OnGuard®

OpenAccess User Guide

LenelS2 OnGuard® 8.3 OpenAccess User Guide
This guide is item number DOC-1057-EN-US, revision 14.046 September 2024.

SOURCES. THE ABILITY OF SOFTWARE, HARDWARE AND RELATED SERVICES TO WORK PROPERLY DEPENDS ON A NUMBER OF PRODUCTS AND SERVICES MADE AVAILABLE BY THIRD PARTIES OVER WHICH LENELS2 HAS NO CONTROL INCLUDING, BUT NOT LIMITED TO, INTERNET, CELLULAR AND LANDLINE CONNECTIVITY; MOBILE DEVICE AND RELATED OPERATING SYSTEM COMPATABILITY; OR PROPER INSTALLATION, CONFIGURATION AND MAINTENANCE OF AUTHORIZED HARDWARE AND OTHER SOFTWARE.

LENELS2 MAY MAKE CERTAIN BIOMETRIC CAPABILITIES (E.G., FINGERPRINT, VOICE PRINT, FACIAL RECOGNITION, ETC.), DATA RECORDING CAPABILITIES (E.G., VOICE RECORDING), AND/OR DATA/INFORMATION RECOGNITION AND TRANSLATION CAPABILITIES AVAILABLE IN PRODUCTS LENELS2 MANUFACTURES AND/OR RESELLS. LENELS2 DOES NOT CONTROL THE CONDITIONS AND METHODS OF USE OF PRODUCTS IT MANUFACTURES AND/OR RESELLS. THE END-USER AND/OR INSTALLER AND/OR RESELLER/DISTRIBUTOR ACT AS CONTROLLER OF THE DATA RESULTING FROM USE OF THESE PRODUCTS, INCLUDING ANY RESULTING PERSONALLY IDENTIFIABLE INFORMATION OR PRIVATE DATA, AND ARE SOLELY RESPONSIBLE TO ENSURE THAT ANY PARTICULAR INSTALLATION AND USE OF PRODUCTS COMPLY WITH ALL APPLICABLE PRIVACY AND OTHER LAWS, INCLUDING ANY REQUIREMENT TO OBTAIN CONSENT. THE CAPABILITY OR USE OF ANY PRODUCTS MANUFACTURED OR SOLD BY LENELS2 TO RECORD CONSENT SHALL NOT BE SUBSTITUTED FOR THE CONTROLLER'S OBLIGATION TO INDEPENDENTLY DETERMINE WHETHER CONSENT IS REQUIRED, NOR SHALL SUCH CAPABILITY OR USE SHIFT ANY OBLIGATION TO OBTAIN ANY REQUIRED CONSENT TO LENELS2.

FOR MORE INFORMATION ON PRODUCT WARNINGS AND DISCLAIMERS, SEE THE "LENELS2 PRODUCT WARNINGS AND DISCLAIMERS" KNOWLEDGE BASE ARTICLE IN THE LENELS2 KNOWLEDGE BASE. THIS INFORMATION IS SUBJECT TO CHANGE WITHOUT NOTICE.

Table of Contents

CHAPTER 1

Revision History

The following changes were made to the OpenAccess API in support of OnGuard 8.3:

- Updated SignalR server and client component version requirement from 2.4.0 or 2.4.1 to version 2.4.3. For more information, refer to Event API Reference on page 193.
- Added **PanelName** and **SegmentID** properties to **Lnl_Reader**. For more information, refer to Lnl_Reader on page 294.
- Added **InputID**, **ReaderName**, and **SegmentID** properties to **Lnl_ReaderInput**. For more information, refer to Lnl_ReaderInput on page 300, Lnl_ReaderInput1 on page 301, and Lnl_ReaderInput2 on page 303.
- Added **OutputID**, **ReaderName**, and **SegmentID** properties to **Lnl_ReaderOutput**. For more information, refer to Lnl_ReaderOutput on page 304, Lnl_ReaderOutput1 on page 305, and Lnl_ReaderOutput2 on page 307.
- Added **SegmentID** property to **Lnl_AlarmPanel**. For more information, refer to Lnl_AlarmPanel on page 239.
- Added **device groups** functionality. For more information, refer to Device Groups on page 184.
- Modified **Lnl_Timezone** and **Lnl_TimezoneInterval** properties. For more information, refer to Lnl_Timezone on page 312 and Lnl_TimezoneInterval on page 313.
- Added **LockdownGroupState**, **LockDownGroupActivate**, **LockDownGroupDeactivate**, and **GetLockDownGroupState** to **Lnl_DeviceGroup.** For more information, refer to Lnl_DeviceGroup on page 256.
- Updated **workstation** property to badge printer calls. For more information, refer to add badge print_request on page 96, get badge_printers on page 101, post badge printers on page 104, put badge printers on page 105, and delete badge printers on page 106.
- Added **DoubleCardAuthority** and **GlobalIntrusionCommandAuthority** properties to Lnl_AccessLevel on page 224.
- Updated holiday property details. For more information, refer to Lnl_Holiday on page 263, Lnl_HolidayType on page 264, and Lnl_HolidayTypeLink on page 264.
- Added **Lnl_AccessPanelUser** data class. For more information, refer to Lnl_AccessPanelUser on page 228.
- Added **DownloadTLSCertificate** and **DownloadPeerCertificate** methods to **Lnl_Panel**. For more information, refer to Lnl_Panel on page 286.
- Added **has_only_inactive_badges** and **last_activity_filter** to **get cardholders**. For more information, refer to get cardholders on page 113.

- Added **SegmentName** property to **Lnl_Holiday** and **Lnl_Timezone**. For more information, refer to Lnl_Holiday on page 263 and Lnl_Timezone on page 312.
- Added **BiometricBodyPart** and **AcceptanceThreshold** to **Lnl_MultimediaObject**. For more information, refer to Lnl_MultimediaObject on page 281.
- Added **is_default** property to the response of badge printer calls. For more information, refer to get badge_printers on page 101, post badge printers on page 104, put badge printers on page 105, and delete badge printers on page 106.
- Added **bulk export** functionality. For more information, refer to get instances (bulk export) additional parameters on page 94.
- Added **lockdown** and **lockdown deactivation** information to **Lnl_Reader**. For more information, refer to Lnl_Reader on page 294.
- Added **ExtendedOptions** to **Lnl_Segment**. For more information, refer to Lnl_Segment on page 311. Also added **extended_options** to **get segmentation_settings** response. For more information, refer to get segmentation_settings on page 165.
- Updated **last_activity_filter** in **get cardholders**. For more information, refer to get cardholders on page 113.
- Added **bulk export** parameter to **get instances** response. For more information, refer to get instances on page 91.
- Removed **DownloadTLSCertificate()** and **DownloadPeerCertificate()** methods, and added **DownloadPanelCertificates()** method. For more information, refer to Lnl_Panel on page 286.
- Added **get access_panel_with_certs_and_users** and response parameters. For more information, refer to get access panel with certs and users on page 189.
- Added **get marketplace** call. For more information, refer to get marketplace on page 191.
- The **get visitevent_status** endpoint is extended with the optional **visitor_id_list** parameter to allow searching only statuses of those visit events in which any of the selected visitor(s) take part. For more information, refer to get visitevent_status on page 123.
- **Lnl_Holiday** class has been extended to support add, modify and delete operations. For more information, refer to Lnl_Holiday on page 263.
- **Lnl_TimeZone** class has been extended to support add, modify and delete operations. For more information, refer to Lnl_Timezone on page 312.
- **Lnl_TimezoneInterval** class has been extended to support add, modify and delete operations. For more information, refer to Lnl_TimezoneInterval on page 313.
- **Lnl_HolidayType** was extended to support modify operations. For more information, refer to Lnl_HolidayType on page 264.
- **Lnl_HolidayTypeLink** was extended to support add and delete operations. For more information, refer to Lnl_HolidayTypeLink on page 264.
- **GET logged_in_user** permission map was extended to expose all user permissions. For more information, refer to get logged_in_user on page 133.
- Added **SegmentID** property to **get instance Lnl_AccessLevel** and **get /user/{id}/ managed_access_levels** API calls. For more information, refer to Lnl_AccessLevel on page 224 and get user managed_access_levels on page 134.

CHAPTER 2

Introduction

This document provides information about the LS OpenAccess service that can be used to manage OnGuard and to integrate it with external systems such as IT systems. The LS OpenAccess service is the API into OnGuard, and provides access to ID management data, hardware events, software events, and access control events when changes are made to cardholders and their credentials.

Note: The system is bundled with third-party software including Open Source that is subject to separate license agreements. These licenses and source code, where applicable, are available on the LenelS2 website or via a media distribution. Refer to the OnGuard Attributions document included in the **program files\doc** directory with this product.

The REST proxy that is part of the LS OpenAccess service allows you to create a client against a REST API to OnGuard through NGINX as the web service which abstracts the Advanced Message Queuing Protocol (AMQP) language. The LS Web Service is the service hosting NGINX. OpenAccess requires the LS Message Broker service, and Secure Socket Layer (SSL) must be enabled. The client uses the REST proxy to communicate with the LS OpenAccess service.

Note: If using OpenAccess or Enterprise in a cluster environment, and using the default installed certificates, the certificates may need to be reissued on the machine running the LS Message Broker service. For instructions, refer to "Manually Issue an SSL Certificate" in the *NEC ExpressCluster X R3 Installation Guide* or the *Using Microsoft Cluster Services with OnGuard* guide. Also refer to the "OnGuard and the Use of Certificates" appendix in the *OnGuard Installation Guide*.

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

Notes: OpenAccess is not intended to perform large batch processing tasks. If performing batch processing, you will achieve improved performance by using the DataExchange Server instead of OpenAccess.

To achieve the best performance in OpenAccess when sending events, confirm that the Linkage Server is configured (the machine host name is set in **System Administration > Administration > System Options > Linkage Server host**), and the service is running.

There are some minor differences in behaviors between OpenAccess and legacy thick clients such as Alarm Monitoring and System Administration. For more information, refer to Expectations and Behaviors of OpenAccess on page 18.

- 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.

Expectations and Behaviors of OpenAccess

For applications that are built on the OpenAccess platform, there are minor differences in behavior between the web applications and existing client applications such as OnGuard Alarm Monitoring or OnGuard System Administration. The following sections describe these differences. Use this information in addition to Troubleshooting on page 349 to diagnose OpenAccess-related issues that may occur.

Confirming the Installed Version of OnGuard

Verify that OpenAccess and its dependent services are configured correctly by confirming that the following URL can be accessed to retrieve the installed OnGuard version:

```
https://<servername>:8080/api/openaccess/version?version=1.0
```

where `<servername>` is the name of the OnGuard Server where Open Access is running.

The expected result should be:

```
{"product_name":"OnGuard 7.x Enterprise
(Standard)","product_version":"7.x.xxx.x"}
```

If this test fails, refer to refer to Chapter 10: Troubleshooting on page 349.

Stopping and Restarting the Services

Stopping and restarting the services is generally unnecessary. The services are installed with their properties configured to start automatically. However, if there is an issue with a service, refer to Stopping and Restarting the Services on page 32 for more information.

Note: When enabling segmentation in an OnGuard system, or when enabling or disabling an OnGuard system's segmentation options, restart the OpenAccess service to prevent unexpected behavior. It is *not* necessary to restart the OpenAccess service when changing a user's segment access, as segment assignments are refreshed every 1 minute by default. This refresh interval is configured using the permission_refresh_interval property. For more information, refer to Caching Properties on page 22.

Authorization

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

OpenAccess caches user credentials and segments for 1 minute by default. This is done for performance reasons. Therefore, if a user is using an application built on the OpenAccess platform and that user's permissions or segments change, the user will continue to have his old permissions until the 1-minute timeout is reached.

The Event Context Provider service, which is responsible for sending events matching event subscriptions, caches user credentials and segments for 15 minutes by default. OnGuard Monitor requires the Event Context Provider service.

User-Defined Fields

The user-defined field schema is updated every 5 minutes. If a user changes, adds, or deletes a property using FormsDesigner, it will take up to 5 minutes for the change to appear in the LS OpenAccess service. For more information, refer to User-Defined Fields on page 47.

IMPORTANT: It is vital to understand that OnGuard default fields and data types can be modified or deleted by OnGuard customers using the FormsDesigner application. Therefore, it is important that OAAP Members make their integration configurable to handle any changes or deletions the customer may have made. Currently, the forms that can be edited using FormsDesigner are Cardholder Person Type, Visitor Person Type, Badge, Visits and Assets.

OpenAccess and Brute Force Attack Protection

OpenAccess protects users against Brute Force Attacks, where an attacker attempts to log into a user account repeatedly in an attempt to determine the password. The number of attempts and duration of lockout can be configured using the **put password policy settings** call. For more information, refer to put password policy settings on page 163.

For more information about brute force attacks, refer to OpenAccess and Brute Force Attack Protection on page 55.

Using OpenAccess to Issue Mobile Badges

If you are using an application built on the OpenAccess platform to issue mobile badges and are behind a network proxy, an error might occur when issuing or managing mobile credentials. To resolve this error, on the server where the LS OpenAccess service is running, change the logon account for the LS OpenAccess service from Local System to a user whose account has the correct proxy settings configured. For more information, refer to get badge mobile_devices on page 98.

Authenticated Token and Inactivity Timeouts

When using an application built on the OpenAccess platform, there are two properties that terminate authenticated sessions.

The **authenticated token timeout** property terminates an authenticated session after a predetermined, user-configurable time period. The default value for this time period is 8 hours. During this period, if there is no activity from the authenticated user within a predetermined, user-

configurable time period (default of 15 minutes), the **inactivity_timeout_in_minutes** property of the **password policy settings** call terminates the authenticated session.

These properties are system-wide, which means every browser-based client of that OpenAccess Server will have the same timeout settings applied. In an Enterprise system, these properties can be configured at each region to support local usage and regulation of the applications.

Notes: The authenticated token inactivity timeout property only applies if there are no cameras added to OnGuard Surveillance. If there is at least one camera added, OnGuard Surveillance polls OnGuard every 30 seconds, thus negating the no-activity period that this property monitors.

The inactivity timeout only affects browser-based clients that use OpenAccess. The inactivity timeout does not affect OnGuard thick clients.

The **authenticated token timeout** property can be configured in the **openaccess.ini** file. For more information about the **openaccess.ini** file, refer to OpenAccess Custom Configuration on page 21. For more information on the **inactivity_timeout_in_minutes** property, refer to get password policy settings on page 161 and put password policy settings on page 163.

OpenAccess Custom Configuration

OpenAccess can be configured by using an optional **openaccess.ini** file. This file is not provided upon installation of OpenAccess or the OnGuard software. Use a text editor to create an INI file in ***C:\ProgramData\Lnl***. Properties in the **openaccess.ini** file should remain unchanged. However, if a property is modified, restart the LS OpenAccess service in order for changes to take effect.

INI files typically organize properties into sections. For example, the following is an example of how the `authenticated_token_timeout` property should be set in the authentication section:

```
[authentication]
authenticated_token_timeout=12
```

Refer to the following sections for configurable properties.

Note: If the selected value cannot be parsed, the default value is used. If the property supports a range and the value specified is below the supported minimum value, the minimum value is used. Similarly, if the value specified is above the supported maximum value, the maximum value is used

Authentication Property

Property	Section	Default	Range	Description
authenticated_token_ timeout	authentication	8	1 to 24	The authenticated token timeout, in hours.

Caching Properties

Note: Changing the caching properties to be more frequent than the default values will negatively affect performance. It is recommended to not modify the caching properties.

Property	Section	Default	Range	Description
hardware_configuration_refresh_interval	cache	300	1 to 3600	The hardware configuration refresh interval, in seconds.
hardware_status_thread_shutdown_interval	cache	15	1 to <*value of panel_status_refresh_interval*>	The amount of time after which a hardware status thread will shut down and stop waiting for an asynchronous status response from a particular panel, in seconds.
panel_status_refresh_interval	cache	60	5 to 3600	The time to check if cached panel's status is stale, in seconds. It is stale if the panel's status was read <*panel_status_refresh_interval*> seconds or more ago. When the UpdateHardwareStatus method is called, the OpenAccess service re-reads the panel's status if it is stale.
password_policy_setting_refresh_interval	cache	60	1 to 3600	The password policy setting refresh interval for an Enterprise system, in seconds.
permission_refresh_interval	cache	1	1 to 1440	The time in minutes to check if the following cached items are stale. It is stale if the cached item was read <*permission_refresh_interval*> minutes or more ago: • The logged-in user's permissions • The logged-in user's segment infomation • Cardholder options • System options
udf_refresh_interval	cache	5	1 to 99999	The time in minutes to check if the following cached items are stale. It is stale if the cached item was read <*udf_refresh_interval*> minutes or more ago: • UDF • List Builder • The permission group's possible values provided in the *get Lnl_User type* call

Property	Section	Default	Range	Description
user_cache_per_sid_ count_threshold	cache	150	1 to 99999	The maximum number of cached connections per user. For each normal API, the OpenAccess service needs a connection to do actions like database query, user cache, and so on. The connection is locked at the begin of API and released at the end of API. When the OpenAccess service gets an API call from a user, it will do one of following: • Creates a new connection and caches it if all connections for the same user are in use and the number of connections is less than *<user_cache_per_sid_count_threshold>* • Use a cached connection if there is a cached connection for the same user and it is not in use • Waits for a cached connection to be released if all connections for the same user are in use and the maximum number of connections is reached

Send Incoming Events Property

Name	Section	Default	Range	Description
max_incoming_ events_count	send-incoming-events	10	1 to 99999	The maximum count of incoming events to send in one call.

Badge Printing Properties

Use these properties to control how items are cleared from cache after making print requests. The expiration threshold is counted from the **submitted_at** property's value returned with the print request.

Property	Section	Default	Range	Description
poll_in_minutes	badgeprinting	15	1 to 1440	Determines how often the background thread polls for old badge print requests, in minutes.
expiration_threshold_in_minutes	badgeprinting	60	5 to 1440	Dictates how long the badge print requests will exist in the in-memory cache, in minutes.

Sample openaccess.ini content:

```
[badgeprinting]
poll_in_minutes=1
expiration_threshold_in_minutes=5
```

HTTP Request Properties

Use these properties to control how many HTTP requests OpenAccess can handle simultaneously.

Property	Section	Default	Range	Description
request_pool_size	http_request	32	1 to ({job_runner_name}_ thread_number - 3)	Configures how many HTTP requests OpenAccess can handle simultaneously. OpenAccess creates threads for each of those simultaneous requests. So, its maximum value should be three fewer than the size of the thread pool ({job_runner_name}_ thread_number - 3). Increasing or decreasing this value will impact system performance, since it will decrease or increase the number of other threads that handle these tasks: • Request queue task • Request timeout task • Handle a message from the message bus

Property	Section	Default	Range	Description
busy_request_pool_size	http_request	8	0 to ({job_runner_name}_thread_number - 3 - request_pool_size)	If this value is greater than 0, OpenAccess creates additional threads to handle the requests simultaneously. The client will receive "503 request pool full" error if the number of queued requests is greater than request_pool_size. It is recommended to modify this value to 0 to avoid this 503 error.

Sample openaccess.ini content:

```
[http_request]
request_pool_size = 100
busy_request_pool_size = 0
```

Queuing Property

Property	Section	Default	Range	Description
task_expiration	queue	60	1 to 1440	The time to expire a queued task, in minutes.

Job Runner/Thread Pool Properties

Property	Section	Default	Range	Description
names	job_runner	default	default	Lists the job runner names to be configured. Job runner names should match the service they are used by. The default job runner is named **default**. The OpenAccess job runner should be named **openaccess**. The REST proxy job runner should be named **rest_proxy**. Names should be comma separated. For example: **names=default,rest_proxy,openaccess**.
{job_runner_name}_ thread_number	job_runner	256	1 to 65535	Configures the size of the thread pool for the given job runner.
{job_runner_name}_ jobs_limit	job_runner	1024	1 to 65535	Configures the maximum number of queued jobs for the given job runner.

Sample openaccess.ini content:

```
[job_runner]
names=default
default_thread_number=30
default_jobs_limit=100
```

Timeout Property

Property	Section	Default	Range	Description
request_timeout	timeout	30	1 to 300	The OpenAccess timeout, in seconds. Requests taking longer than this value will result in an OpenAccess timeout error.

Event Context Provider Properties

Property	File > Section	Default	Description
HardwareCacheRefreshRate InHours	Lnl.OG. EventContext ProviderService. exe.config > appSettings	1	Hardware related cache refresh interval.
MinutesBetweenPrincipal CacheCleanups	application.config > appSettings	15	The permission cache refresh interval.

