultimediaObjectOwner	107
Lnl_OffBoardRelay	118
Lnl_OnBoardRelay	119
Lnl_OtherSecurityEvent	112
Lnl_Output	119
Lnl_Panel	94, 120
Lnl_Person	95
Lnl_PersonAccount	107
Lnl_Reader	95, 121
Lnl_ReaderEntersArea	107
Lnl_ReaderExitsArea	108
Lnl_ReaderInput	123
Lnl_ReaderInput1	123
Lnl_ReaderInput2	123
Lnl_ReaderOutput	123
Lnl_ReaderOutput1	123
Lnl_ReaderOutput2	123
Lnl_SecurityEvent	112
Lnl_Segment	96
Lnl_SegmentGroup	96
Lnl_SegmentGroupMember	108
Lnl_SegmentUnit	96
Lnl_StatusChangeEvent	113
Lnl_Timezone	97
Lnl_TimezoneInterval	97
Lnl_TransmitterEvent	113
Lnl_User	98
Lnl_UserAccount	99
Lnl_UserFieldPermissionGroup	101
Lnl_UserPermissionDeviceGroupLink	101
Lnl_UserPermissionGroup	99, 101
Lnl_UserSecondarySegment	102
Lnl_VideoEvent	114
Lnl_Visit	102
Lnl_VisitEmailRecipient	103
Lnl_Visitor	104
Lnl_VisitorAccount	108
Lnl_VisitorMultimediaObject	109
LnlEventGeneratoru.dll	
location	137
registering	137
Loading an event list	141
Log for errors	36
Logging in	148
LS Linkage Server	14
M	
Menus for Event Generator	136
MobileVerify	
working with	33
Modify	
cardholders	151
DataConduIT Device	59
DataConduIT message queue	52
DataConduIT Source	57
DataConduIT Sub-Device	61
objects	22
Multimedia objects	24
N	
Namespace definition	11
O	
Object/instance definition	11
Objects	
adding	21
deleting	23
modifying	22
searching for	19
OPC Connections	
folder	63
Overview	
DataConduIT functions	16
P	
Permissions - Remote Enable	15
Person definition	11
PIN code	24
Pre-call checklist	145
PreviousInstance	28
Problems	143
Procedures	
adding objects	21
changing the database connection pool	
time	37
deleting objects	23
modifying objects	22
receiving error information from	
DataConduIT	35
receiving events	14, 27, 28
registering to receive events	26, 27
searching for objects	19
stopping and restarting the DataConduIT	
service	38
using DataConduIT from a remote	
computer	14
using DataConduIT to receive events	25
using permanent event consumers with	
DataConduIT	28
viewing DataConduIT classes with the	
Microsoft WMI SDK	15
Property qualifiers used in DataConduIT	127

-
- R**
- Receiving
 - alarms from OnGuard 149
 - error information from DataConduIT . 35
 - events 14, 27, 28
 - Reference 71
 - References and applicable documents 11
 - Registering the LnlEventGeneratoru.dll ... 137
 - Registering to receive events 26, 27
 - Registry settings
 - DATABASETIMEOUT 37
 - DebugFile 36
 - DebugLevel 36
 - Remote Enable permission 15
 - Required
 - Visual Basic Demo files 147
- S**
- Sample code
 - add a cardholder 22
 - common software event queries 27
 - connect to the namespace used by
DataConduIT (JScript) 19
 - delete an object 23
 - delete an object in DataConduIT 23
 - example of a simple temporary event
consumer 25
 - find all active badges that are APB exempt
(WQL query) 20
 - find all directories with a specified
hostname (WQL query) 20
 - find all people whose last name is not
"Lake" (WQL Query) 20
 - find all readers (WQL query) 20
 - get a cardholder if you know the
cardholder's ID 21
 - hardware event queries 26
 - modifying objects 22
 - multiple key properties in the class 21
 - print first and last names of all cardholders
in OnGuard 20
 - retrieve error information 31, 35
 - use a WQL query with the ExecQuery()
method 21
 - Samples 10
 - Saving an event list 141
 - SDK definition 11
 - Searching for
 - cardholders 151
 - objects 19
 - Security identifier 24
 - Send alarms to OnGuard 149
 - SendIncomingEvent 31
 - Setting up the Event Generator 137
 - Single sign-on 13
 - SINK_OnObjectReady() function 26
 - Software event definition 11
 - Software events 25
 - SSO 13
 - Stopping and restarting the DataConduIT
service 38
 - SWbemLastError object 35
 - SWbemServices 19
- T**
- TargetInstance 28
 - Technical support pre-call checklist 145
 - Test Event From DataConduIT alarm 31
 - Timeout value for the database connection pool
37
 - Troubleshooting and advanced options 35
 - Tuning parameters 37
- U**
- UDF 127
 - User account 14
 - User-defined fields 127
 - User-defined list values 24
 - User-defined value lists 104
 - Using
 - permanent event consumers with
DataConduIT 28
 - Visual Basic Demo 148
 - Using DataConduIT
 - for data access 19
 - from a remote computer 14
 - to receive events 25
 - to send alarms to OnGuard 31
- V**
- VBScript 10
 - Viewing DataConduIT classes with the
Microsoft WMI SDK 15
 - Visitors 23
 - Visits 24
 - Visual Basic Demo
 - configuration prerequisites 147
 - installing 147
 - integrating OnGuard with Active
Directory 151
 - logging in 148
 - modify cardholders 151
 - receive alarms from OnGuard 149
 - required files 147
 - search for cardholders 151
 - send alarms to OnGuard 149
 - using 148
 - work with cardholders 150
 - Visual Basic Demo configuration prerequisites
147
- W**
- Windows Management Instrumentation
definition 11
-

WMI	
definition	11
Working with	
cardholders	150
MobileVerify	33

Lenel Systems International, Inc.
1212 Pittsford-Victor Road
Pittsford, New York 14534 USA
Tel 866.788.5095 Fax 585.248.9185
www.lenel.com
docfeedback@lenel.com