Bulk Export Properties

Parameters are fully customizable in the **ACS.ini** config file. To modify them, create a new [OpenAccessBulkExport] section. For example:

```
[OpenAccessBulkExport]
CleanupTimeout=120
ChunkSize=500
```

The following properties are available:

Property	Default	Range	Description
BinaryEncoding	base64	base64, hex	Encoding method for binary data (used in Lnl_MultimediaObject).
ChunkSize	100	0 to 10000	Exported file max size in MiB. Unlimited if set to 0.
CleanupTimeout	300	1 to 3600	Time in seconds to clean up the file path specified in ExportedFilesPath (all files kept at this location will be periodically removed).

Property	Default	Range	Description
DataSeparator	~ (tilde)	3 to 12 characters with UTF-8 encoding (depending on the number of encoded bytes per single UTF-8 character), but no more than 12 bytes.	Used to set one or more special characters for separating column data. **Note:** Not all special characters have the same byte length, which may vary from 1 to 4 bytes per single character. For example, the euro symbol (€) is 3 bytes long.
ExportedFilesPath	"C:/ProgramData/Lnl/nginx/temp/bulk_export_temp"	Directory path	Storage path for exported files (folders are created). **Note:** The API request will be rejected if the available disk space is less than 2 GiB plus the calculated size of the resulting uncompressed file.
MaxProcessingThreadCount	20	1 to 40	Max number of tasks used to process single request.
NullValueWriting	0	0 or 1	0 = columns with null value will be skipped 1 = columns with null value will be marked as <null>
OutputCompression	0	0 or 1	Allows you to enable additional .gzip compression for output files to reduce the file size and time required to download files.
ProcessingThreadCount	12	1 to 40, limited by Max-Processing-Thread-Count	Number of tasks used to process single request.

Property	Default	Range	Description
SQLQueryTimeout	600	0 to 3600	Timeout for SQL query request in seconds. Used to extend the timeout in case of large datasets and long-lasting queries.

Definitions, Acronyms, Abbreviations

Class
A definition of a type of object. For example, the Lnl_Reader class is a definition for an access control reader.

Client
A script or application that uses OpenAccess.

JSON
JavaScript Object Notation.

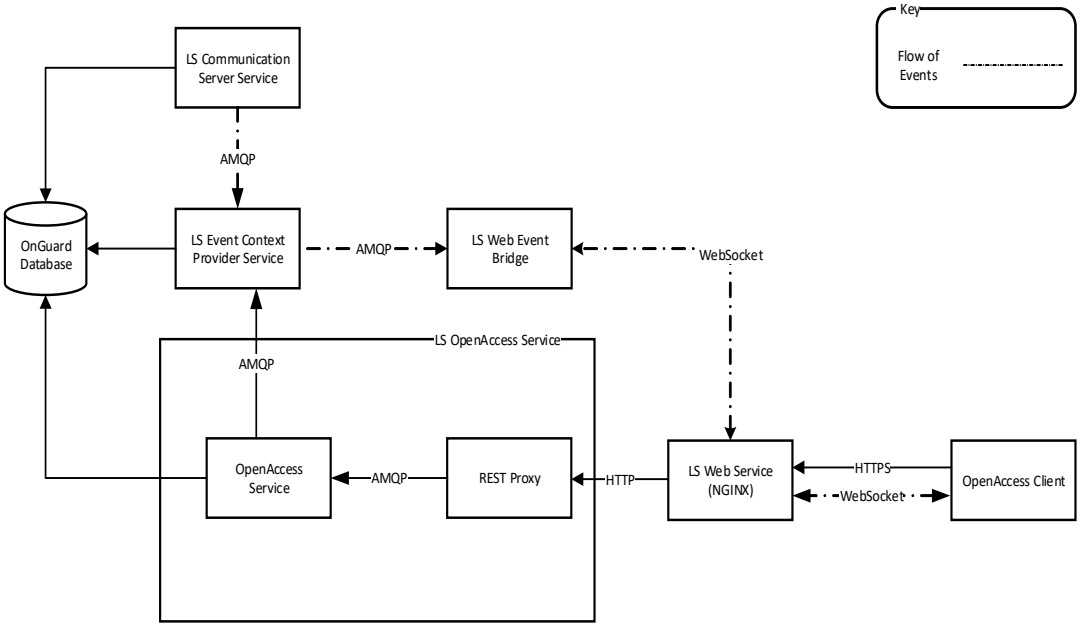
Object/Instance
A representation of a particular class with actual data.

Person
A cardholder or visitor.

SDK
Software Development Kit.

OpenAccess Architecture

OpenAccess Architecture



The LS Communication Server service publishes an event to the LS Event Context Provider service, which provides additional detail about the event. If the subscriber is using the LS Web Event Bridge, this service will begin publishing events to the client via WebSocket. For example, if the LS Communication Server service publishes an Access Granted event, the LS Event Context Provider service adds cardholder details. The event, with the added detail, is provided to the AMQP queue for each subscriber that has permission to receive information about the event. If the subscriber is using the LS Web Event Bridge, this service will publish events to the client via WebSocket.

The LS OpenAccess Service includes both the OpenAccess Service and REST Proxy. The LS Message Broker service provides the AMQP protocol. The LS Web Service (NGINX) exposes endpoints for each web service.

Note: Each subscriber has its own queue on the LS Message Broker service. This is done for security purposes, allowing subscribers to see only the event information they are authorized to see.

References and Applicable Documents

Note: Throughout this document, references to the **<OnGuard installation directory>** means the OnGuard installation directory. This is typically **C:\Program Files (x86)\OnGuard**, but may be different depending on system configuration and any custom path selected during OnGuard installation.

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

Information on JavaScript Object Notation (JSON) can be found at http://www.json.org/.

Information about NGINX can be found at http://nginx.org/.

CHAPTER 3

Getting Started

This section provides details about procedures that must be performed before using the LS OpenAccess service, including:

License for OpenAccess

OpenAccess is a licensed feature. For more information, refer to *Install Your OnGuard License* in the *Installation Guide*.

Application ID and Getting Started with Development

Each application or solution using OpenAccess must have a unique application ID and a specific license. You can obtain this development license along with additional license information by sending an email to openaccess@carrier.com with the subject *OA Dev Kit Request*. Your message should include the following:

- Contact information
- General description of the integration type you will develop using OpenAccess services

A company representative will contact you and help you obtain an OpenAccess license.

Starting OpenAccess

The LS OpenAccess service requires the LS Message Broker Service, and Secure Socket Layer (SSL) must be enabled. The LS Message Broker service is deployed with OnGuard Servers automatically. For information on configuring the LS Message Broker Service, refer to the *System Options Folder* chapter in the *System Administration User Guide*.

1. Confirm that the LS Message Broker service is running on the workstation identified on the **System Administration > System Options** form.
2. Confirm that the LS OpenAccess service is running on the workstation identified on the **System Administration > System Options** form.

Note: Both the LS Message Broker service location and the LS OpenAccess service location configured on the **System Administration > System Options** form must match the deployed certificate name perfectly, or SSL/TLS errors will result. For more information, refer to SSL/TLS Secure Channel Errors on page 352.

3. Confirm that the LS Web Service is running.
4. Confirm that the LS Event Context Provider service is running.

Note: The LS Event Context Provider service must run on the same host as the LS OpenAccess service.

5. Confirm that the LS Web Event Bridge service is running.

Note: By default, the LS Web Event Bridge service is configured to locate LS OpenAccess on the same server. If you installed the LS Web Event Bridge service on a different server than the LS OpenAccess service, open the **Lnl.OG.WebEventBridgeService.exe.config** file and edit the proxy to the Fully Qualified Domain Name (FQDN) of the server running LS OpenAccess.

For more information, refer to OpenAccess Architecture on page 30.

LS OpenAccess can also be run as an application. For troubleshooting purposes, select **Start > All Programs > OnGuard > Service and Support > OpenAccess**.

Stopping and Restarting the Services

Stopping and restarting the services is generally unnecessary. The services are installed with their properties configured to start automatically.

In a few limited circumstances, however, you will need to stop and restart the LS OpenAccess service and the LS Event Context Provider service to allow it to retrieve new configuration information. You should stop and then restart these services after any of the following changes are made:

- You change the database connection information. For more information, refer to the *Configuration Editor* appendix in the *Installation Guide*.
- You install a new license.
- You make segmentation changes.

Note: When enabling segmentation in an OnGuard system, or when enabling or disabling an OnGuard system's segmentation options, restart the OpenAccess service to prevent unexpected behavior. It is *not* necessary to restart the OpenAccess service when changing a user's segment access, as segment assignments are refreshed every 1 minute

by default. This refresh interval is configured using the permission_refresh_interval property. For more information, refer to Caching Properties on page 22.

- You make hardware changes, and you don't want to wait for the LS Event Context Provider to refresh its hardware cache. For more information, refer to Deploying the LS Event Context Provider Service on page 35.

If you change the location of the LS Message Broker service, you must also restart the following services:

- LS OpenAccess service
- LS Web Event Bridge
- LS Event Context Provider service

LS OpenAccess Service

REST service provider URL: <protocol>://<host>:8080/api/access/onguard/openaccess

The REST proxy that is part of the LS OpenAccess service interprets web requests intended for OpenAccess, and allows web clients to interface with the LS OpenAccess service. The LS OpenAccess service uses NGINX as the web service.

For information on how to format the "REST Request URL" proxy calls for each method, refer to Chapter 5: REST API Reference on page 59.

For some methods, "REST Request Body Contents" is also provided if a response is expected. The body is a JavaScript Object Notation (JSON) representation of the key-value pairs for each method.

Sample Request and Response With an Error

```
POST /api/access/onguard/openaccess/authentication?version=value

Header:
Application-Id: SUPPLIED_APPLICATION_ID
Session-Token: 12345-67890-12345-67890

Body:
{
    "user_name": "admin",
    "password": "badpass",
    "directory_id": "directory",
    "application_id":"OPEN_ACCESS_NON_PRODUCTION"
}

HTTP/1.1 401
{
    "error":
    {
        "code":"openaccess.general.invalidapplicationid",
        "message":"You are not licensed for OpenAccess."
    }
}
```

Authorization

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

Notes: OpenAccess caches user credentials and segments for 1 minute by default. This is done for performance reasons. Therefore, if a user is using OpenAccess and that user's permissions or segments change, the user will continue to have his old permissions until the 1-minute timeout is reached.

The Event Context Provider service, which is responsible for sending events matching event subscriptions, caches user credentials and segments for 15 minutes by default.

Authentication

Authentication to the LS OpenAccess service uses the OnGuard internal account or manual Single Sign-On (SSO) only. This differs from DataConduIT, which uses automatic SSO only. For more information, refer to the *Single Sign-On* section of the *Installation Guide*.

The following calls do not require authentication:

- get directories (See get directories, on page 68 for details.)
- get version (See get version, on page 62 for details.)

All other calls require authentication. Call **add authentication** to perform the authentication to the service. By default, an authenticated session expires 8 hours after it was created. For more information, refer to add authentication on page 69.

Supported Authentication Modes

OpenAccess supports the following authentication modes. Regardless of mode, the client begins with the **add authentication** call. Subsequent calls require an authentication token provided in one of the following ways.

Token-based Authentication Requests (all OnGuard versions)

For token-based authentication, the Session-Token parameter returned in the **add authentication** response must be provided in the request header for all subsequent calls.

Cookie-based Authentication Requests (OnGuard 7.6 and later)

The response to the **add authentication** call includes a Set-Cookie header resulting in a **OASessionID** session cookie. This cookie contains the session token and has the following attributes:

- HttpOnly
- Secure
- Samesite=Strict

To take advantage of cookie-based authentication, the application no longer needs to persist the session_token contained in the response body. Instead, the application must associate the cookie with a subsequent OpenAccess REST API request.

JavaScript API Request Example

Regardless of which JavaScript API you use, you must explicitly define or set the credentials for the HTTP request as shown below with **xhr.withCredentials=true**.

```
1  var xhr = new XMLHttpRequest();
2  xhr.open('GET', 'https://<URL>:8080/api/access/onguard/openaccess/
   type?type_name=lnl_reader&version=1.0', true);
3  xhr.withCredentials = true;
4  xhr.send(null);
```

Adding and Deleting Authentication

Every **add authentication** call should have a matching **delete authentication** call. Not calling **delete authentication** consumes resources unnecessarily over time.

It is also recommended that you keep a session token available (that is, do not **delete authentication**) until you no longer need to execute commands for at least several minutes. In other words, avoid rapidly adding and then deleting authentication within seconds. This can happen if you have a snippet of code that executes very frequently, and that code adds and then deletes authentication. Consider keeping the authentication token and executing the snippet frequently, and then deleting authentication only if the code won't execute again for another 1 or 2 minutes.

Deploying the LS Event Context Provider Service

The Communication Server publishes an event to the LS Event Context Provider service, which provides additional details about the event. For example, if the Communication Server publishes an Access Granted event, the LS Event Context Provider service adds cardholder information details. The event, with the added detail, is provided to the Direct Subscriber and Web Subscribers Event Queues where it can be shared with both Direct and Web Subscribers.

Note the following details about the LS Event Context Provider service:

- This service will only run on the workstation configured to run the LS OpenAccess service.
- This service logs all activity to the **EventContextProviderService.log** file located in the **C:\ProgramData\Lnl\logs** directory.
- The LS Event Context Provider service refreshes its cached information every 1 hour. This includes badge/cardholder details as well as hardware information.

Enabling Verbose Logging

By default, the log file only shows error messages. Enable Verbose Logging when additional log details are required, such as when troubleshooting OpenAccess issues.

Note: The Event Generator is another useful troubleshooting tool. Use Event Generator to create “fake” events that can be received by event subscribers. For more information, refer to Appendix A: Event Generator on page 357.

To enable Verbose Logging:

1. Launch the Configuration Editor by selecting **Start > All Programs > OnGuard > Service and Support > Configuration Editor**.
2. Select **Show advanced settings**.

3. In the Verbose Logging section, select **LS OpenAccess**.
4. Click [Save Changes].

Note: You do not need to restart the LS OpenAccess service after enabling Verbose Logging.

By default, the OpenAccess.log file is located in **C:\ProgramData\Lnl\logs**. Disable Verbose Logging when finished troubleshooting to prevent the log file from growing too large.

Sample Applications

Sample applications that demonstrate how to use the OpenAccess API are located in **<OnGuard installation directory>\doc\en-US\OpenAccess Samples**.

Note: LenelS2 does not offer any guarantee or extend any support for our sample applications, or any other third-party software used in creating those applications. Use these samples at your own risk.

Sample Web Applications

The following table lists the sample web applications:

Application	Description	APIs Used
Cardholder Search	Demonstrates how to authenticate, use pagination while searching, and provide some cardholder details such as the photo.	• get directories • add/delete authentication • get instances
Command and Control	Demonstrates how to list panels, readers, and panel status; search for panels by name; search for readers by name; paging; open doors; and change reader modes.	• get directories • add/delete authentication • get instances • execute method
Event Subscriber	Demonstrates how to create a subscription to receive events.	• get directories • add/delete authentication • add/modify/delete event_subscriptions • Web Event Bridge for receiving events using WebSocket

Configuring the Sample Web Applications

1. Load the sample web applications using one of the following methods:
   - Temporarily add CORS support for sites accessed on a local drive by uncommenting the example configuration for the “null” origin in the **C:\ProgramData\Lnl\nginx\conf\cors.conf** file. For more information, refer to Cross-Origin Resource Sharing on page 53.
   - Host the samples in NGINX to avoid CORS errors, by doing the following:

i. Rename **C:\ProgramData\Lnl\nginx\conf\modules\openaccess_samples.conf.disabled** to **openaccess_samples.conf**, removing the ".disabled" suffix. You can disable the samples again by adding the ".disabled" suffix again.

ii. Depending on where OnGuard is installed, you might need to update the value of **$onguard_install_dir** in **C:\ProgramData\Lnl\nginx\conf\environment.conf**.

2. Regardless of which method you used to load the sample web applications, restart the LS Web Service to pick up any NGINX configuration changes.

3. Each web application uses https://localhost:8080/api/access/onguard/openaccess as the default URL for the OpenAccess API. Each sample web application has a line in the **app.js** JavaScript file that looks similar to the following:

```
API_URL = 'https://localhost:8080/api/access/onguard/
openaccess', // OpenAccess REST API endpoint
```

Modify this line with the Fully Qualified Domain Name (FQDN) of your server.

Notes: If developing your own application, using WebSockets as the transport improves performance. To do this, target .NET Framework 4.6.1 or later instead of .NET Framework 4.0, as shown in this sample application. WebSockets functions correctly with any version of Windows supported by the OnGuard software.

When the LS Web Event Bridge service is restarted, it loses subscription details for all existing clients. Therefore, clients must re-subscribe to continue receiving events. New transient subscriptions must be created, but durable subscriptions can be re-established with the **ModifySubscription** call (ModifySubscription on page 197).

Running the Sample Web Applications

If loading the sample web applications from a local drive, use a web browser to load the web application's **index.html** directly from the local drive.

If hosting the sample web applications in NGINX, open the URL of the sample in the web browser.

Sample C# Applications

The following table lists the sample C# applications:

Application	Description	APIs Used
Command and Control	Demonstrates how to list panels and readers, change reader mode, and open doors.	• get directories • add/delete authentication • get instances • execute method
Event Subscriber	Demonstrates how to create a subscription to receive hardware and software events.	• add/delete authentication • add/modify/delete event_subscriptions • Web Event Bridge for receiving events using WebSocket

Configuring the Sample C# Applications

For the Command and Control sample, the API URL is initially hardcoded to https://localhost:8080/api/access/onguard/openaccess. Modify the **API_URL** in the **RequestBuilder.cs** file to the Fully Qualified Domain Name (FQDN) of your server.

For the Event Subscriber sample, the API URLs, credentials, and subscription parameters are configured in the **App.config** file.

Notes:

If developing your own application, using WebSockets as the transport improves performance. To do this, target .NET Framework 4.6.1 or later instead of .NET Framework 4.0, as shown in this sample application. WebSockets functions correctly with any version of Windows supported by the OnGuard software.

When the LS Web Event Bridge service is restarted, it loses subscription details for all existing clients, and in some cases (such as not using WebSockets, stopping the LS Web Event Bridge process and then restarting it immediately) the client is not notified that the LS Web Event Bridge service has restarted. Therefore, clients must re-subscribe to continue receiving events. Clients can call **ConnectionHeartbeat** periodically (for example, make the call every 10 seconds) to monitor the connection and re-subscribe events. ConnectionHeartbeat might generate an exception for some cases, such as:

The LS Web Event Bridge service is stopped. Clients can call **Microsoft.AspNet.SignalR.Client.Connection.Start()** to re-establish the connection, and then call **ConnectionHeartbeat** or **ModifySubscription** to re-subscribe events.

The OpenAccess session token is expired. Clients can call OpenAccess **add_authentication** to get new session token.

Building the Sample C# Applications

You can compile the C# applications with Visual Studio 2015 or later. These projects use NuGet for third party dependencies, so your workstation needs access to https://www.nuget.org for the NuGet packages to restore successfully.

Sample Java Application

The following table describes the sample Java application:

Application	Description	APIs Used
Event Subscriber	Demonstrates how to create a subscription to receive events. The sample Java application builds with Gradle (http://gradle.org).	• add/delete authentication • Web Event Bridge for receiving events using long polling

Configuring the Sample Java Application

The OpenAccess service URL, login credentials, and other parameters are defined in **src/main/java/Program.java**. Update these parameters to reflect your environment.

Building the Sample Java Application

1. Install the Java Development Kit (JDK).
2. Execute `gradlew build` at a command prompt. The first time you run this command, Gradle and the Java dependencies are downloaded. If you are behind a proxy, you might need update the

gradle.properties file with the correct proxy information. Uncomment each line by removing the # and specify the proxy host and port. Update all four lines to set the proxy for both **HTTP** and **HTTPS** protocols.

Running the Sample Java Application

Make sure the root certificate of the SSL certificate is installed in the Java **cacerts** certificate store, making the SSL connection to OpenAccess trusted.

1. On the OnGuard Server, the certificate file **ls_server_cert.pem** is available in the nginx conf folder by default. For example, if OnGuard is installed on the **C:** drive, then the default certificates folder is **C:\programdata\Lnl\nginx\conf\ls_server_cert.pem**. This certificate should be trusted on the OpenAccess client workstation. To import the certificate and trust it on the OpenAccess client workstation, copy the certificate file (**ls_server_cert.pem**) to the OpenAccess client workstation and import it to the trust store as described in step 2. If the certificate is replaced with a Certificate Authority (CA) trusted certificate, then determine the location of the certificate file as described in the "OnGuard and the Use of Certificates" appendix of the *OnGuard Installation Guide*.
2. Import the certificate to the default Java trust store using the Java Keytool. In the following sample command, *jdk1.8.0_65* is the installed Java Development Kit version, but you will modify the command based on your installed version:

   ```
   "c:\Program Files\Java\jdk1.8.0_65\jre\bin\keytool.exe" -import -v -trustcacerts -alias <Onguardhostname> -file <ls_server_cert.pem file location> -keystore "C:\Program Files\Java\jdk1.8.0_65\jre\lib\security\cacerts"
   ```

   The default password for the key store is **changeit** and **changeme**.

   After running this command, the Keytool asks `Trust this certificate?` If you respond [Yes], the certificate is added to the keystore. For more information, refer to https://docs.oracle.com/cd/E19509-01/820-3503/6nf1il6er/index.html.

Sample Python Applications

The following table describes a sample Python applications:

Application	Description	APIs Used

Enhanced transaction logging capabilities	Demonstrates how some OpenAccess calls can be enhanced with additional information, improving the readability of User Transaction Logs in OnGuard.	• add authentication
		• add instances
		  • Lnl_Cardholder
		  • Lnl_Badge
		  • Lnl_AccessLevelAssignment
		• delete instances
		  • Lnl_Cardholder

Bulk export sample	Demonstrates how Bulk Export could be performed with different parameters and instances, and provides a link to the download ouput file.	• add authentication
		• add instances
		  • Lnl_Cardholder
		  • Lnl_Badge
		• get instances
		  • defined by user, Lnl_Cardholder by default
		• delete instances
		  • Lnl_Cardholder

Enhanced Transaction Logging Sample

Two demonstrated methods are defined in the application:

Main.onboard_employee

An example of an employee on-boarding procedure that provides additional information for enhanced logging. Each OpenAccess request is enhanced with the operation GUID and operation description, so that the log reflects the intent behind each API call.

Main.offboard_employee

An example of an employee on-boarding procedure that provides additional information for enhanced logging. In this scenario, only one OpenAccess call used—delete instance Lnl_Cardholder. Providing the operation GUID and operation description in this case makes sense because one API call can make multiple changes to the system. In case of delete cardholder, it also removes the cardholder's badges, user accounts and so on. All changes are reflected in the log as separate actions, and all of them are tied together with the provided operation GUID and description.

Configuring the Enhanced Transaction Logging Sample Python Application

1. Install Python 3.10.0.
2. Install the Python Requests library in version 2.24.0 (www.docs.python-requests.org).
3. Update the **Config.py** file with your environment and OnGuard configuration details.
4. Check if your OnGuard system is set to manual Badge ID Allocation (**System Administration > Administration > Cardholder Options > Badge ID Allocation**). If it is, make any required changes in the **OpenAccessAPI.create_badge** method so that it provides the appropriate BadgeIDs required for your system configuration.

Running the Sample Python Application

When executed, this application makes changes in your OnGuard system. The application adds cardholders with badges, assigns them an access level, and deletes one of them afterwards. Make sure this is acceptable before proceeding.

1. Make sure that all settings in **Config.py** are correct for your environment.
2. Understanding that the script will make changes in your OnGuard system, open **Main.py** and uncomment lines 48 through 55 by removing the # characters before them.
3. Open a Windows Command Prompt and change the directory to the directory containing the **Main.py** file.
4. Execute the script by running `py .\Main.py`.

A message indicating that the application ran successfully should be visible in the command prompt window.

Bulk Export Sample

Two demonstrated methods are defined in the application:

Main.bulk export

Generates a bulk export file using custom parameters. Each OpenAccess request is decorated with GUID and operation description so that the log reflects the intent of each API call.

Main.format parameters

Format of the key-value parameter pairs for a bulk export API call.

Configuring the Bulk Export Sample Python Application

1. Install Python 3.10.0.
2. Install the Python Requests library in version 2.24.0 (www.docs.python-requests.org).

3. Update the **Config.py** file with your environment and OnGuard configuration details.
4. Modify the parameters in the **Main.api_parameters** dictionary so that it provides the desired output file (used values are default).
5. Adjust the number of loops executed in line 110 to create more employees, if necessary (the default is 20).

Running the Sample Python Application

When executed, this application makes changes in your OnGuard system. The application adds cardholders with badges, assigns them an access level, and deletes one of them afterwards. Make sure this is acceptable before proceeding.

1. Make sure that all settings in **Config.py** are correct for your environment.
2. Understanding that the script will make changes in your OnGuard system, open **Main.py** and uncomment lines 104 through 120 by removing the # characters before them.
3. Open a Windows Command Prompt and change the directory to the directory containing the **Main.py** file.
4. Execute the script by running `py .\Main.py`.

A message indicating that the application ran successfully should be visible in the command prompt window.

Swagger Specification and Interactive Documentation

Many developers find the Swagger specification and interactive documentation useful for testing an API and discovering how to work with it. Swagger is supported by many tools, which might be useful when developing solutions that use the OpenAccess REST API.

A Swagger specification is available for the OpenAccess REST API at **<OnGuard installation directory>\doc\en-us\OpenAccess Swagger\swagger.yaml** or at **https://<server>:8080/api/access/onguard/openaccess/swagger.yaml**. Live documentation is also available at **https://<server>:8080/api/access/onguard/openaccess/swagger**.

For information about Swagger, refer to http://swagger.io/. For information about the Swagger documentation specification, refer to http://swagger.io/specification/.

Notes: Live documentation is disabled by default. To enable live documentation, rename **C:\ProgramData\Lnl\nginx\conf\modules\openaccess_swagger.conf.disabled** to **openaccess_swagger.conf**, removing the ".disabled" suffix. You can disable the live documentation by adding the ".disabled" suffix again.

Depending on where OnGuard is installed, you might need to update the value of **$onguard_install_dir** in **C:\ProgramData\Lnl\nginx\conf\environment.conf**.

Restart the LS Web Service to activate the NGINX configuration changes.

Using Response Headers to Develop Secure Web Applications

To mitigate attacks and security vulnerabilities in web applications, you should utilize response headers as shown in the **httpsecurity.conf** file, located by default in the **C:\ProgramData\Lnl\nginx\conf** directory. You can either reference this **httpsecurity.conf** file, or you can specify the response headers you need directly in your web application code.

For more information about response headers and best practices for security, refer to:

- https://www.owasp.org/index.php/Main_Page
- https://www.owasp.org/index.php/List_of_useful_HTTP_headers#tab=Headers
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers#Security

CHAPTER 4

Using OpenAccess

Searching for Objects

Filters are specified in OpenAccess syntax, which is a subset of the Structured Query Language (SQL) supported by most databases.

The expected format of a filter is:

```
PROPERTY_NAME = VALUE
```

To help you understand the filtering syntax, here are some filters that you could use with OpenAccess. You could use these filters with the **get instances** call. For more information, refer to get instances on page 91.

Notes: You must use double-quotes around string delimiters when filtering. Single-quotes will result in a `system.parse` error.

You cannot search on some fields, such as encrypted text and password fields. If you search on an encrypted text or password field, an error is shown. Refer to the display_attributes response from get type on page 88 to determine if a field is searchable.

When creating dynamic filters, you must account for the maximum default lengths of the HTTP packet and header length depending on the type of API call. For example, when using GET calls, the filter is included in the URL, so the total combined length of the URL cannot exceed 2048 characters. We therefore recommend using filters less than 1500 characters.

If the `\` or `"` characters are part of a name, those characters must be escaped in the search string. For example, if the name to search for is `Includes\Backslash`, it should be entered in the filter as `Includes\\Backslash`, and if the name is `Includes"Quote`, it should be entered as `Includes\"Quote`.

Find all cardholders whose last name is not “Lake”

```
LastName != "Lake"
```

Find all cardholders whose last name starts with “La”

```
LastName like "La%"
```

Find all cardholders with either the last name is “Lake” or the first name is “Lisa”

```
LastName = "Lake" OR FirstName = "Lisa"
```

For more information, refer to Chapter 7: Data and Association Class Reference on page 223.

Date/Time Format

General Date/Time Format Information

Remember the following general date/time guidelines when using OpenAccess:

- Date/time formats that don’t contain seconds are assumed to be local time (not UTC)
- If sending or receiving date/time data that includes seconds, the data should be UTC time and include the UTC offset
- If receiving date/time data that doesn’t include the UTC offset, interpret the data as local time
- When receiving date/time data in the ISO 8601 format, ignore all unnecessary parts of the date/time string
- When sending **LongDateTime** or **ShortDateTime** data in the ISO 8601 format, truncate the length of the string according to the **max_length** values in the ALARM_ACK_HISTORY_UDF table
- To maintain backward compatibility, all date/time data created by customers using FormsDesigner is assumed to be in local time (not UTC)

Date/Time Format When Using OpenAccess API Calls

OpenAccess formats date/time values using the ISO 8601 standard:

`YYYY-MM-DDTHH:MM:SS+/-00:00`

For selected parameters, truncated representations of the ISO 8601 are used. For these representations, the time is assumed to be in local time. Examples:

- **Badge ACTIVATE/DEACTIVE date:** `YYYY-MM-DDTHH:MM`
- **Badge BDATE (birth date of the cardholder/user):** `YYYY-MM-DD`

All date and time values are reported to the server as strings, and are returned as strings in this format. The following example shows the time that came from an OpenAccess Server running in the Eastern Time Zone while daylight savings time is in effect:

`2016-04-05T20:33:47-04:00`

Date/Time Format When Using Events

The OpenAccess format for date/time strings does not apply when receiving events through subscriptions. In those instances, the date and time is a 64-bit integer that identifies the number of milliseconds after January 1, 1970, in UTC time.

For example: Modify a badge at 4/12/2024 1:41:34

```
new_LASTCHANGED :  1712943694000
```

Converting `1712943694000` returns `April 12, 2024 05:41:34.000 PM UTC`, which can then be converted to the local time zone of the subscriber.

Badge Activation and Deactivation Events

You must take special considerations when receiving Badge Activation and Deactivation event times.

Badge Activation and Deactivation date/time values are never associated with a world time zone in OnGuard systems. These dates and times are always considered to be the local time of the OnGuard server or client where these times are shown.

When a badge is added or modified, the software event received by a subscriber contains the new and old values.

```
new_DEACTIVATE:1837011600000
old_DEACTIVATE: 1710774000000
```

These values should be converted as if they were standard Unix timestamps, but the UTC time zone designation should be ignored. For example, the above values should be converted to:

```
new_DEACTIVATE:Saturday, March 18, 2028 5:00:00 PM UTC
old_DEACTIVATE: Monday, March 18, 2024 3:00:00 PM UTC
```

But the values should be applied using the local time of the subscriber:

```
new_DEACTIVATE:Saturday, March 18, 2028 5:00:00 PM (local time)
old_DEACTIVATE: Monday, March 18, 2024 3:00:00 PM   (local time)
```

Note: The local time of the subscriber should be the local time of any system viewing the activation or deactivation date/time of any badge. This includes any third-party integrations, OnGuard servers, clients, and panels.

Binary Format

When doing a get instances call, the REST proxy that is part of the LS OpenAccess service returns binary properties (indicated as **binary** in Data Classes on page 223) as base64-encoded strings. When doing an add or modify instance call for a type with binary data, OpenAccess expects the data as a base64-encoded string (for example, `iVBORw0KGgoAAAANSUhEUgAAAGIAAABUCAIAA...`).

Binary data is returned to a client as a map with the following structure:

```
"content_type":"image/jpeg",
"data":"[base64 encoded string]"
```

Notes: `"image/jpeg"` is an example of the content_type. The actual value is determined by the binary data.

When doing an add or modify call, the request does not include a map. Only the response on a get instance includes a map.

String Format

All strings are expected in UTF-8 format.

Features and Limitations

The following features and limitations are specific to class.

IMPORTANT: It is vital to understand that OnGuard default fields and data types can be modified or deleted by OnGuard customers using the FormsDesigner application. Therefore, it is important that OAAP Members make their integration configurable to handle any changes or deletions the customer may have made. Currently, the forms that can be edited using FormsDesigner are Cardholder Person Type, Visitor Person Type, Badge, Visits and Assets.

Cardholders and Visitors

Each cardholder and visitor instance has all of its user-defined fields (UDFs) exposed through OpenAccess. 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 OpenAccess. 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. 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 OpenAccess. 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.

User-Defined Fields

The user-defined field schema is updated every 5 minutes. If a user changes, adds, or deletes a property using FormsDesigner, it will take up to 5 minutes for the change to appear in the LS OpenAccess service.

Notes: OpenAccess generates property names based on the field names shown in FormsDesigner.

When provided via the object name of a User Defined Field (UDF) in FormsDesigner, the display_name attribute is the user-friendly name of the item. For more information, refer to get type on page 88. Also refer to the "Field Properties Folder – General Settings Form" section in the FormsDesigner User Guide.

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

When doing a get type call, if the type is a UDF type such as **cardholder** or **badge**, and if the type contains list builder items, the list builder items themselves are returned as possible values for that property. The type definitions themselves have a 5-minute UDF refresh interval, but the values of the properties on the possible value list is refreshed each time you call a get type. You can also call get instances on the list builder type directly to get all possible values.

Therefore, if you perform a get type call for Lnl_Cardholder, the Title property returns a list of possible values associated with it. The schema for the Lnl_TITLE type and the Lnl_Cardholder type will refresh every 5 minutes, but the list of possible values for the Title property is not cached and is provided for convenience. These values are refreshed each time you call a get type on Lnl_Cardholder. You can also get this information by doing a get instances on Lnl_TITLE directly at any time to get current values for the type.

SegmentID

SEGMENTID only appears as a property in data classes that support segmentation when segmentation for that class is enabled. For more information, refer to get segmentation_settings on page 165. Restarting the LS OpenAccess service is required when making segmentation changes.

Receiving Events

Durable vs. Transient Event Subscribers

An event subscriber can be *durable* or *transient*, which impacts how many events are received, as well as how often a **modify event_subscriptions** call must be sent in order to keep the subscriber active.

- *Durable event subscribers* receive events that occur while the subscriber is online (for a process) or logged in (for a user), as well as events that occur when the subscriber is offline/logged out. When the subscriber comes online/logs in again, the system sends the missed events to the subscriber. To continue receiving events and remain active, a durable subscriber must send a **modify event_subscriptions** call every seven days.

Note: Because a durable subscriber's events are stored while the subscriber is offline, you should minimize offline time and delete durable subscribers that are no longer needed, to avoid overwhelming the Message Broker.

- *Transient (non-durable) event subscribers* only receive events that occur while the subscriber is online (for a process) or logged in (for a user). Events that occur when the subscriber is offline/logged out are not sent. To continue receiving events and remain active, a transient subscriber must send a **modify event_subscriptions** call every 24 hours.

Note: If either the LS Message Broker service or the LS Event Context Provider service is not running, hardware and alarm acknowledgement events might not reach the client even if those events are reported within Alarm Monitoring and are using a durable event subscription.

If a subscriber fails to send a **modify event_subscriptions** call in the expected time frame (seven days for a durable subscription, 24 hours for a transient subscription), the system will delete the subscription and stop sending events. The LS Event Context Provider checks for and deletes expired subscriptions every 10 minutes.

To learn more about **event_subscriptions** calls:

- See add event_subscriptions on page 78.
- See modify event_subscriptions with id on page 80.
- See delete event_subscriptions with id on page 82.

Note: Deleted subscriptions cannot be reinstated. Create a new subscription using the **event_subscriptions** method.

Using Event Filters with Subscriptions

When an event filter is specified with a subscription, only the events that match the criteria specified in the filter are forwarded to the subscriber. The grammar of the filter supports a basic subset of the OData filter expression language. Visit http://www.odata.org/documentation/odata-version-2-0/uri-conventions/#FilterSystemQueryOption for details.

There are two formats for filtering event properties:

- `<property name> <operator> <property value>`

  With this filter format, the *property name* is not case sensitive, but the *operator* and *property value* are case sensitive. All hardware and alarm acknowledgement events, as well as the common properties of software events, use this filter format. For more information about

common properties of software events, refer to Common Properties for All Software Events on page 215.

For example: `business_event_class eq 'software_event'` is a valid filter, but `business_event_class Eq 'Software_Event'` is not a valid filter.

- `<new_/old_properties>/[<object property name>] <operator> <value>`

  With this filter format, the *new/old properties* is not case sensitive, but the *object property name*, *operator*, and *value* are case sensitive. All software event object properties use this filter format. For more information, refer to Software Event Reference on page 215.

  For example: `new_properties/[LASTNAME] eq 'Smith'` is a valid filter, but `new_proproties/[LastName] Eq 'smith'` is not a valid filter. Also with this format, the value for a property that is an int64 must have an 'L' appended. For example: `new_properties/[ID] eq 8` for filtering software events by badge ID will not work. That filter must be written as `new_properties/[ID] eq 8L`.

Notes: OpenAccess will not return an error if you filter on a field that does not exist.

Also, you cannot filter software or hardware events using `timestamp` or `object_id`.

If the `\` or `"` characters are part of a name, those characters must be escaped in the search string. For example, if the name to search for is `Includes\Backslash`, it should be entered in the filter as `Includes\\Backslash`, and if the name is `Includes"Quote`, it should be entered as `Includes\"Quote`.

Here are some examples of event filters:

Example	Event Filter
Receive only hardware events with event ID equal to 214. (Set reader mode PIN or Card)	business_event_class eq 'hardware_event' and event_id eq 214
Receive only hardware events related to a specific cardholder.	business_event_class eq 'hardware_event' and cardholder_last_name eq 'Smith'
Receive software events.	business_event_class eq 'software_event'
Receive hardware events.	business_event_class eq 'hardware_event'
Receive only software events related to a specific badge.	business_event_class eq 'software_event' and software_event_object_type eq 'Badge' and new_properties/[ID] eq 1L

The following hardware and alarm acknowledgement event properties can only be specified in the definition of the filter parameter for subscription API calls:

Note: The following table is for hardware and alarm acknowledgement events only. All software events can be specified in the definition of the filter parameter for subscription API calls. For more information, refer to Software Event Reference on page 215.

Field Name	Field Description
access_granted_entry_-made	**Definition:** See Properties for Access Granted Events on page 206. **Type:** Boolean **Example:** access_granted_entry_made eq true
alarm_id	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** 32-bit signed integer **Example:** alarm_id eq 12
alarm_name	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** String **Example:** alarm_name eq ‘Access Granted Entry Made'
area_entering_id	**Definition:** See Properties for Access Granted Events on page 206. **Type:** 32-bit signed integer **Example:** area_entering_id eq 3
area_entering_name	**Definition:** See Properties for Access Granted Events on page 206. **Type:** String **Example:** area_entering_name eq ‘Default Area’
area_exiting_id	**Definition:** See Properties for Access Granted Events on page 206. **Type:** 32-bit signed integer **Example:** area_exiting_id eq 3
area_exiting_name	**Definition:** See Properties for Access Granted Events on page 206. **Type:** String **Example:** area_exiting_name eq ‘default area’
asset_id	**Definition:** See Properties for Asset Events on page 208. **Type:** string **Example:** asset_id eq ‘7’
associated_text	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** String **Example:** associated_text eq ‘secured room’
badge_extended_id	**Definition:** The full Federal Agency Smart Credential Number (FASC-N) or full UUID from a Personal Identity Verification (PIV)-based card or other Federal Information Processing Standard (FIPS) 201-based card. **Type:** String; maximum length = 64 characters **Example:** badge_extended_id eq ‘11112222333333456666666666788889’
badge_issue_code	**Definition:** See Properties for Access Granted Events on page 206. **Type:** 32-bit unsigned integer **Example:** badge_issue_code eq 4

Field Name	Field Description
badge_key	**Definition:** See Properties for Access Granted Events on page 206. **Type:** 64-bit signed integer **Example:** badge_key eq 1326
badge_key_str	**Definition:** See Properties for Access Granted Events on page 206. **Type:** String **Example:** badge_key_str eq '1326'
badge_id	**Definition:** The ID encoded on a badge. **Type:** 64-bit signed integer **Example:** badge_id eq 123456789
badge_id_str	**Definition:** The ID encoded on a badge. **Type:** String **Example:** badge_id_str eq '123456789'
badge_status_name	**Definition:** See Properties for Access Granted Events on page 206. **Type:** String **Example:** badge_status_name eq 'Active'
badge_type_name	**Definition:** See Properties for Access Granted Events on page 206. **Type:** String **Example:** badge_type_name eq 'Employee'
biometric_score	**Definition:** See Properties for Biometric Events on page 209. **Type:** 32-bit unsigned integer **Example:** biometric_score eq 13
business_event_class	**Definition:** The type of event that occurred. **Type:** String **Example:** business_event_class eq 'hardware_event' **Note:** Valid values include Acknowledgement Event, generic_event, hardware_event, hardware_status, software_event, routing_event, shutdown_thread, or text_message.
cardholder_first_name	**Definition:**See Properties for Access Granted Events on page 206. **Type:** String **Example:** cardholder_first_name eq 'John'
cardholder_key	**Definition:** See Properties for Access Granted Events **on page 206.** **Type:** 32-bit integer **Example:** cardholder_key eq 636719
cardholder_last_name	**Definition:** See Properties for Access Granted Events on page 206. **Type:** String **Example:** cardholder_last_name eq 'Smith'
controller_id	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** 16-bit unsigned integer **Example:** controller_id eq 5 **Note:** The ListEntityData service can be used to request a list of controllers in the system.

Field Name	Field Description
controller_name	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** String **Example:** controller_name eq ‘access panel 13’ **Note:** The ListEntityData service can be used to request a list of controllers in the system.
controller_time_zone_id	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** 16-bit unsigned integer **Example:** controller_time_zone_id eq 22 **Note:** The ListEntityData service can be used to request a list of controllers in the system.
device_id	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** 16-bit unsigned integer **Example:** device_id eq 123456
device_name	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** String **Example:** device_name eq ‘reader2’
device_type	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** 8-bit signed integer **Example:** device_type eq 1 **Note:** Valid values include 1 CCTV camera and 0 (all other device types)
event_parameter	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** 32-bit unsigned integer **Example:** event_parameter eq 12
event_parameter description	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** string **Example:** event_parameter_description eq ‘channel number3’
event_source_name	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** string **Example:** event_source_name eq ‘access panel 13’
event_subtype	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** 16-bit unsigned integer **Example:** event_subtype eq 76
event_type	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** 8-bit unsigned integer **Example:** event_type eq 0
intrusion_area_id	**Definition:** See Properties for Intrusion Events on page 210. **Type:** 16-bit unsigned integer **Example:** intrusion_area_id eq 5

Field Name	Field Description
intrusion_user_id	**Definition:** See Properties for Intrusion Events on page 210. **Type:** string **Example:** intrusion_user_id eq '5'
receiver_area_id	**Definition:** See Properties for Intrusion Events on page 210. **Type:** 16-bit unsigned integer **Example:** receiver_area_id eq 3
receiver_controller_id	**Definition:** See Properties for Intrusion Events on page 210. **Type:** 16-bit unsigned integer **Example:** receiver_controller_id eq 6
receiver_line_number	**Definition:** See Properties for Intrusion Events on page 210. **Type:** 16-bit unsigned integer **Example:** receiver_line_number eq 4
source	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** string **Example:** source eq 'CommServer@DPSARRO1-VM2012'
segment_id	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** 32-bit unsigned integer **Example:** segment_id eq 3
subdevice_id	**Definition:** See Properties for Controller-Based Events on page 205. **Type:** 16-bit unsigned integer **Example:** subdevice_id eq 3
transmitter_id	**Definition:** See Properties for Transmitter Events on page 210. **Type:** 32-bit signed integer **Example:** transmitter_id eq 4
transmitter_input_id	**Definition:** See Properties for Transmitter Events on page 210. **Type:** 32-bit signed integer **Example:** transmitter_input_id eq 6
video_channel	**Definition:** See Common Properties for All Hardware Events on page 203. **Type:** 32-bit signed integer **Example:** video_channel eq 7

Cross-Origin Resource Sharing

If you have a web application or site that makes requests against the OpenAccess API but is hosted on a different server, you must enable Cross-Origin Resource Sharing (CORS):

1. Locate the **cors.conf** file and open it for editing. This file is located in **C:\ProgramData\Lnl\nginx\conf**.
2. Find the section that begins with the following line:

```
map $http_origin $cors_http_origin {
```

3. Add an entry for each HTTP origin that accesses the OpenAccess API. There are several commented out examples in the config file (remove the "#" and then modify them as needed). There is support for simple strings as well as regular expressions. Refer to http://nginx.org/en/docs/http/ngx_http_map_module.html for more details about the NGINX map directive.
4. Save the file and restart the LS Web Service service.

OpenAccess Operations From Behind a Network Proxy

If you are using OpenAccess to perform OpenAccess operations from behind a network proxy (for example, issue mobile credentials, or using a third-party provider for OnGuard authentication, or performing any other operation that requires OpenAccess to reach an external network location) and are behind a network proxy, you must make the following configuration change on the server where the LS OpenAccess service is running. Change the logon account for the LS OpenAccess service from Local System to a user whose account has the correct proxy settings configured.

Version

Every OpenAccess API call must include a version, with versions starting at “1.0” and incrementing up from there. OpenAccess uses the version to maintain backward compatibility as the API is updated.

Versions are formatted `<major>.<minor>`. Each API call is versioned independently. For example, you can call `get event_subscriptions (version = "1.0")` and then call `authenticate (version = "2.7")`. Versions with the same `<major>` components are compatible, but might offer different optional features. For example, calling `authenticate` version 1.3 might offer a `fast=true` property. This property might be ignored by version 1.0, but the basic authenticate functionality is the same. Versions with different `<major>` components are not forward compatible. For example, an API version 1.0 call that contains API version 2.0 parameters will result in an error.

Operation GUID

Some OpenAccess calls specify the optional parameter **operation_guid**, which is an arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. This GUID will be used to group operations together in the User Transaction Log reports, enhancing readability. In conjunction with the **operation_description** parameter (Operation Description on page 55), use **operation_guid** to describe what functionality a set of executed operations will deliver.

For example, you can correlate the `GET /instances?type_name="Lnl_Cardholder"`, `GET /instances?type_name="Lnl_Badge"`, `GET /instances?type_name="Lnl_AccessLevelAssignment"` calls with the same GUID and describe them as “View cardholder details”.

Generate a new GUID each time a set of operations is invoked.

Operation Description

Some OpenAccess API calls specify the optional parameter **operation_description**. This is a user-friendly free-form description for the operation being performed. For example, "View a list of cameras". Use this description to enhance the readability of the User Transaction Log reports.

OpenAccess and Brute Force Attack Protection

OpenAccess protects users against Brute Force Attacks, where an attacker attempts to log into a user account repeatedly in an attempt to determine the password.

For internal accounts, three failed log-in attempts to the same account will lock that account from OpenAccess for 5 minutes.

Note: This Brute Force Attack protection only applies to internal accounts. Directory accounts are protected according to directory policies.

Preventing Malicious Code in OpenAccess Responses

Developers of OpenAccess clients can prevent clients from receiving strings in plain text that might contain malicious code that would be executed by a browser.

For example, the names of directories returned in the **get directories** call are strings. If someone embedded JavaScript code in a directory name, then the response to that call would contain that JavaScript code, and would be executed by the browser. For example, the **Name** property below contains JavaScript code:

```
{
   "property_value_map": {
   "ID": "id9",
   "Name": "<script>alert(1);</script>"
   "directory_type": 0
   }
}
```

To prevent malicious code from running on the client browser, the OpenAccess Server encodes non-alphanumeric characters in text strings, eliminating raw HTML elements. For example, encoding the code sample shown above would result in the client receiving something like this instead:

```
{
   "property_value_map": {
   "ID": "id9",
   "Name": "&lt;script&gt;alert&#x28;1&#x29;&#x3B;&lt;&#x2F;script&gt;",
   "directory_type": 0
   }
}
```

This encoding would prevent the script from being executed by the browser.

Note: Only strings from free-format fields containing user-entered text are encoded. Other string properties are not encoded.

But this also means that the client would need to decode the encoded characters in order to display the string correctly. For example, a directory like **<internal>** would be encoded as **<internal>** even though this is not an HTML tag, and it would be displayed that way unless the client decodes it.

The mechanisms being used to encode the characters are known as *HTML Entity Encoding* and *HTML Attribute Encoding*, described here and elsewhere online: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet#Output_Encoding_Rules_Summary.

For OpenAccess clients to take advantage of this encoding, they would have to use the updated versions of the methods listed below, and they would have to implement the decoding logic for any returned string properties:

Method	Minimum version that supports string encoding
get directories (see get directories on page 68)	1.1
get instances (see get instances on page 91)	1.4
get event_subscriptions (see get event_subscriptions on page 74)	1.1
add event_subscriptions (see add event_subscriptions on page 78)	1.1
modify event_subscriptions with id (see modify event_subscriptions with id on page 80)	1.1
get logged_in_user (see get logged_in_user on page 133)	1.1
get managed_access_levels (see get user managed_access_levels on page 134)	1.1
get user (see get user on page 137)	1.1
get editable_segments (see get editable_segments on page 140)	1.1
get cardholders (see get cardholders on page 113)	1.1
get user_segments (see get user_segments on page 141)	1.1
get authorizaton warning settings (see get authorization warning settings on page 154)	1.1

Method	Minimum version that supports string encoding
get visit settings (see get visit settings on page 167)	1.1
get badge_mobile_devices (see get badge mobile_devices on page 98)	1.1
add badge_issue_mobile_credential (see add badge issue_mobile_credential on page 99)	1.1
get print_request (see get badge print_request on page 95)	1.1
add print_request (see add badge print_request on page 96)	1.1
get directory_accounts (see get directory_accounts on page 148)	1.1
get directory_accounts_matching_cardholders (see get directory_accounts_matching_cardholders on page 149)	1.1
get user_preferences (see get user preferences on page 143)	1.1
put user_preferences (see put user preferences on page 144)	1.1
get video_recorders (see get video_recorders on page 125)	1.2
get auth_data (see get video_recorder auth_data on page 128)	1.1
get logged_events (see get logged_events on page 82)	1.1
get badge_printers (see get badge_printers on page 101)	1.1
get console layouts (see get console layouts on page 153)	1.1
put console layouts (see put console layouts on page 154)	1.1

CHAPTER 5

REST API Reference

This section provides details about the LS OpenAccess service's Application Programming Interface (API).

The REST proxy that is part of the LS OpenAccess service allows you to create a client against a REST API to OnGuard through NGINX as the web service which abstracts the AMQP language. The LS Web Service is the service hosting NGINX. Use the REST Request URL and body contents described below for each API call.

Notes:

The errors you might receive in the response header are very helpful when creating a client application that uses OpenAccess. Also, any request taking longer than 60 seconds to fulfill results in a timeout error. For more information, refer to Error Messages on page 349.

You will receive an HTTP 200 code whenever an API call executes successfully.

API calls are handled asynchronously. It is the responsibility of the client to handle synchronization as needed.

When creating Body content, this sample shows when to use quotation marks:

```
{
    "some_string": "I am a string",
    "some_number": 1000,
    "some_bool": false
}
```

Task queuing: dealing with long running requests

Some requests might take a long time, especially requests that access external systems, such as Active Directory. Standard OpenAccess requests will time out after 30 seconds if the HTTP request doesn't time out sooner, depending on the client. Any request that you expect to run long can be queued as a task by adding a `queue` property to the request, set to `true`. For example:

```
GET /directory_accounts_matching_cardholders?directory_id=id1
&cardholder_ids=[1,2,3,4,5,6,7,8,9,10]
&filter=displayname has 'firstname' and displayname has 'lastname'
&queue=true
```

```
&version=1.0
```

When a request is queued in this way, OpenAccess will queue a task for execution and return a 202 (Accepted) HTTP status code and a response identical to `GET /queue/{id}`. For example:

```
{
  "id": "5c4b7890-ee73-4199-b3d3-366003eb8ca1",
  "status": "pending",
  "version": "1.0"
}
```

The `id` property indicates the ID of the queued task, which can be used to check the status of the task:

```
GET /queue/5c4b7890-ee73-4199-b3d3-366003eb8ca1?version=1.0
```

When the task is complete, the response will include the response to the queued request:

```
{
  "id": "5c4b7890-ee73-4199-b3d3-366003eb8ca1",
  "response": {
    ...
  },
  "status": "complete",
  "version": "1.0"
}
```

The response can be retrieved any number of times until the task is deleted. A completed task can be deleted with `DELETE /queue/{id}` or it will be deleted automatically after 1 hour.

Even though you can queue any request, it is only recommended when a request is expected to run long, like `GET /directory_accounts` and `GET /directory_accounts_matching_cardholders`.

User Transaction Logging

In the [Performance] section of the **ACS.ini** file, adding the flag `UserTransLoggingLevel=1` will cause all OpenAccess get/read API calls that support this flag to be logged to the user transaction log.

Required Parameters for OpenAccess Requests

Name	Type	Location	Required	Description
Session-Token	string	header	yes (for token-based authentication)	The authentication token for the current user session. **OR** Client session cookie containing the authentication token. For more information, refer to Authentication on page 34.
OASessionID	string	header	yes (for cookie-based authentication)	
Application-Id	string	header	yes	A unique Application-Id is provided by LenelS2 OnGuard Technical Support. For more information, refer to License for OpenAccess on page 31.
do_not_reset_ inactivity_timer	boolean	query	no	Default = false. If set to true, the OpenAccess call does not reset the inactivity timer to zero. If this parameter is not provided, it is assumed to be false and the call will reset the inactivity timer to zero. This parameter is available to all OpenAccess calls except the following: • get version • get directories • get identity_provider_url • get keepalive • post authentication • delete authentication
queue	boolean	query	no	Queues the request as a task, and returns a response identical to `GET /queue/{id}`. Defaults to `false` if not provided.
version	string	query	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Note: The following methods are exceptions and do not require an authentication token:

- get version
- get directories
- add authentication

General OpenAccess API Calls

get version

Used to retrieve the OnGuard product name and version information.

REST Request URL: `GET /api/access/onguard/openaccess/version?version=value`

get version

Name	Type	Required	Description
operation_ description	string	no	Some OpenAccess calls specify this additional optional parameter, which is a user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	Some OpenAccess calls specify this additional optional parameter, which is an arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get version response

Name	Type	Description
product_name	string	A string representing the product name and major version (stored in the Windows registry as "InstalledProductName"). For example: `OnGuard #.#`.
product_version	string	A string representing the detailed version information (stored in the Windows registry as "ProductVersion"). For example: `(#.#.###)`.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Additional operation_guid Details

Some OpenAccess calls specify this additional optional parameter, which is an arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. This GUID will be used to group operations together in the User Transaction Log reports, enhancing readability. In conjunction with the **operation_description** parameter, use **operation_guid** to describe what functionality a set of executed operations will deliver.

For example, you can correlate the `GET /instances?type_name="Lnl_Cardholder"`, `GET /instances?type_name="Lnl_Badge"`, `GET /instances?type_name="Lnl_AccessLevelAssignment"` calls with the same GUID and describe them as “View cardholder details”.

Generate a new GUID each time a set of operations is invoked.

get keepalive

Used to prevent idle session timeout.

REST Request URL: `GET /api/access/onguard/openaccess/keepalive?version=value`

get keepalive

Name	Type	Required	Description
only_forever_user	boolean	no	If true, and if the logged-in user has the permission to allow non-expiring sessions, prevents both the inactivity timeout and the token session timeout from occurring.

get keepalive response

Name	Type	Description
session_token	string	An authentication token with the same properties as the token returned in response to the **add authentication** request. For the **get keepalive** request, however, the session_token is returned only if: • the only_forever_user parameter was provided in the request with a value of `true`, and • the logged-in user has the “Allow non-expiring session” permission enabled, and • the current time is within one “inactivity timeout” period before the current token will expire (or within 15 minutes if inactivity timeout is not enabled) When this parameter is returned, the client can provide it in sub-sequent requests to avoid the session timeout.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get feature_availability

Used to check if an OnGuard license feature is available.

REST Request URL: `GET /api/access/onguard/openaccess/feature_availability?version=value`

get feature_availability response

Name	Type	Description
is_available	boolean	Indicates if this license feature is available.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get licensedata

Used to get the values of the OnGuard license feature from the License Server.

Note: There is a 6 hour cache on the values queried from the License Server. If a new license is installed, the response values will refresh after the cache time has expired (no more than 6 hours). If you need these values to refresh immediately, then restart the License Server.

REST Request URL: `GET /api/access/onguard/openaccess/licensedata?version=value&feature_id={feature_id}`

get licensedata

Name	Type	Required	Description
feature_id	string	yes	The feature ID for which the client is requesting details.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get licensedata response

Name	Type	Description
FeatureName	string	The name of the feature as seen in License Administration.
InUse	int32	The InUse count of the feature as seen in License Administration.
Max	int32	The Max count of the feature as seen in License Administration.
Value	string	The Value of the feature as seen in License Administration.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get queue

Gets the queued tasks created by the user. This method is only intended to check the status of multiple tasks. Request a specific task to get the response. Users can only view their own queued tasks.

Note: This call does not return queues associated with **add authentication** requests.

REST Request URL: `GET /api/access/onguard/openaccess/queue?version=value`

get queue response

Name	Type	Description
item_list	list	A list of queued tasks. Each task in the list is provided with its unique ID and status.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get queue/{id}

Gets the queued task with the given ID, which includes the response when the task is complete. Users can only get their own queued tasks except for the **add authentication** queue task. This is because OpenAccess does not know which user created the **add authentication** queue tasks before that user has logged in.

REST Request URL: `GET /api/access/onguard/openaccess/queue/{id}?version=value`

get queue/{id}

Name	Type	Required	Description
id	string	yes	The ID of the task to return.

get queue/{id} response

Name	Type	Description
id	string	The ID of the task to return.
response	map	The response of a queued task.
status	string	The status of the queued task.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete queue/{id}

Deletes the queued task with the given ID. All queued tasks will be deleted automatically after 1 hour if not manually deleted. Only complete tasks can be deleted, and users can only delete their own queued tasks except for the **add authentication** queue task. This is because OpenAccess does not know which user created the **add authentication** queue tasks before that user has logged in.

REST Request URL: `DELETE /api/access/onguard/openaccess/queue/{id}?version=value`

delete queue/{id}

Name	Type	Required	Description
id	string	yes	The ID of the task to return.

delete queue/{id} response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add partner_values

Used by OnGuard software partners.

REST Request URL: `POST /api/access/onguard/openaccess/partner_values?version=value`

add partner_values

Name	Type	Required	Description
partner_value_1	int32	no	First partner value.
partner_value_2	int32	no	Second partner value.
partner_value_3	int32	no	Third partner value.
partner_value_4	int32	no	Fourth partner value.
partner_value_5	int32	no	Fifth partner value.

add partner_values response

Name	Type	Description
result	boolean	Result of the operation.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

modify partner_values

Used by OnGuard software partners.

REST Request URL: `PUT /api/access/onguard/openaccess/partner_values?version=value`

modify partner_values

Name	Type	Required	Description
partner_value_1	int32	no	First partner value.
partner_value_2	int32	no	Second partner value.
partner_value_3	int32	no	Third partner value.
partner_value_4	int32	no	Fourth partner value.
partner_value_5	int32	no	Fifth partner value.

modify partner_values response

Name	Type	Description
result	boolean	Result of the operation.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get susp_expiration

Retrieves the OnGuard SUSP expiration information.

REST Request URL: `GET /api/access/onguard/openaccess/susp_expiration?version=value`

get susp_expiration

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get susp_expiration response

Name	Type	Description
susp_expiration	datetime (string)	The SUSP expiration date. This date is maintained by LenelS2 and is made available to OnGuard during license activation.
message_status	string	Provides the status of the SUSP expiration. Possible values are ACTIVE, EXPIRED, GRACE_PERIOD, INACTIVE, DISABLED, LONG_NOTICE, SHORT_NOTICE, MEDIUM_NOTICE. The values for the NOTICE statuses are available in the OnGuard license file.
SUSP_to_expire	int32	Provides the number of days remaining before the SUSP expires. This number is calculated based on the formula: **today_ date – susp_expiration**.
days_to_expire	int32	Provides the number of days remaining before the SUSP expires. This number is calculated based on the formula: **today_ date – susp_expiration + grace_period**. The grace_period value is provided in OnGuard license file.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Login and Logout

get directories

Returns a list of directories configured within the OnGuard software. If using an internal account for authentication, you can call **add authentication** without specifying a directory ID. It is generally called prior to **add authentication** to get the user's directory ID.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/directories?version=value`

get directories response

Name	Type	Description
total_items	int32	The total number of directories in the filter result.
item_list	list	A list of items returned if directories exist. If present, each item consists of a property_value_map.

get directories response

Name	Type	Description

property_value_map	map	A map of directory attributes:
		• ID: Internal directory ID
		• Name: Name of the directory
		• directory_type: Directory type. Possible values:
		  • -1: Internal Directory
		  • 0: LDAP
		  • 1: Microsoft Active Directory
		  • 2: Microsoft Windows NT 4 Domain
		  • 3: Windows Local Accounts
		  • 4: OpenID Connect

version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add authentication

IMPORTANT: Version 2.0 of this call was introduced in OnGuard 7.5.

Authenticates a user with the LS OpenAccess service.

Notes: The add authentication call returns a token to be used in all subsequent authorized calls.

For information about how OpenAccess protects against Brute Force Attacks, refer to OpenAccess and Brute Force Attack Protection on page 19.

The response to the **add authentication** call issues a Set Cookie response header that establishes the HTTPOnly cookie. For more information, refer to Authentication on page 34.

Every **add authentication** call should have a matching **delete authentication** call. Not calling **delete authentication** consumes resources unnecessarily over time.

It is also recommended that you keep a session token available (that is, do not **delete authentication**) until you no longer need to execute commands for at least several minutes. In other words, avoid rapidly adding and then deleting authentication within seconds. This can happen if you have a snippet of code that executes very frequently, and that code adds and then deletes authentication. Consider keeping the authentication token and executing the snippet frequently, and then deleting authentication only if the code won't execute again for another 1 or 2 minutes.

REST Request URL: `POST /api/access/onguard/openaccess/authentication?version=value`

REST Request Body Contents:

Note: The **oidc_token** name:value pair was introduced in Version 2.0 of the **add authentication** call.

```
{
    "user_name":"value",
```

```
    "password":"value",
    "directory_id":"value",
    "oidc_token":"value"
}
```

add authentication

Name	Type	Required	Version	Description
user_name	string	Required for Version 1.0. Required if the **Support OAuth2 Client Credentials Flow** option is enabled.	1.0 and later	The user's user name, in plain text. Or, if **directory_id** identifies a directory of type **OpenIDConnect** with the **Support OAuth2 Client Credentials Flow** option enabled, then **user_name** is interpreted as the client ID configured for client credentials flow at the provider. In this case, the OIDC authentication flow is not initiated as it would be for a normal user. Instead, the list of users linked with this OIDC directory is searched to find the one whose preferred username matches the provided client ID. For more information about the configuration of a directory to use OAuth2 Client Credentials Flow, refer to “Directories Form (Advanced Sub-tab)” in the *System Administration User Guide*. Also see the footnote below this table.
password	string	Required for Version 1.0. Required if the **Support OAuth2 Client Credentials Flow** and the **Use Introspect Endpoint** options are enabled.	1.0 and later	The user's password, in plain text. Or, if **directory_id** identifies a directory of type **OpenIDConnect** with the **Support OAuth2 Client Credentials Flow** option enabled and the **Use Introspect Endpoint** option enabled, then **password** is interpreted as the client secret. In this case, the **oidc_token** is provided to the introspect endpoint along with the client ID and client secret for validation. For more information about the configuration of a directory to use OAuth2 Client Credentials Flow, refer to “Directories Form (Advanced Sub-tab)” in the *System Administration User Guide*.

add authentication

Name	Type	Required	Version	Description
directory_id	string	yes	1.0 and later	The user's directory ID, as a string. To get a list of available directory IDs, refer to get directories on page 68.
oidc_token	string	Not available for Version 1.0. For Version 2.0 and later, you must provide either the **user_name** and **password** *or* the **oidc_token**.	2.0 and later	An OpenID Connect access token. Introduced in Version 2.0 of the **add authentication** call.
operation_ description	string	no	1.0 and later	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	1.0 and later	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

Notes: *When an OAuth2 token is included in an **add authentication** request:*

If a user_name value is included, OnGuard assumes it is an OAuth2 Client Credentials Flow and uses the following validation logic:

- **If the “Use Introspect Endpoint” option is enabled,** then the token is expected to be opaque, and the configured **Introspect Address** is used to pass the token, along with the client ID and client secret, to the third-party identity provider for validation. As part of the validation process, it is expected that the client ID is also validated by the third-party identity provider.
- **If the “Use Introspect Endpoint” option is *not* enabled,** then the token is expected to be a JSON Web Token that contains the client ID. As part of the validation, the configured **Client ID Claim Name** is required, and the client ID in the token is validated to ensure that it matches the client ID included in the **add authentication** request. The configured **Metadata Address** is required to obtain the signing key data. The token is then validated locally by comparing its content against the configured options (**Validate Audience**, **Validate Issuer**, **Validate Algorithms**, **Validate App Role**, **Validate Scope**).
- If the token is valid, the **add authentication** request is successful.

If a user_name value is 'empty', OnGuard assumes it is *not* an OAuth2 Client Credentials Flow, and validates the token as a standard OIDC **user authentication** request.

add authentication response

Name	Type	Version	Description
session_token	string	1.0 and later	The authentication token, which is returned with a successful response.
password_ expiration_ time	datetime (string)	1.0 and later	This represents the time when the user password will expire, in UTC time. The client should use this information to change password as needed. For example: `2016-10-07T22:05:02+00:00`. This only exists if the user logged in with internal account and the password expiration policy is enabled.
token_expiration_time	datetime (string)	1.0 and later	This represents the time when the authenticated token will expire, in UTC time. The client should use this information to re-authenticate as needed. For example: `2016-10-07T22:05:02+00:00`
version	string	1.0 and later	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.
warning	string	1.0 and later	If present, contains additional information that might be useful to the user even though the authentication was successful. For example, password expiration information would be contained here. For more information, refer to Warning List on page 352.

delete authentication

Logs a user out of the LS OpenAccess service by invalidating the token and removing the user from its internal map.

Notes: Every **add authentication** call should have a matching **delete authentication** call. Not calling **delete authentication** consumes resources unnecessarily over time.

It is also recommended that you keep a session token available (that is, do not **delete authentication**) until you no longer need to execute commands for at least several minutes. In other words, avoid rapidly adding and then deleting authentication within seconds. This can happen if you have a snippet of code that executes very frequently, and that code adds and then deletes authentication. Consider keeping the authentication token and executing the snippet frequently, and then deleting authentication only if the code won't execute again for another 1 or 2 minutes.

REST Request URL: `DELETE /api/access/onguard/openaccess/authentication?version=value`

get session

Retrieves session data for a session token.

REST Request URL: `GET /api/access/onguard/openaccess/session?version=value`

get session response

Name	Type	Description
session_token	string	The authentication token, which is returned with a successful response.
token_ expiration_time	datetime (string)	The time the token will expire, in UTC time. For example: `2016-10-07T22:05:02+00:00`
token_start_time	datetime (string)	The time the token was first issued, in UTC time. For example: `2016-10-07T22:05:02+00:00`
user_id	string	The user's ID, as a string.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get identity_provider_url

Gets the URL that users authenticating with a third-party OpenID Connect provider should be directed to in their browsers.

REST Request URL: `GET /api/access/onguard/openaccess/identity_provider_url?version=value&directory_id=value&redirect_url=value&response_mode=value`

get identity_provider_url

Name	Type	Required	Description
directory_id	string	yes	The directory ID of the selected identity provider. Must refer to an OpenId Connect directory.
redirect_url	string	yes	The URL to which the identity provider should send its response.
response_mode	string	yes	The mode the identity provider should use to respond. Valid values are "form_post" and "fragment". "form_post" causes the identity provider to respond with an HTTP POST to the redirect_url, with the content in the message body. "fragment" will contain the response in the redirect URL.

get identity_provider_url response

Name	Type	Description
url	string	The URL to send the user to for authentication.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Receive Events

get event_subscriptions

Retrieves event subscriptions, and details about the subscriptions. Non-System Account (SA) users can only retrieve their own event subscriptions.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/event_subscriptions?version=value`

get event_subscriptions

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
page_number	int32	no	The page number to be returned when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.

get event_subscriptions

Name	Type	Required	Description
order_by	string	no	A field or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by created_date. Fields must be valid properties of the requested object type. For more information, refer to Additional order_by Details on page 75.

Additional order_by Details

When using order_by to specify that a field is sorted in descending order, add a minus character ("-") in front of the field name. Without the minus character, the field will be sorted in ascending order. Also, different fields can be sorted differently. For example, to sort created_date in descending order and message_broker_hostname in ascending order:

```
GET /api/access/onguard/openaccess/event_subscriptions?
page_number=1&page_size=20&
order_by=-created_date,message_broker_hostname&version=value
```

get event_subscriptions response

Name	Type	Description
item_list	list	A list of items returned, if instances exist. If a valid order_by parameter was provided in the request, then the list of items is sorted accordingly. If present, each item consists of the properties of the event subscription.
id	int32	The ID of the event subscription to retrieve.
user_id	string	The ID of the user who owns the subscription, as a string.
page_number	int32	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total existing number of instances of the object being requested.
description	string	A description of the subscription.
filter	string	This optional parameter filters the events that are received. If no filter is specified, all events are forwarded to the subscriber. For more information refer to Searching for Objects on page 43 and Using Event Filters with Subscriptions on page 48.

get event_subscriptions response

Name	Type	Description
monitoring_ zone_id	int32	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.
is_durable	boolean	Indicates if this is a durable subscription. Default is “false”. For more information, refer to Durable vs. Transient Event Subscribers on page 48.
message_broker_hostname	string	The hostname of the message broker where the events are published.
message_broker_port	int32	The port of the message broker where the events are published.
requires_secure_connection	boolean	Indicates if an SSL connection should be opened with the message broker where the events are published.
exchange_name	string	The exchange name on the message broker where events will be published.
binding_key	string	The unique binding key with which events will be published on the exchange.
created_date	datetime (string)	The date and time when the subscription was created.
last_updated_ date	datetime (string)	The date and time when the subscription was last updated.
count	int32	The total number of records in the filter result.
queue_name	string	The name of the durable queue on the message broker where events will be published for durable subscriptions. Only included in the response when **is_durable** is true.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get event_subscriptions with id

Retrieves a specific event subscription. Non-System Account (SA) users can only retrieve their own event subscriptions.

REST Request URL: `GET /api/access/onguard/openaccess/event_subscriptions/{id}?version=value`

get event_subscriptions with id

Name	Type	Required	Description
id	int32	yes	The ID of the event subscription to retrieve.

get event_subscriptions with id response

Name	Type	Description
id	int32	The unique subscription ID.
user_id	string	The ID of the user who owns the subscription, as a string.
description	string	A description of the subscription.
filter	string	This optional parameter filters the events that are received. If no filter is specified, all events are forwarded to the subscriber. For more information refer to Searching for Objects on page 43 and Using Event Filters with Subscriptions on page 48.
monitoring_ zone_id	int32	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.
is_durable	boolean	Indicates if this is a durable subscription. Default is “false”. For more information, refer to Durable vs. Transient Event Subscribers on page 48.
message_broker_hostname	string	The hostname of the message broker where the events are published.
message_broker_port	int32	The port of the message broker where the events are published.
requires_secure_connection	boolean	Indicates if an SSL connection should be opened with the message broker where the events are published.
exchange_name	string	The exchange name on the message broker where events will be published.
binding_key	string	The unique binding key with which events will be published on the exchange.
created_date	datetime (string)	The date and time when the subscription was created.

get event_subscriptions with id response

Name	Type	Description
last_updated_ date	datetime (string)	The date and time when the subscription was last updated.
queue_name	string	The name of the durable queue on the message broker where events will be published for durable subscriptions. Only included in the response when **is_durable** is true.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add event_subscriptions

Adds an event subscription.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `POST /api/access/onguard/openaccess/event_subscriptions?version=value`

add event_subscriptions

Name	Type	Required	Description
description	string	no	A description of the subscription.
filter	string	no	This optional parameter filters the events that are received. If no filter is specified, all events are forwarded to the subscriber. For more information refer to Searching for Objects on page 43 and Using Event Filters with Subscriptions on page 48
is_durable	boolean	no	Indicates if this is a durable subscription. Default is “false”. For more information, refer to Durable vs. Transient Event Subscribers on page 48.

add event_subscriptions

Name	Type	Required	Description
monitoring_ zone_id	int32	no	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.

add event_subscriptions response

Name	Type	Description
id	int32	The unique subscription ID.
user_id	string	The ID of the user who owns the subscription, as a string.
description	string	A description of the subscription.
filter	string	This optional parameter filters the events that are received. If no filter is specified, all events are forwarded to the subscriber. For more information refer to Searching for Objects on page 43 and Using Event Filters with Subscriptions on page 48.
monitoring_ zone_id	int32	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.
is_durable	boolean	Indicates if this is a durable subscription. Default is “false”. For more information, refer to Durable vs. Transient Event Subscribers on page 48.
message_bro- ker_hostname	string	The hostname of the message broker where the events are published.

add event_subscriptions response

Name	Type	Description
message_broker_port	int32	The port of the message broker where the events are published.
requires_secure_connection	boolean	Indicates if an SSL connection should be opened with the message broker where the events are published.
exchange_name	string	The exchange name on the message broker where events will be published.
binding_key	string	The unique binding key with which events will be published on the exchange.
created_date	datetime (string)	The date and time when the subscription was created.
last_updated_date	datetime (string)	The date and time when the subscription was last updated.
queue_name	string	The name of the durable queue on the message broker where events will be published for durable subscriptions. Only included in the response when **is_durable** is true.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

modify event_subscriptions with id

Modifies an event subscription. Users other than the System Account (SA) user or SA delegate users can only modify their own event subscriptions. The SA user and SA delegate users can modify all event subscriptions.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `PUT /api/access/onguard/openaccess/event_subscriptions/{id}?version=value`

modify event_subscriptions with id

Name	Type	Required	Description
id	int32	yes	The unique subscription ID.
description	string	no	A description of the subscription.
filter	string	no	This optional parameter filters the events that are received. If no filter is specified, all events are forwarded to the subscriber. For more information refer to Searching for Objects on page 43 and Using Event Filters with Subscriptions on page 48.

modify event_subscriptions with id

Name	Type	Required	Description
monitoring_ zone_id	int32	no	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.

modify event_subscriptions with id response

Name	Type	Description
id	int32	The unique subscription ID.
user_id	string	The ID of the user who owns the subscription, as a string.
description	string	A description of the subscription.
filter	string	This optional parameter filters the events that are received. If no filter is specified, all events are forwarded to the subscriber. For more information refer to Searching for Objects on page 43 and Using Event Filters with Subscriptions on page 48.
monitoring_ zone_id	int32	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.
is_durable	boolean	Indicates if this is a durable subscription. Default is “false”. For more information, refer to Durable vs. Transient Event Subscribers on page 48.
message_bro- ker_hostname	string	The hostname of the message broker where the events are published.

modify event_subscriptions with id response

Name	Type	Description
message_broker_port	int32	The port of the message broker where the events are published.
requires_secure_connection	boolean	Indicates if an SSL connection should be opened with the message broker where the events are published.
exchange_name	string	The exchange name on the message broker where events will be published.
binding_key	string	The unique binding key with which events will be published on the exchange.
created_date	datetime (string)	The date and time when the subscription was created.
last_updated_date	datetime (string)	The date and time when the subscription was last updated.
queue_name	string	The name of the durable queue on the message broker where events will be published for durable subscriptions. Only included in the response when **is_durable** is true.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete event_subscriptions with id

Deletes an event subscription. Users other than the System Account (SA) user and SA delegate users can only delete their own event subscriptions. The SA user and SA delegate users can delete all event subscriptions.

REST Request URL: `DELETE /api/access/onguard/openaccess/event_subscriptions/{id}?version=value`

delete event_subscriptions with id

Name	Type	Required	Description
id	int32	yes	The unique subscription ID.

Manage Instances

get logged_events

Retrieves a page of logged events from the OnGuard database.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/logged_events?version=value`

get logged_events

Name	Type	Required	Description
filter	string	yes	The clause text used to count only those instances that match a given attribute. For example, `firstname="Lisa"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. OpenAccess does not support filtering with the following properties: • `EVENT_SOURCE_NAME` • `CARDHOLDER_FIRST_NAME` • `CARDHOLDER_LAST_NAME` • `DEVICE_NAME` • `SUBDEVICE_NAME` • `ACCESS_RESULT` • `CARDHOLDER_ENTERED` • `DURESS` • `ALARM_ACK_BLUE_CHANNEL` • `ALARM_ACK_GREEN_CHANNEL` • `ALARM_ACK_RED_CHANNEL` • `ALARM_BLUE_CHANNEL` • `ALARM_GREEN_CHANNEL` • `ALARM_RED_CHANNEL` For more information refer to Searching for Objects on page 43.
monitoring_ zone_id	int32	no	A field for filtering events generated by devices assigned to a specific monitoring zone. If a monitoring zone ID is not provided, then OpenAccess returns only events assigned to the same monitoring zone in which the user belongs. If the logged-in user provides a monitoring zone ID to a monitoring zone to which the user does not have access, then OpenAccess returns the error `Invalid property monitoring_zone_id`.
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.

get logged_events

Name	Type	Required	Description
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
order_by	string	no	A field or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by created_date. Fields must be valid properties of the requested object type. For more information, refer to Additional order_by Details on page 75.

get logged_events response

Name	Type	Description
ack_status	int32	The latest value of ack status from the ALARM_ACK_HISTORY table, filtered by serial_number. If the ALARM_ACK_HISTORY for a particular serial_number contains more than one element, then the element with the latest time stamp is shown. If ALARM_ACK_HISTORY does not contain elements with a serial_number, then this property is not shown. The status of the alarm, with possible values: 0: Not acknowledged 1: Acknowledged 2: Acknowledged with note 3: Marked in-progress Elements are shown using the Lnl_AlarmAckHistory structure.
active_alarm	boolean	If true, indicates that the alarm is configured to be defined as **active**, a designation that can be used in different ways by the receiving client. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
aggregate_alarm	boolean	If true, indicates that the alarm is configured so that all alarms of the same type are combined into a single alarm, typically with an incrementing count of how many alarms of this type have been received. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
alarm_ack_blue_channel	int32	The blue component of the RGB color for the alarm after it is acknowledged (0 to 255).
alarm_ack_green_channel	int32	The green component of the RGB color for the alarm after it is acknowledged (0 to 255).
alarm_ack_red_channel	int32	The red component of the RGB color for the alarm after it is acknowledged (0 to 255).

get logged_events response

Name	Type	Description
alarm_blue_ channel	int32	The blue component of the RGB color for the alarm (0 to 255).
alarm_green_ channel	int32	The green component of the RGB color for the alarm (0 to 255).
alarm_red_ channel	int32	The red component of the RGB color for the alarm (0 to 255).
alarm_id	int32	The alarm ID for the event. The client can use this property for alarm aggregation.
alarm_priority	int32	Alarm priority (0 to 255).
access_result	int32	The level of access that was granted, resulting from reading the card. 0: Other 1: Unknown 2: Granted 3: Denied 4: Not Applicable
asset_id	int32	Asset (where available) that caused the event.
badge_extend- ed_id	string	Extended identifier of the card that caused the event.
badge_id	int64	Card (where available) that caused the event.
badge_id_str	string	A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off. **Note:** This property is only returned when **get instances** is called with Version 1.2 or later.
badge_ issue_code	int32	Issue code of the card that caused the event.
can_change_ response	boolean	If true, indicates that the alarm is configured to allow the operator to change the response text after the alarm has been acknowledged. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
cardholder_ entered	boolean	True if entry was made by the cardholder.
cardholder_ first_name	string	The first name of the cardholder.
cardholder_key	int32	Internal identifier of the person who is assigned the badge at the time of the access event. See Lnl_Person.ID.
cardholder_ last_name	string	The last name of the cardholder.

get logged_events response

Name	Type	Description
controller_id	int32	Controller at which the event occurred. Key field. Reference to Lnl_Panel ID.
controller_name	string	The name of the controller at which the event occurred.
count	int32	The number of logged events returned.
description	string	Description of the event.
device_id	int32	Device at which the event occurred (for example, Lnl_Reader, Lnl_AlarmPanel, etc.).
display_alarm	boolean	If true, indicates that the alarm is configured to be displayed. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
display_map	boolean	If true, indicates that the alarm is configured so that upon receipt, a map containing the alarm's location is shown. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
do_not_delete_ on_acknowledge	boolean	If true, the alarm is not deleted from the client view after being acknowledged. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
duress	boolean	True if this card access indicates an under duress/emergency state.
event_type	int32	Event type (for example, Duress, System, etc.). Corresponds to Lnl_EventSubtypeDefinition.TypeID and LnlEventType.ID.
event_source_ name	string	The name of the device at which the event occurred.
event_subtype	int32	Event sub-type (for example, Granted, Door Forced Open, etc.). Corresponds to Lnl_EventSubtypeDefinition.SubTypeID.
event_text	string	Text associated with the event.
must_ acknowledge	boolean	If true, the alarm must be acknowledged before it is cleared. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
must_ add_notes_on_ acknowledge	boolean	If true, indicates that the alarm is configured to require a response in the **Notes** field to acknowledge it. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
must_login_on_ acknowledge	boolean	If true, indicates that the alarm is configured to require that the operator log in to acknowledge it. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
must_mark_in_ progress	boolean	If true, the alarm must be marked in progress before it is cleared. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.

get logged_events response

Name	Type	Description
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
print_alarm	boolean	If true, indicates that the alarm is configured to be printed. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
serial_number	int32	Serial number of the event. Key field.
segment_id	int32	Segment where the event occurred.
show_cardholder	boolean	If true, indicates that the alarm is configured so that upon receipt, a cardholder view for the badge specified in the alarm is shown. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
subdevice_id	int32	Secondary device at which the event occurred (for example, Lnl_Input).
timestamp	string	Time when the event occurred.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total existing number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.
video_ verification	boolean	If true, indicates that the alarm is configured so that upon receipt, a video verification view is shown. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.
visual_ notification	boolean	If true, indicates that the alarm is configured to provide a visual notification to the user upon receipt. For more information, refer to “Alarm Definitions Form” in the *System Administration User Guide*.

get types

Retrieves a list of types available via the LS OpenAccess service.

REST Request URL: `GET /api/access/onguard/openaccess/types?version=value`

get types response

Name	Type	Description
types	map	A map of type names to parent type names. All types ultimately derive from "Lnl_Element", except for "Lnl_Element" itself, which will have an empty string as its parent type name.
total_items	int32	The total number of types that are exposed to the user and returned in the types map.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get type

Retrieves information for a specific type.

REST Request URL: `GET /api/access/onguard/openaccess/type?type_name=value&version=value`

get type

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
type_name	string	yes	The name of the type for which to retrieve information.

get type response

Name	Type	Description
type_name	string	The type name.
properties	list	The properties of the type. See get type response: properties list on page 89.

get type response

Name	Type	Description
access	string	Indicates whether the type is view only, read only, or editable. Possible return values: • **View**: Indicates the user cannot change the type. • **Read**: Indicates the type can be added or deleted. • **Edit**: Indicates the type can be added, modified, or deleted.
methods	list	The methods available for this type. See get type response: methods map on page 90.
display_name	string	When provided via the object name of a User Defined Field (UDF) in FormsDesigner, the display_name attribute is the user-friendly name of the item. For more information, refer to Features and Limitations on page 46. Also refer to the “Field Properties Folder – General Settings Form” section in the *FormsDesigner User Guide*.
display_groups	list	Includes a list of user-defined and name attribute that follows the tab order specified in FormsDesigner.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get type response: properties list

Name	Type	Description
name	string	The name of the property.
type	string	The type of the property.
access	string	Indicates whether the property is view only, read only, or editable. Possible return values: • **View**: Indicates the user cannot change the property. • **Read**: Indicates the property value can be specified during add only. • **Edit**: Indicates the property value can be changed at any time.
is_key	boolean	Indicates if the property is a key property.
is_required	boolean	Indicates if the property is required.
max_length	int32	The maximum length of the string, datetime or binary property. For datetime properties `max_length = 0` means no length control.
default_value	string	A default value of the property.
possible_values	map	A map of numerical keys to string values. For example: (0, "Zero"; 1, "One")

get type response: properties list

Name	Type	Description
display_name	string	When provided via the object name of a User Defined Field (UDF) in FormsDesigner, the display_name attribute is the user-friendly name of the item. For more information, refer to Features and Limitations on page 46. Also refer to the "Field Properties Folder – General Settings Form" section in the *FormsDesigner User Guide*.
display_ attributes	map	Displays the following attributes that describe the behavior of user-defined fields: • **is_password**: If enabled, the password is masked as it is entered into a password field. • **is_searchable**: If enabled, the user can search on this property. **Note:** You cannot search on encrypted text or password fields. • **permission**: Indicates the field's permissions. For more information, refer to Data Classes on page 223. • **template**: Specifies a template used to ensure the integrity of data entered into the field.

get type response: methods map

Name	Type	Description
name	string	The name of the method.
in_parameters	map	The parameters expected to be sent along with the execution request of the method. This can be empty. See get type response: method parameter map on page 90.
out_parameters	map	The parameters that represent the result of the method execution. This can be empty.

get type response: method parameter map

Name	Type	Description
name	string	The name of the parameter.
type	string	The type of the parameter.

get count

Used to retrieve the number of existing instances of a given object type.

REST Request URL: `GET /api/access/onguard/openaccess/count?type_name=value&filter=value&version=value`

get count

Name	Type	Required	Description
type_name	string	yes	A string representing the name of the type for which instances will be counted. For example, `Lnl_Cardholder`.
filter	string	no	The clause text used to count only those instances that match a given attribute. For example, `firstname="Lisa"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. For more information refer to Searching for Objects on page 43.

get count response

Name	Type	Description
total_items	int32	The total number of instances of the object type being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get instances

Retrieves instances of a particular type based on the client-supplied filter.

When using this call for types with binary properties (Lnl_MultimediaObject), the binary data is returned base64 encoded.

Notes: You must use Version 1.3 or later of this method if you need support for *BadgeID_str*, *badge_id_str*, *badge_key_str*, *CardNumber_str*, or *AssignedBadgeID_str*.

Versions 1.4, 1.5, and 1.6 of this method support string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/instances?page_number=value&page_size=value&order_by=value&type_name=value&filter=value&version=value`

Note: Page_number and page_size are optional. The default page_number = 1, and the default page_size = 20. Paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100. To preserve system

performance such as when using multimedia objects, you might need to choose a page size smaller than 100.

get instances

Name	Type	Required	Description
type_name	string	yes	The name of the type being added. For example, `Lnl_Cardholder`.
filter	string	no	The filter used to retrieve instances. For example, `Lastname = "Smith" and Firstname = "Lisa"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. For more information refer to Searching for Objects on page 43.
page_number	int32	no	The page number to be returned when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
order_by	string	no	A field or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by key field(s). Fields must be valid properties of the requested object type. For more information, refer to Additional order_by Details on page 92.

Additional order_by Details

For Lnl_AlarmDefinition, you could pass `Priority,Description` (or `Priority , Description` because spaces are ignored). Results would be ordered by Priority (ALARM.ALPRIORITY) followed by Description (ALARM.ALDESCR). In general, filtering behaves the same as queries behave in System Administration.

Use care when selecting which values you specify with your order_by, as the request might take too long to fulfill. This is a problem if you order_by very large classes, such as Lnl_LoggedEvent (Lnl_LoggedEvent on page 274), which might result in a timeout error. For more information, refer to Error Messages on page 349.

In general, using the default order_by works well because key fields are optimized for performance through the use of an index. If you order_by fields that are not indexed and are large classes, performance might suffer.

When using order_by to specify that a field is sorted in descending order, add a minus character ("-") in front of the field name. Without the minus character, the field will be sorted in ascending order. Also, different fields can be sorted differently. For example, to sort lastname in descending order and firstname in ascending order:

```
GET /api/access/onguard/openaccess/
instances?page_number=1&page_size=20&
order_by=-lastname,firstname&type_name=Lnl_Cardholder&version=value
```

Note: The **order_by** parameter is not supported for association data classes.

get instances response

Name	Type	Description
bulk_export	boolean	Exports large datasets quickly, as .csv files. For more information, refer to get instances (bulk export) additional parameters on page 94.
page_number	int32	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total existing number of instances of the object being requested.
count	int32	The total number of records in the filter result.
item_list	list	A list of items returned if instances exist. If a valid order_by parameter was provided in the request, then the list of items is sorted accordingly. If present, each item consists of type_name and property_map.
type_name	string	The name of the type being returned.
property_value_map	map	This is a map where the key is property name and the value is the actual property value.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get instances (bulk export) additional parameters

To improve the reading of very large amounts of configuration data, OnGuard 8.3 introduced the "bulk export" method of downloading data through OpenAccess. This allows you to use all existing **get instances** queries, but unlike the standard response in the form of an HTTP message containing a body in the JSON format, a bulk export query results in a file in the **.csv** format. With bulk export, the client can download very large amounts data at least 50 times faster than with other download methods (for example, downloading 1 million Lnl_Cardholder records with 30 columns), and can be even faster if the number of columns is limited by using the **select** query parameter.

REST Request URL: GET `/api/access/onguard/openaccess/instances?bulk_export=true&type_name=value&chunk_size=value&select=value1,..,valueN&null_value_writing=value&data_separator=value&binary_encoding=value&sql_query_timeout=value&max_thread_count=value&version=value`

The user can adjust the maximum size, content, and processing method of the bulk export file with the following query parameters of the REST API call:

Note: You must use version 1.9 or later of this method if you need bulk export support. Parameters from this API call have higher priority over the parameters specified in the **ACS.ini** file, meaning that ACS.ini parameter values will be replaced by this REST API call's parameters. All parameters described below are optional and ignored if **bulk_export=false**. Default values and ranges are described in Bulk Export Properties on page 27.

get instances (bulk export) parameters

Name	Type	Description
chunk_size	int32	Maximum size of a single result file, expressed in MiB (the size applies to the resulting file before data compression). Unlimited if set to 0.
select	string	List of fields to download, separated by commas. For example, `"select=personid,lastchanged"`.
null_value_writing	boolean	false =columns with null value will be skipped. true = columns with null value will be marked as <null>.
data_separator	string	One or more special characters used to separate fields in a line. The separator should be a character or group of characters that are different from the characters appearing in the read data.
binary_encoding	string	Encoding method for binary data (base64, hex).
output_compression	boolean	Allows you to enable additional .gzip compression for output files to reduce file size and time required to download files.
sql_query_timeout	int32	Timeout for SQL query request in seconds. Should be used to extend the timeout in case of large datasets and long-lasting queries.
max_thread_count	int32	Number of tasks used to process a single request.

Using the bulk export feature

To use the bulk export feature, add the **bulk_export=true** parameter to the standard **get instances** query. Because downloading large amounts of data might involve long processing times, bulk export

uses the task queuing option by default (the **queue=true** parameter is enabled implicitly when **bulk_export=true** is used in the query). This means that after sending the request, an immediate response is received containing the task ID. This allows us to send subsequent queries about the status of the task. Example of the bulk export call:

```
https://127.0.0.1:8080/api/access/onguard/openaccess/
instances?queue=true&output_type=file&type_name=lnl_cardholder&
version=1.9
```

When the bulk export file is ready for download, the response to the task status query includes the ID of each page, which must be used again in the **queue** API call to extract the file name (the **output_file_name** parameter) and the number of records contained in the file (the **output_count** parameter). For example:

```
"property_value_map": {
      "output_count": 1000000,
      "output_file_name": "6597515_Lnl_Badge_
      lvTD7y6Wrsx…gmlns9Gn_Z6QV8bQSkZsRqAJgpVY7_0001.csv"
  }
```

The result **.csv** file will be available to download at:

```
<OpenAccess server and port>/api/access/onguard/openaccess/files/
<result file name>
```

For example:

```
https://127.0.0.1:8080/api/access/onguard/openaccess/files/
9351593_Lnl_Badge_hzcprAxLxR127t8bInohd_0001.csv
```

The file can be downloaded using any application that supports HTTP transfer. The resulting file should be downloaded immediately when it is ready because the file sharing time by the OpenAccess server is limited. Files resulting from the bulk export feature are deleted from the disk after a specified period of time to free up space for files generated by subsequent queries.

Bulk export .csv file

Individual lines of the file contain subsequent database records. Individual elements in each line are separated by separator characters. The fields returned in the **.csv** file might differ from the fields returned for a standard query. For example, boolean fields return values of 0 or 1 (instead of true or false). If **output_compression=true** is used in the request, every resulting **.csv** file is compressed to the **.gzip** formatted archive. In this case, output files have **.csv.gz** extension.

Note: UTF-8 encoding is used to write **.csv** file content.

get badge print_request

Returns the status of the request to print a badge.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/badge/{badge_print_request_id}/print_request?version=value`

get badge print_request

Name	Type	Required	Description
badge_print_request_id	string	yes	Represents a GUID that is system generated. Each print request has a unique id.

get badge print_request response

Name	Type	Description
badgekey	int32	The unique identifier of the badge assigned to a person. For more information, refer to Lnl_Badge on page 242.
badge_print_request_id	string	Represents a GUID that is system generated. Each print request has a unique id.
message	string	Only applies to error messages returned from the badge printing service.
status	string	Internal system codes indicating the status of the badge printing request as it is processed by the print service. Possible statuses: • Pending • Received • Waiting_for_printer_access • Printing • Completed • Completed_skipped_errors • Aborted_fatal_error • Canceled by user
submitted_at	datetime	Represents when the request was sent to the print service.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add badge print_request

Submits a print request to print the badge.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `POST /api/access/onguard/openaccess/badge/{badgekey}/print_request?version=value`

add badge print_request

Name	Type	Required	Description
badgekey	int32	yes	The unique identifier of the badge assigned to a person. For more information, refer to Lnl_Badge on page 242.

add badge print_request

Name	Type	Required	Description
print-request	JSON	no	Message body, in JSON format.
printer_name	string	no	This property is added to version 1.3 and later. The printer name corresponds to the printers returned from the **GET badge_printers** API call. If the **printer_name** property is not provided or is empty, the default printer assigned to the workstation (if any) is used.
workstation	string	no	The workstation corresponding to the printers returned from the **GET /badge_printers** API call. For more information, refer to get badge_printers on page 101.

add badge print_request response

Name	Type	Description
badgekey	int32	The unique identifier of the badge assigned to a person. For more information, refer to Lnl_Badge on page 242.
badge_print_request_id	string	Represents a GUID that is system generated. Each print request has a unique id.
message	string	Only applies to error messages returned from the badge printing service.
status	string	Internal system codes indicating the status of the badge printing request as it is processed by the print service. Possible statuses: • Pending • Received • Waiting_for_printer_access • Printing • Completed • Completed_skipped_errors • Aborted_fatal_error • Canceled by user
submitted_at	datetime	Represents when the request was sent to the print service.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete badge print_request

Deletes a print request to print the badge that hasn't completed.

REST Request URL: `DELETE /api/access/onguard/openaccess/badge/{badge_print_request_id}/print_request?version=value`

delete bade print_request

Name	Type	Required	Description
badge_print_request_id	string	yes	Represents a GUID that is system generated. Each print request has a unique id.
request body	string	no	Pass an empty request body.

delete badge print_request response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get badge mobile_devices

This method retrieves a list of mobile devices for the person associated with a badge. The list is provided by the mobile credentialing services associated with the badge type of this badge.

Notes: If you are using OpenAccess to issue mobile badges and are behind a network proxy, an error might occur when issuing or managing mobile credentials. To resolve this error, on the server where the LS OpenAccess service is running, change the logon account for the LS OpenAccess service from Local System to a user whose account has the correct proxy settings configured.

Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/badge/{badgekey}/mobile_devices?version=value`

get badge mobile_devices

Name	Type	Required	Description
badgekey	int32	yes	The badgekey of the mobile device assigned to a person. For more information, refer to Lnl_Badge on page 242.

get badge mobile_devices response

Name	Type	Description
total_items	int32	The total existing number of instances.
mobile_device_list	list	A list of mobile devices for the person associated with the badge. See get badge mobile_devices response: mobile_device_list properties on page 99.

get badge mobile_devices response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get badge mobile_devices response: mobile_device_list properties

Name	Type	Description
mobile_device_id	integer	The mobile device's ID.
mobile_device_ description	string	The mobile device's descriptive name.
mobile_device_ active	boolean	Identifies whether or not the mobile device is active.

add badge issue_mobile_credential

This method issues a credential to a mobile device for the person with the given badge.

If you are using OpenAccess to issue mobile badges and are behind a network proxy, an error might occur when issuing or managing mobile credentials. To resolve this error, on the server where the LS OpenAccess service is running, change the logon account for the LS OpenAccess service from Local System to a user whose account has the correct proxy settings configured.

When OnGuard is first installed, the LS Cumulus Connector Service is configured for Manual start. If you want to use Cumulus with OnGuard, you should:

1. Make the above Cumulus configuration changes in OnGuard.
2. Configure the LS Cumulus Connector Service for Automatic start on the workstation that will connect to Cumulus.
3. Start the LS Cumulus Connector service.

The LS Cumulus Connector Service must always be running for OnGuard (and also OpenAccess and WATCH) to make calls to Cumulus. If the service is interrupted, an appropriate error message is shown to the user. When this occurs, you should restart the LS Cumulus Connector Service, and any application calls that make a connection to Cumulus must be executed again.

Notes: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

Version 1.2 and later of this method supports Cumulus mobile credential issuance.

REST Request URL: `POST /api/access/onguard/openaccess/badge/{badgekey}/issue_mobile_credential?version=value`

add badge issue_mobile_credential

Name	Type	Required	Description
badgekey	int32	yes	The unique identifier of the badge for which a mobile credential should be issued. For more information, refer to Lnl_Badge on page 242.

add badge issue_mobile_credential

Name	Type	Required	Description
in_parameter_value_map	map	yes	A list of optional parameters to configure on the issued mobile credential. See add badge issue_mobile_credential: in_parameter_value_map properties on page 100. This parameter is ignored if using Cumulus mobile badge issuance.

add badge issue_mobile_credential: in_parameter_value_map properties

Name	Type	Required	Description
mobile_device_id	string	no	The mobile device's ID.
send_email	boolean	no	Set this value to False to prevent a welcome email from being sent to the cardholder upon issuance of the mobile credential. The default is to send an email.
mobile_issuance_method	string	no	Set this value to "regenerate" to resend the welcome email to a cardholder whose badge already had a mobile credential issued. Not specifying a value, or specifying any other value, causes a new mobile credential to be issued to the given badge. **Note:** The regenerate option is not supported for HID Cumulus and Cumulus Universal credentials. To send the cardholder another invitation, you must issue a new credential.

add badge issue_mobile_credential response

Name	Type	Description
mobile_device_activation_code	string	The activation code to use when issuing a credential to the mobile device. This parameter is returned both when using Cumulus and when using non-Cumulus mobile credential issuance, *except* when using Cumulus BlueDiamond or Cumulus Universal mobile credential issuance, which *does not* return an activation code.
mobile_issuance_message	string	An optional message reported from the credentialing service to indicate additional issuance status information. This parameter is not returned when using Cumulus mobile credential issuance.
mobile_device_pin	string	The personal identification number (PIN) provided by the installer when using the BlueDiamond Toolkit. This parameter is not returned when using Cumulus mobile credential issuance.

add badge issue_mobile_credential response

Name	Type	Description
mobile_device_ for_installer	boolean	If true, indicates that the credential is assigned to an access control system installer using the BlueDiamond Toolkit to install BlueDiamond Mobile readers. This parameter is not returned when using Cumulus mobile credential issuance.
CumulusCredential	map	This parameter is only returned when using Cumulus mobile credential issuance. See add badge issue_mobile_credential response: CumulusCredential properties on page 101.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add badge issue_mobile_credential response: CumulusCredential properties

Name	Type	Description
bundle	string	A GUID that uniquely identifies the Cumulus bundle from which the credential was issued.
card_format	string	Indicates whether the card format for the credential is controller-defined (defined in OnGuard) or custom (defined in Cumulus).
card_number	string	The badge ID of the issued credential.
created	string	The date on which the credential was issued.
expires	string	The date on which the issued credential expires.
external_id	string	Currently unused.
facility_code	int32	The facility code encoded in the issued credential.
id	string	A GUID that uniquely identifies the issued credential in Cumulus.
issued_by	string	A GUID that identifies the OnGuard system for which the credential was issued.
person	string	A GUID that uniquely identifies the cardholder to whom the credential was issued.
revoked_on	string	Not applicable for this method.
status	string	The status of the issued credential (possible values are: PENDING, FAILED, EXPIRED, NOT_SUPPORTED, REVOKED, DELETED, DELIVERED, UNBOUND, ISSUING, ISSUED, or REVOKING, but not all are applicable for this request).
type	string	The type of credential (either HID, BlueDiamond, or Cumulus Universal).

get badge_printers

Retrieves a list of printers available for badge printing.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/badge_printers?version=value&badge_type_id=value&filter=value&page_number=value&page_size=value&order_by=value`

get badge_printers

Name	Type	Required	Description
badge_type_id	int32	no	Represents the badge type id found in the **BadgeType** table.This parameter requires version 1.0 or 1.1 of the **get badge printers** method.
filter	string	no	The search string delimiters used to count only those instances that match a given attribute. Single-quotes will result in an InvalidQuery error. For example, `workstation="myhostname"`. **Note:** You must use double-quotes around the search string. For more information, refer to Searching for Objects on page 43. This parameter requires version 1.2 or later of the **get badge printers** method.
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used in conjunction with **page_size**. Defaults to the first page (1) if not provided, and if provided, must be numeric. This parameter requires version 1.2 or later of the **get badge printers** method.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with **page_number**. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100. This parameter requires version 1.2 or later of the **get badge printers** method.
order_by	string	no	A field or comma-separated list of fields to use for sorting list of printers. If not provided, results are ordered by **badge_type_id**. For more information, refer to Additional order_by Details on page 92. This parameter requires version 1.2 or later of the **get badge printers** method.

get badge_printers

Name	Type	Required	Description
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get badge_printers response

Name	Type	Description
printers	array	An array describing the available printers.
badge_type_id	int32	The badge type ID.
printer_name	string	The printer name, or the network path to the printer.
workstation	string	The workstation associated with the printer. An asterisk (*) indicates the printer is associated with the workstation configured in the **System Administration > System Options > Default Badge Printing Service host**.
is_default	boolean	A flag indicating whether this printer is default in this workstation for the specific badge type. This parameter requires version 1.3 or later of the **get badge printers** method. Version 1.2 or earlier of the **get badge printers** method only returns printers if **is_default = true**.
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with **page_size**. Defaults to the first page (1) if not provided, and if provided, must be numeric. This parameter requires version 1.2 or later of the **get badge printers** method.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with **page_number**. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100. This parameter requires version 1.2 or later of the **get badge printers** method.
total_pages	int32	The total number of pages, given the existing number of instances (**total_items**) and the **page_size** being used. This parameter requires version 1.2 or later of the **get badge printers** method.
total_items	int32	The total existing number of instances of the requested object.

get badge_printers response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Sample JSON Response

```
{
  "printers": [
    {
      "badge_type_id": 1,
      "printer_name": "\\\\PC-2016\\Printer Brand and Model 1",
      "workstation": "*",
      "is_default": "false"
    },
    {
      "badge_type_id": 1,
      "printer_name": "ABC Card Printer",
      "workstation": "PC-2016",
      "is_default": "true"
    }
    {
      "badge_type_id": 1,
      "printer_name": "Microsoft Print to PDF",
      "workstation": "PC-2016",
      "is_default": "false"
    }
  ],
  "total_items": 3,
  "version": "1.3"
}
```

post badge printers

Add a badge printer.

REST Request URL: `POST /api/access/onguard/openaccess/badge_printers?version=value`

post badge printers

Name	Type	Required	Description
badge_type_id	int32	yes	Represents the badge type ID found in the **BadgeType** table.
printer_name	string	yes	The printer name, or the network path to the printer.
workstation	string	yes	The workstation associated with the printer. An asterisk (*) indicates the printer is associated with the workstation configured in **System Administration > System Options > Default Badge Printing Service host**.

post badge printers

Name	Type	Required	Description
is_default	boolean	no	A flag indicating whether this printer is default in this workstation for the specific badge type. A specific workstation and badge type ID combination only has one default printer. This parameter requires version 1.1 or later. Default value (is_default = true) is used if the property is not provided. In version 1.0, it adds the printer with **is_default = true**.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

post badge printers response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put badge printers

Update an existing badge printer.

REST Request URL: `PUT /api/access/onguard/openaccess/badge_printers?version=value`

put badge printers

Name	Type	Required	Description
badge_type_id	int32	yes	Represents the badge type ID found in the **BadgeType** table.
printer_name	string	yes	The printer name, or the network path to the printer.
workstation	string	yes	The workstation associated with the printer. An asterisk (*) indicates the printer is associated with the workstation configured in **System Administration > System Options > Default Badge Printing Service host**.

put badge printers

Name	Type	Required	Description
is_default	boolean	no	A flag indicating whether this printer is default in this workstation for the specific badge type. A specific workstation and badge type ID combination only has one default printer. This parameter requires version 1.1 or later. Default value (is_default = true) is used if the property is not provided. In version 1.0, this parameter updates the printer with **is_default = true**.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put badge printers response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete badge printers

Delete an existing badge printer.

REST Request URL: `DELETE /api/access/onguard/openaccess/badge_printers?version=value`

delete badge printers

Name	Type	Required	Description
badge_type_id	int32	yes	Represents the badge type ID found in the **BadgeType** table.
printer_name	string	yes	The printer name, or the network path to the printer. This parameter requires version 1.1 or later. In version 1.0, the property is not required, and it deletes the printer with **is_default = true**.
workstation	string	yes	The workstation associated with the printer. An asterisk (*) indicates the printer is associated with the workstation configured in S**ystem Administration > System Options > Default Badge Printing Service host.**

delete badge printers

Name	Type	Required	Description
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete badge printers response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add instances

Adds instances of a particular type.

Note: You must use Version 1.1 or later of this method if you need support for *BadgeID_str*, *badge_id_str*, *badge_key_str*, *CardNumber_str*, or *AssignedBadgeID_str*.

REST Request URL: `POST /api/access/onguard/openaccess/instances?version=value`

REST Request Body Contents:

```
{
    "type_name":"value",
    "property_value_map":
    {
        "property_name":value,
        ...
    }
}
```

add instances

Name	Type	Required	Description
type_name	string	yes	The name of the type being added. For example "Lnl_Cardholder".
property_value_map	map	yes	The property name to property value map that represents the instance data to add.
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.

add instances

Name	Type	Required	Description
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

add instances response

Name	Type	Description
type_name	string	The name of the type being added. For example "Lnl_Cardholder".
property_value_map	map	The property name to property value map that represents the instance data of the added object. Only key properties are returned for add instances calls.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

modify instances

Modifies existing instances of a particular type.

Note: You must use Version 1.1 or later of this method if you need support for *BadgeID_str*, *badge_id_str*, *badge_key_str*, *CardNumber_str*, or *AssignedBadgeID_str*.

REST Request URL: `PUT /api/access/onguard/openaccess/instances?version=value`

REST Request Body Contents:

```
{
    "type_name":"value",
    "property_value_map":
    {
        "property_name":value,
        ...
    }
```

```
}
```

modify instances

Name	Type	Required	Description
force	boolean	no	Used to update instances of Lnl_Badge when using version 1.3 or later. OpenAccess will update the credential status in Cumulus if the badge status is changed. If true, indicates that the badge should be updated from the OnGuard database even if an associated mobile credential's status could not be updated on the Cumulus Server. Default = false. If using version 1.0 or 1.2, the instance of Lnl_Badge is updated in the OnGuard database even if an associated mobile credential's status could not be updated on the Cumulus Server.
type_name	string	yes	The name of the type being modified. For example, "Lnl_Cardholder".
property_value_map	map	yes	The property name to property value map that represents the instance data to be modified. **Note:** Key properties must be specified here to resolve the object that will be modified properly.
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, "Viewing a list of cameras." Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: "00000000-0000-0000-0000-000000000000") to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

modify instances response

Name	Type	Description
type_name	string	The name of the type to modify. For example, "Lnl_Cardholder".
property_value_map	map	The property name to property value map that represents the instance data of the modified object. Only key properties are returned for modify instances calls.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

bulk modify instance property

Bulk modifies the value of an instance's property.

REST Request URL: `PUT /api/access/onguard/openaccess/property_bulk_update?version=value`

REST Request Body Contents:

```
{
    "property_name":"value",
    "property_value":"value"
}
```

bulk modify instance property

Name	Type	Required	Description
type_name	string	yes	The name of the type. Currently only "Lnl_User" is supported.
property_name	string	yes	The name of the property. Currently only "PasswordChangeRequired" is supported.
property_value	string	yes	The new property value. For example, input "true" or "false" for property "Lnl_User.PasswordChangeRequired".
id_list	list	no	List of instance IDs in the format [1,2,3,...]. If no list is provided, all instances are modified. For example, if the property is "Lnl_User.PasswordChangeRequired" and no list is provided, all users with internal accounts are modified.

bulk modify instance property response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete instances

Deletes existing instances of a particular type.

REST Request URL: `DELETE /api/access/onguard/openaccess/instances?version=value`

REST Request Body Contents:

```
{
    "type_name":"value",
    "property_value_map":
    {
        "property_name":value,
        ...
    }
```

```
}
```

delete instances

Name	Type	Required	Description
force	boolean	no	Used to delete instances of Lnl_Badge or Lnl_Cardholder when using version 1.1 or later. If true, indicates that the badge or cardholder should be deleted from the OnGuard database even if an associated mobile credential or person could not be deleted from the Cumulus Server. Default = false. If using version 1.0, the instance of Lnl_Badge or Lnl_Cardholder is deleted from the OnGuard database even if an associated mobile credential or person could not be deleted from the Cumulus Server.
operation_ description	string	no	Some OpenAccess calls specify this additional optional parameter, which is a user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	Some OpenAccess calls specify this additional optional parameter, which is an arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
property_value_ map	map	yes	The key property name to key property value map that represents the instance data to be deleted. **Note:** Key properties must be specified here in order to properly resolve the object to be deleted.
type_name	string	yes	The name of the type being deleted. For example "Lnl_Cardholder".

execute_method

Executes a supported method against an existing instance of a particular type. For an example, refer to Chapter 8: Using OpenAccess to Send Alarms to OnGuard on page 337.

Note: You must use Version 1.1 or later of this method if you need support for *BadgeID_str*, *badge_id_str*, *badge_key_str*, *CardNumber_str*, or *AssignedBadgeID_str*.

REST Request URL: `POST /api/access/onguard/openaccess/execute_method?version=value`

REST Request Body Contents:

```
{
    "method_name":"value",
    "type_name":"value",
    "property_value_map":
    {
        "property_name":value,
        ...
    },
    "in_parameter_value_map":
    {
        "property_name":value,
        ...
    }
}
```

execute method

Name	Type	Required	Description
type_name	string	yes	The name of the type being operated upon. For example "Lnl_IncomingEvent".
operation_ description	string	no	User friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports. Some methods make use of this optional parameter, others do not. All methods supporting this parameter will have it stated in their documentation. For more information, refer to Operation Description on page 55.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000- 0000-0000-000000000000”) to indicate a correlation among multiple API calls that achieve a common purpose. Some methods make use of this optional parameter, other do not. All methods supporting this parameter will have it stated in their documentation. For more information, refer to Operation GUID on page 54.
property_value_ map	map	yes	The key property name to key property value map that represents the instance data to be operated on. **Note:** Key properties must be specified here to properly resolve the object on which to execute the method.
method_name	string	yes	The name of the method to be executed. Supported methods are returned in the get type response. For example, “SendIncomingEvent”.

execute method

Name	Type	Required	Description
in_parameter_value_map	map	no	The name/value map of any input parameters to the method.

execute method response

Name	Type	Description
out_parameter_value_map	map	The name/value map of any output of the method.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get cardholders

Performs an advanced cardholder search, optionally searching on badge fields. Returns instances that match the search criteria. For more information, refer to Lnl_Cardholder on page 255.

Note: Versions 1.1 and 1.2 of this method support string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/cardholders?auto_load_badge=value&auto_load_multimedia_object=value&auto_load_access_level=value&auto_load_reader=value&auto_load_timezone=value&auto_load_timezone_interval=value&version=value&page_number=value&page_size=value&order_by=value&cardholder_filter=value&badge_filter=value&has_badges=value&has_photo=value&has_signature=value&has_any_directory_account=value&access_level_filter=value&access_level_list=[value1,value2,...,valueN]&access_level_search_type=value&reader_filter=value&reader_name_list=[value1,value2,...,valueN]&reader_name_list_search_type=value&timezone_name_list=[value1,value2,...,valueN]&timezone_name_list_search_type=value`

get cardholders

Name	Type	Required	Description
access_level_filter	string	no	The filter, based on Access Level properties (for example, `Name="name1"` or `AccessMode=1`). This parameter requires Version 1.2 or later of the **get cardholders** method.
access_level_list	list	no	A list of access level IDs for which to search cardholders. For example: [1,2,3]. This parameter must be used with the access_level_search_type property.

get cardholders

Name	Type	Required	Description
access_level_-search_type	string	no	The type of access level search to apply. This parameter describes how to interpret access_level_list: • **any_of** - Finds cardholders with any of the access levels in access_level_list (at least one). • **none_of** - Finds cardholders with none of the access levels in access_level_list. • **all_of** - Finds cardholders with all of the access levels in access_level_list. • **exactly** - Finds cardholders with exactly the access levels in access_level_list (all of the access levels and no others).
auto_load_badge	boolean	no	A flag indicating whether to load the badges assigned to cardholders in the response. This parameter requires Version 1.2 or later of the **get cardholders** method. Value is assumed **true** by default.
auto_load_multimedia_object	boolean	no	A flag indicating whether to load the multimedia objects (such as cardholder photos and signatures) assigned to cardholders in the response. This parameter requires Version 1.2 or later of the **get cardholders** method. In version 1.2 of the method, the value is assumed **true** by default. Beginning in version 1.3, the value is assumed **false** by default.
auto_load_access_level	boolean	no	A flag indicating whether to load the access levels assigned to cardholders in the response. This parameter requires Version 1.2 or later of the **get cardholders** method. Value is assumed **true** by default.
auto_load_reader	boolean	no	A flag indicating whether to load the readers assigned to access levels in the response. This parameter requires Version 1.2 or later of the **get cardholders** method. Value is assumed **true** by default.
auto_load_timezone	boolean	no	A flag indicating whether to load the timezones assigned to access levels in the response. This parameter requires Version 1.2 or later of the **get cardholders** method. Value is assumed **true** by default.
auto_load_timezone_interval	boolean	no	A flag indicating whether to load the timezone intervals assigned to the timezone. This parameter requires Version 1.2 or later of the **get cardholders** method. Value is assumed **true** by default.

get cardholders

Name	Type	Required	Description
badge_filter	string	no	The filter, based on the badge properties. For more information refer to Searching for Objects on page 43 and Lnl_Badge on page 242.
cardholder_filter	string	no	The filter, based on the cardholder properties. For more information refer to Searching for Objects on page 43 and Lnl_Cardholder on page 255.
has_badges	boolean	no	Boolean search for confirming that the cardholder has a badge. • If **has_badges** = false, cardholders that have no badges are returned as specified by **cardholder_filter**. • If **has_badges** = true, cardholders that have at least one badge are returned as specified by **cardholder_filter**. • If **has_badges** is not specified in the request, cardholders are returned as specified by **cardholder_filter**. • If specifying **has_badges** = false, it cannot be combined with **badge_filter**. **InvalidRequest** error is returned if you specify both.
has_only_ inactive_badges	boolean	no	Boolean search for cardholders having only inactive badges. This parameter requires Version 1.4 or later of the **get cardholders** method. **Note:** Only "true" are shown in search results.
has_photo	boolean	no	Boolean search for confirming that the cardholder has a photo.
has_signature	boolean	no	Boolean search for confirming that the cardholder has a signature.
last_activity_filter	string	no	Filter based on **EventTime** properties from **Lnl_BadgeLastLocation**. This parameter requires Version 1.4 or later of the **get cardholders** method. **Note:** For `EventTime=null`, only cardholders who never used their badges are returned. For `EventTime != null`, only cardholders who used their badges at least once are returned.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.

get cardholders

Name	Type	Required	Description
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
order_by	string	no	A field or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by key field(s). Fields must be valid properties of the requested object type. For more information, refer to Additional order_by Details on page 92.
page_number	int32	no	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
reader_filter	string	no	The filter, based on Reader properties. This parameter requires Version 1.2 or later of the **get cardholders** method.
reader_name_list	list	no	A list of reader names for which to search cardholders. For example: [name1,name2,name3]. This parameter must be used with the reader_name_list_search_type. This parameter requires Version 1.2 or later of the **get cardholders** method.

get cardholders

Name	Type	Required	Description
reader_name_list_search_type	string	no	The type of reader name list search to apply. This parameter describes how to interpret reader_name_list. • **any_of** - Finds cardholders with any of the readers in reader_name_list (at least one). • **none_of** - Finds cardholders with none of the readers in reader_name_list. • **all_of** - Finds cardholders with all of the readers in reader_name_list. • **exactly** - Finds cardholders with exactly the readers in reader_name_list (all of the readers and no others). This parameter requires Version 1.2 or later of the **get cardholders** method.
sum_badge_filter	boolean	no	A flag indicating whether to join cardholder and badge filter with the OR instead of AND operator. This parameter requires version 1.3 or later of the **get cardholders** method.
timezone_name_list	list	no	A list of timezone names for which to search cardholders. For example: [name1,name2,name3]. This parameter must be used with the timezone_name_list_search_type. This parameter requires Version 1.2 or later of the **get cardholders** method.
timezone_name_list_search_type	string	no	The type of timezone name list search to apply. This parameter describes how to interpret timezone_name_list. • **any_of** - Finds cardholders with any of the timezones in timezone_name_list (at least one). • **none_of** - Finds cardholders with none of the timezones in timezone_name_list. • **all_of** - Finds cardholders with all of the timezones in timezone_name_list. • **exactly** - Finds cardholders with exactly the timezones in timezone_name_list (all of the timezones and no others). This parameter requires Version 1.2 or later of the **get cardholders** method.

get cardholders response

Name	Type	Description
count	int32	The total number of records in the filter result.
item_list	list	A list of Lnl_Cardholder items returned, if instances exist. If a valid order_by parameter was provided in the request, then the list of items is sorted accordingly. If present, each item consists of a property_value_map for the cardholder being returned (see footnote below).
page_number	int32	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
property_value_map*	map	This is a map where the key is property name and the value is the actual property value. Each property_value_map entry in the item_list represents a single cardholder. For more information, refer to Lnl_Cardholder on page 255. Each cardholder can also contain nested sub-objects, as described in the footnote below. For more information about those objects, refer to Lnl_Badge on page 242, Lnl_MultimediaObject on page 281, Lnl_AccessLevel on page 224, Lnl_Reader on page 294, Lnl_Timezone on page 312, and Lnl_TimezoneInterval on page 313.
total_items	int32	The total existing number of instances of the object being requested.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

* The property_value_map in **get cardholders response** can optionally contain badges and multimedia objects, depending on how the following boolean values were configured in the request:

- auto_load_badge
- auto_load_multimedia_object
- auto_load_access_level
- auto_load_reader
- auto_load_timezone
- auto_load_timezone_interval

It's important to understand the following object hierarchy:

Cardholders				
	• Multimedia_objects			

	• Badges			
		• Access levels		
			• Reader assignment	
				• Readers
				• Timezones
				• Timezone intervals

In the **get cardholders** request, enabling a boolean value for an object type without having enabled the boolean value for all higher object types will generate an error. For example, if you enable auto_load_access_level without having also enabled auto_load_badge, then an error is generated.

get visitors

Performs an advanced visitor search, optionally searching on badge fields. Returns instances that match the search criteria. For more information, refer to Lnl_Visitor on page 324.

Note: Version 1.0 of this method supports string encoding.

REST Request URL: `GET /api/access/onguard/openaccess/visitors?auto_load_badge=value&auto_load_multimedia_object=value&auto_load_access_level=value&auto_load_reader=value&auto_load_timezone=value&auto_load_timezone_interval=value&version=value&page_number=value&page_size=value&order_by=value&visitor_filter=value&badge_filter=value&has_badges=value&has_photo=value&has_signature=value&has_any_directory_account=value&access_level_filter=value&access_level_list=[value1,value2,...,valueN]&access_level_search_type=value&reader_filter=value&reader_name_list=[value1,value2,...,valueN]&reader_name_list_search_type=value&timezone_name_list=[value1,value2,...,valueN]&timezone_name_list_search_type=value`

get visitors

Name	Type	Required	Description
visitor_filter	string	no	The filter, based on the visitor properties. For more information refer to Searching for Objects on page 43 and Lnl_Visitor on page 324.
badge_filter	string	no	The filter, based on the badge properties. For more information refer to Searching for Objects on page 43 and Lnl_Badge on page 242.
has_badges	boolean	no	Boolean search for confirming that the visitor has a badge.
has_photo	boolean	no	Boolean search for confirming that the visitor has a photo.

get visitors

Name	Type	Required	Description
has_signature	boolean	no	Boolean search for confirming that the visitor has a signature.
access_level_filter	string	no	The filter, based on Access Level properties.
access_level_list	list	no	A list of access level IDs for which to search visitors. For example: [1,2,3]. This parameter must be used with the access_level_search_type property.
access_level_-search_type	string	no	The type of access level search to apply. This parameter describes how to interpret access_level_list: • **any_of** - Finds visitors with any of the access levels in access_level_list (at least one). • **none_of** - Finds visitors with none of the access levels in access_level_list. • **all_of** - Finds visitors with all of the access levels in access_level_list. • **exactly** - Finds visitors with exactly the access levels in access_level_list (all of the access levels and no others).
auto_load_badge	boolean	no	A flag indicating whether to load the badges assigned to visitors in the response. The value is assumed **true** by default.
auto_load_ multimedia_object	boolean	no	A flag indicating whether to load the multimedia objects (such as photos and signatures) assigned to visitors in the response. In version 1.0 of this method, the value is assumed **true** by default. Beginning in version 1.1, the value is assumed **false** by default.
auto_load_ access_level	boolean	no	A flag indicating whether to load the access levels assigned to visitors in the response. The value is assumed **true** by default.
auto_load_reader	boolean	no	A flag indicating whether to load the readers assigned to access levels in the response. The value is assumed **true** by default.
auto_load_ timezone	boolean	no	A flag indicating whether to load the timezones assigned to access levels in the response. The value is assumed **true** by default.
auto_load_ timezone_interval	boolean	no	A flag indicating whether to load the timezone intervals assigned to the timezone. The value is assumed **true** by default.
reader_filter	string	no	The filter, based on Reader properties.

get visitors

Name	Type	Required	Description
reader_name_list	list	no	A list of reader names for which to search visitors. For example: [name1,name2,name3]. This parameter must be used with the reader_name_list_search_type.
reader_name_list_search_type	string	no	The type of reader name list search to apply. This parameter describes how to interpret reader_name_list. • **any_of** - Finds visitors with any of the readers in reader_name_list (at least one). • **none_of** - Finds visitors with none of the readers in reader_name_list. • **all_of** - Finds visitors with all of the readers in reader_name_list. • **exactly** - Finds visitors with exactly the readers in reader_name_list (all of the readers and no others).
sum_badge_filter	boolean	no	A flag indicating whether to join visitor and badge filter with the OR instead of AND operator. This parameter requires version 1.1 or later of the **get visitors** method.
timezone_name_list	list	no	A list of timezone names for which to search visitors. For example: [name1,name2,name3]. This parameter must be used with the timezone_name_list_search_type.
timezone_name_list_search_type	string	no	The type of timezone name list search to apply. This parameter describes how to interpret timezone_name_list. • **any_of** - Finds visitors with any of the timezones in timezone_name_list (at least one). • **none_of** - Finds visitors with none of the timezones in timezone_name_list. • **all_of** - Finds visitors with all of the timezones in timezone_name_list. • **exactly** - Finds visitors with exactly the timezones in timezone_name_list (all of the timezones and no others).
page_number	int32	no	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.

get visitors

Name	Type	Required	Description
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
order_by	string	no	A field or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by key field(s). Fields must be valid properties of the requested object type. For more information, refer to Additional order_by Details on page 92.

get visitors response

Name	Type	Description
page_number	int32	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total existing number of instances of the object being requested.
count	int32	The total number of records in the filter result.
item_list	list	A list of Lnl_Visitor items returned, if instances exist. If a valid order_by parameter was provided in the request, then the list of items is sorted accordingly. If present, each item consists of a property_value_map for the visitor being returned (see footnote below).
property_value_map*	map	This is a map where the key is property name and the value is the actual property value. Each property_value_map entry in the item_list represents a single visitor. For more information, refer to Lnl_Visitor on page 324. Each visitor can also contain nested sub-objects, as described in the footnote below. For more information about those objects, refer to Lnl_Badge on page 242, Lnl_MultimediaObject on page 281, Lnl_AccessLevel on page 224, Lnl_Reader on page 294, Lnl_Timezone on page 312, and Lnl_TimezoneInterval on page 313.

get visitors response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

* The property_value_map in **get visitors response** can optionally contain badges and multimedia objects, depending on how the following boolean values were configured in the request:

- auto_load_badge
- auto_load_multimedia_object
- auto_load_access_level
- auto_load_reader
- auto_load_timezone
- auto_load_timezone_interval

It's important to understand the following object hierarchy:

Visitors				
	• Multimedia_ objects			
	• Badges			
		• Access levels		
			• Reader assignment	
				• Readers
				• Timezones
				• Timezone intervals

In the **get visitors** request, enabling a boolean value for an object type without having enabled the boolean value for all higher object types will generate an error. For example, if you enable auto_load_access_level without having also enabled auto_load_badge, then an error is generated.

get visitevent_status

This method gets a list of visit events and each visit's status.

REST Request URL: `GET /api/access/onguard/openaccess/visitevent_status?filter=value&status_list=[value1,...,valueN]&version=value`

get visitevent_status

Name	Type	Required	Description
filter	int32	no	The filter, based on LnL_VisitEvent properties. For more information, refer to Lnl_VisitEvent on page 323.
visitevent_status_list	list	no	A list of status names for which to search visit events. For example: `["active", "finished"]`. Allowed values: `"scheduled", "active", "finished"`.
visit_status_list	list	no	A list of visit status names for which to search visit events. List items are processed with the logical OR operator. For example: `["late","overstayed"]` means looking for visit events containing at least one visit with `"late"` status OR at least one visit with `"overstayed"` status. Allowed values: `"scheduled", "late", "active", "overstayed" ,"finished"`.
page_number	int32	no	The page number of the requested subset (page) of instances returned. Same as the corresponding input parameter, or the default value if not provided as input.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, `"Viewing a list of cameras."` Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	string	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: `"00000000-0000- 0000-0000-000000000000"`) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get visitevent_status

Name	Type	Required	Description
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get visitevent_status response

Name	Type	Description
count	int32	The total number of records in the filter result.
item_list	list	A list of status summary. Each summary in the list has following properties: • id (the visit event ID) • status (the visit event status) • visits_count • visits_scheduled • visits_late • visits_active • visits_overstayed • visits_finished
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get video_recorders

This method retrieves one page of the list of all video recorders configured in the OnGuard system.

Notes: This method replaces the previously existing get instances call for the type Lnl_VideoRecorder, which retrieved only LenelS2 NVR video recorders. This method retrieves all recorders, regardless of type.

Version 1.1 and later of this method supports filtering as discussed in get instances on page 91.

Version 1.2 and later of this method supports string encoding, which is helpful in preventing malicious code from being run by the client browser. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

Version 1.3 of this method supports retrieving gateway information linked to the recorder, and recorder information needed by the gateway.

Version 1.4 of this method supports XProtect Management Server and recorders.

REST Request URL: `GET /api/access/onguard/openaccess/video_recorders?version=value`

get video_recorders

Name	Type	Required	Description
order_by	string	no	The fields to use when sorting the results.
filter	string	no	The filter used to retrieve instances. For example, `Lastname = "Smith" and Firstname = "Lisa"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. For more information refer to Searching for Objects on page 43.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
page_number	int32	no	The page number to be returned when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.

get video_recorders

Name	Type	Required	Description
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.

get video_recorders response

Name	Type	Description
computer_name	string	The computer name of the recorder.
count	int32	The number of recorders returned in the response.
database_id	int32	The database ID that identifies the server containing this recorder. Only returned for Enterprise systems.
gateway_http_port	int32	The HTTP port configured for the LenelS2 NVR web service's Gateway.
gateway_https_port	int32	The HTTPS port configured for the LenelS2 NVR web service's Gateway.
gateway_id	int32	The internal database ID of the gateway in the access panel table. Key field.
gateway_name	string	The display name of the gateway.
gateway_network_address	string	The fully qualified name of the gateway.
http_port	int32	The HTTP port configured for the LenelS2 NVR web service.
https_port	int32	The HTTPS port configured for the LenelS2 NVR web service.
id	int32	The internal database ID of the recorder in the access panel table. Key field.
is_daylight_saving	boolean	Whether or not this recorder observes Daylight Saving Time.
is_online	boolean	Whether or not the recorder is online.
mediasource_classid	string	Unique ID of the media source component used by the gateway to communicate with the respective recorder.
name	string	The display name of the recorder
page_number	int32	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.

get video_recorders response

Name	Type	Description
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
panel_type_id	int32	The internal database ID of the type of recorder in the panel type table.
panel_type_name	string	The name of the panel type.
port	int32	The recorder port configured on the Video Recorder form (Connection sub-tab) in System Administration. Used to establish a connection to the recorder.
primary_ip_address	int32	The primary IP address to use when connecting to a server with network access.
segment_id	int32	The segment to which this recorder belongs. Only returned for segmented systems.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total existing number of instances of the object being requested.
workstation	int32	The recorder workstation name.
world_timezone_id	int32	The time zone of the recorder (reference to Lnl_WorldTimezone.ID)
xprotect_ management_ server_id	int32	The internal database ID of the XProtect Management Server. This property is only available if the video recorder is a Milestone recorder.
xprotect_ management_ server_name	string	The display name of the XProtect Management Server. This property is only available if the video recorder is a Milestone recorder.
xprotect_ management_ server_network_ address	string	The host name, IP address, or Fully Qualified Domain Name (FQDN) of the XProtect Management Server. This property is only available if the video recorder is a Milestone recorder.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get video_recorder auth_data

This method retrieves the authentication token for a LenelS2 NVR. This token is used for authentication and authorization against LenelS2 NVR Services. This method replaces the GetAuthenticationData method of the Lnl_VideoRecorder type.

Notes: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

Version 1.2 and later of this method supports the **authentication_data** property.

In OnGuard 7.4, this method is supported for video recorders of type LenelS2 NVR only.

REST Request URL: `GET /api/access/onguard/openaccess/video_recorder/{id}/auth_data?version=value&authentication_data=value`

get video_recorder auth_data

Name	Type	Required	Description
id	int32	yes	The panel ID of the recorder for which the authentication data is being requested.
authentication_data	string	no	Authentication data string from a previous request. Determines if an audit record log is created for this video recorder. If present and can be decrypted, and is not expired, no audit log record is created. If not present (or NULL), or if cannot be decrypted, or if expired, audit log is created.
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get video_recorder auth_data response

Name	Type	Description
authentication_data	string	The authentication token for the specified recorder, or both the recorder and gateway.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get video_recorder credentials

Retrieves user credentials for the video recorder with the provided panel ID and realm.

REST Request URL: `GET /api/access/onguard/openaccess/video_recorder/{id}/credentials?realm_value=value&version=value`

get video_recorder credentials

Name	Type	Required	Description
id	int32	yes	The panel ID of the recorder for which credentials data is being requested.
realm_value	string	yes	String parameter provided by the client, to be used when calculating the hash.

get video_recorder credentials response

Name	Type	Description
video_recorder_hashed_data	string	The hashed string of the user's credentials.
video_recorder_user_name	string	The user's name, in plain text.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get map background

This method retrieves the background for a specific map ID. The response is an array of image data without encoding, in .png or .jpg format.

Note: The maximum image size is 55 MB.

REST Request URL: `GET /api/access/onguard/openaccess/map/{id}/background?version=value`

get map background

Name	Type	Required	Description
id	int32	yes	The map ID.

put access_level

Use this method to update access levels without using System Administration. Currently, this method only allows the assignment of readers to an access level, or the removal of readers from an access level.

REST Request URL: `PUT /api/access/onguard/openaccess/access_level/{id}`

put access_level

Name	Type	Required	Description

readers	JSON body	yes	JSON, in the format:
			`[`
			`{`
			`"reader_id": 5,`
			`"panel_id": 7,`
			`"timezone_id": 1`
			`},`
			`{`
			`"reader_id": 6,`
			`"panel_id": 2,`
			`"timezone_id": 1`
			`}`
			`]`

operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

put access_level response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

post send_incoming_events

Use this method to send a list of events to the OnGuard software.

IMPORTANT: Post send_incoming_event, with the intention of sending events to a Logical Source/Device/SubDevice, requires configuration of at least one access panel such as an LNL-2220, LNL-3300, LNL-4420, and so on. The configured access panel can be a "dummy" panel. For more information, refer to Chapter 8: Using OpenAccess to Send Alarms to OnGuard on page 337.

REST Request URL: `POST /api/access/onguard/openaccess/send_incoming_events?version=value`

post send_incoming_events

Name	Type	Required	Description
incoming_event_list	list	yes	A list of incoming events to send. Each event has the following attributes: • event_type: The event type. • event_id: The event ID. • source_id: The logical source ID. • device_id: The logical device ID. • subdevice_id: The logical sub-device ID. • event_text: The event text. • badge_id: The badge ID. • badge_id_str: A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **badge_id** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off. You cannot provide both **badge_id** and **badge_id_str** in the same call. • extended_id: The badge extended ID. • time: The event time. Optional. If not provided, the current time is used when sending the event.

post send_incoming_events response

Name	Type	Description

failure_list	map	A map of failed event attributes:
		• total_items: The total count of failed events.
		• item_list: The failed event list. Each failed event has the following attributes:
		  • event_type: The event type.
		  • event_id: The event ID.
		  • source_id: The logical source ID.
		  • device_id: The logical device ID.
		  • subdevice_id: The logical sub-device ID.
		  • event_text: The event text.
		  • badge_id: The badge ID.
		  • badge_id_str: A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **badge_id** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
		  • extended_id: The badge extended ID.
		  • time: The event time.

version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Users

get logged_in_user

Returns information pertaining to the authenticated user.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/logged_in_user?version=value`

get logged_in_user response

Name	Type	Description
user_id	string	The user's ID, as a string.
user_name	string	The user's user name, in plain text.
first_name	string	The user's first name.
last_name	string	The user's last name.

get logged_in_user response

Name	Type	Description
operation_ description	string	A description of the OpenAccess operations performed by the user. This parameter requires user transaction logging is enabled in the ACS.ini file. For more information, refer to User Transaction Logging on page 60.
operation_guid	string	The device GUIDs affected by the user's operations. This parameter requires user transaction logging is enabled in the ACS.ini file. For more information, refer to User Transaction Logging on page 60.
password_ expiration_time	datetime (string)	The date and time that the password will expire. This only exists if the user logged in with the password expiration policy enabled.
permission_map	map	A subset of user permissions configured in System Administration. For each entry in the map, the value is true if the user's assigned permission group has this permission, or false if the user's permission group does not have this permission. For more information, refer to "Administration: Users Folder: Permission Groups Tree: User Permissions" in the *System Administration User's Guide*.
ptz_priority	int32	The PTZ priority level of the user. Since only one person can control a PTZ camera at a time, a user with higher priority can take over PTZ control of a camera from someone who has lower priority. The SA user and SA delegate users have a PTZ priority of 1000. Other users are assigned values between 1 (low priority) and 255 (high priority). For more information, refer to "Monitor Permission Groups: Permissions Sub-tab Procedures" in the *System Administration User's Guide*.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get user managed_access_levels

Returns a list of access levels a user can manage, and indicates if the user has Area Access Manager view-only access.

Notes: If an **sa** user or SA delegate user calls get managed_access_levels after authenticating with OpenAccess as "sa", OpenAccess returns no results. The **sa** user and SA delegate users can manage all access levels in the system.

Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/user/{id}/managed_access_levels?version=value`

get user managed_access_levels

Name	Type	Required	Description
id	string	yes	ID of the user for whom you want the managed access levels, as a string.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get user managed_access_levels response

Name	Type	Description
access_level_list	list	The list of access levels a user can manage. Each item in the list contains the **id**, which is the ID of the access level associated with the user, and the **name**, which is the name of the access level. The access level filter and badge filter are combined, so that the access level search is applied only to those badges that match the badge filter. **Note:** With Version 1.3 and later of this endpoint, **segment_id** is also returned if the system is segmented.
total_items	int32	A count of the items in the access_level_list.
has_aam_view_ only_access	boolean	Describes if the user has view-only access to levels in Area Access Manager. If false, the user can control all assigned access levels in Area Access Manager. For a list of access levels the user can control, refer to get user managed_access_levels on page 134.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add user managed_access_levels

Adds to the existing list of access levels a user can manage.

Notes: If adding any of the requested access levels fail, an error code is provided and none of the requested access levels are added.

Access level management cannot be added to the SA user or SA delegate users.

REST Request URL: `POST /api/access/onguard/openaccess/user/{id}/managed_access_levels?version=value`

REST Request Body Contents:

```
{
    "access_level_list":
    [
        access_level_id,
        ...
    ]
}
```

add user managed_access_levels

Name	Type	Required	Description
id	string	yes	ID of the user to which access level management will be added, as a string.
access_level_list	list	yes	A list of access level IDs the user can manage.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

delete user managed_access_levels

Deletes specific access levels from the access levels a user can manage.

REST Request URL: `DELETE /api/access/onguard/openaccess/user/{id}/managed_access_levels?version=value`

REST Request Body Contents:

```
{
    "access_level_list":
    [
        access_level_id,
        ...
    ]
}
```

delete user managed_access_levels

Name	Type	Required	Description
id	string	yes	ID of user from which to remove access level management, as a string.

Name	Type	Required	Description
access_level_list	list	yes	A list of access level IDs the user cannot manage.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get user

Gets the OnGuard-specific properties for a user.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/user/{id}?version=value`

get user

Name	Type	Required	Description
id	string	yes	ID of the user for whom you want the monitoring zone ID and monitoring zone name, as a string.

get user response

Name	Type	Description
database_id	int32	The database identifier in an Enterprise system that identifies the server containing the user. For more information, refer to get enterprise settings on page 158.
monitoring_ zone_id	int32	The ID of the user’s monitoring zone. For more information, refer to Lnl_MonitoringZone on page 278.
monitoring_ zone_name	string	The name of the user’s monitoring zone. If the user is not associated with a monitoring zone, then this property is returned as empty.
has_aam_view_ only_access	boolean	Describes if the user has view-only access to levels in Area Access Manager. If false, the user can control all assigned access levels in Area Access Manager. For a list of access levels the user can control, refer to get user managed_access_levels on page 134.

get user response

Name	Type	Description
is_user_ account_locked	boolean	A flag to indicate if the user's account is locked because of too many incorrect password attempts.
last_ successful_ login_time	datetime	The date and time of the user's last successful login.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

modify user

Modifies the OnGuard-specific properties for a user.

REST Request URL: `PUT /api/access/onguard/openaccess/user/{id}?version=value&database_id=value`

modify user

Name	Type	Required	Description
database_id	int32	no	The database identifier in an Enterprise system that identifies the server containing the user. If changing this value with a **modify user** call, the existing value must be **-1** or the local DatabaseID, or an insufficient privileges error is returned. For more information, refer to get enterprise settings on page 158.
id	string	yes	ID of the user for whom you want to assign the monitoring zone ID, as a string.
monitoring_ zone_id	int32	no	ID of the monitoring zone you want to assign to the user.
has_aam_view_ only_access	boolean	no	Describes if the user has view-only access to levels in Area Access Manager. If false, the user can control all assigned access levels in Area Access Manager. For a list of access levels the user can control, refer to get user managed_access_levels on page 134. **Note:** You can only modify this value if the user has at least one access level to manage.
unlock_account	boolean	no	If true, unlock the account of the user with a locked account because of too many incorrect password attempts.

put user password

Update the current user's password.

REST Request URL: `PUT /api/access/onguard/openaccess/user_password?version=value`

put user password

Name	Type	Required	Description
user_name	string	yes	The user's name.
current_password	string	yes	The current password.
new_password	string	yes	The new password.
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, "Viewing a list of cameras." Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: "`00000000-0000-0000-0000-000000000000`") to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get managers_of_access_level

Gets a list of user IDs for users who can manage the access level.

Note: Users assigned "view-only" permission to an access level are not included in the list returned from this call.

REST Request URL: `GET /api/access/onguard/openaccess/managers_of_access_level?access_level_id=value&version=value`

get managers_of_access_level

Name	Type	Required	Description
access_level_id	int32	yes	ID of the access level for which to retrieve users who can manage that access level.
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, "Viewing a list of cameras." Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: "00000000-0000-0000-0000-000000000000") to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get managers_of_access_level response

Name	Type	Description
total_items	int32	A count of users who can manage the access level.
user_id_list	list	List of user IDs for users who can manage the access level.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get editable_segments

Gets a list of segments and segment groups for which the logged-in user has editable permission. For more information, refer to Lnl_Segment on page 311.

Notes: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

This call is not supported on non-segmented systems. Use the **get segmentation_settings** call to determine if your system supports segmentation (refer to get segmentation_settings on page 165).

REST Request URL: `GET /api/access/onguard/openaccess/editable_segments?version=value`

get editable_segments response

Name	Type	Description
total_items	int32	A count of segments and segment groups for which the logged-in user has editable permission.
segment_list	list	The list of segments assigned to a user. Each item in the list contains the **segment_id**, which is the ID of the segment assigned to the user, the **segment_name**, which is the name of the segment, and **type**, which is either **segment_unit**, or **segment_group**. For Enterprise systems, also returns **database_id** for each item in the **segment_list**, and type can also be **dynamic_segment**. For more information, refer to Lnl_Segment on page 311.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get restricted_segments

Returns a list of segments containing access levels that the logged-in user can assign to badges because of the **Can also be assigned to badges by users with access to** list of segments. These segments are considered “restricted” because the logged-in user might not be allowed to assign them to any object. For more information, refer to Lnl_Segment on page 311. Also refer to “Access Level Additional Segments Form” in the *System Administration User Guide*.

Notes: This call is not supported on non-segmented systems, or when the **Allow access levels to be configured as assignable by users in other segments** feature is disabled. Use the **get segmentation_settings** call to determine if your system supports segmentation. For

more information, refer to get segmentation_settings on page 165. Also refer to "Segment Options Form" in the *System Administration User Guide*.

REST Request URL: `GET /api/access/onguard/openaccess/restricted_segments?version=value`

get restricted_segments response

Name	Type	Description
total_items	int32	A count of segments for which the logged-in user has editable permission.
segment_list	list	The list of segments assigned to a user. Each item in the list contains the **segment_id**, which is the ID of the segment assigned to the user, the **segment_name**, which is the name of the segment, and **type**, which is always the **segment_unit**. For Enterprise systems, also returns **database_id** for each item in the **segment_list**, and type can also be **dynamic_segment**. For more information, refer to Lnl_Segment on page 311.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get user_segments

Returns a list of segments assigned to a user.

Notes: This call is not supported on non-segmented systems. Use the **get segmentation_settings** call to determine if your system supports segmentation. For more information, refer to get segmentation_settings on page 165.

Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/user/{id}/segments?version=value`

get user_segments

Name	Type	Required	Description
id	string	yes	ID of the user for whom you want to retrieve segments, as a string.

get user_segments response

Name	Type	Description
segment_list	list	The list of segments assigned to a user. Each item in the list contains the **segment_id**, which is the ID of the segment assigned to the user, the **segment_name**, which is the name of the segment, and **type**, which is either **segment_unit**, or **segment_group**. For Enterprise systems, also returns **database_id** for each item in the **segment_list**, and type can also be **dynamic_segment**. For more information, refer to Lnl_Segment on page 311.
total_items	int32	A count of the segments in the segment_list.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

add user_segments

Adds to the existing list of segments assigned to a user. Use the **get editable_segments** call to determine which segments can be assigned to a user. For more information, refer to get editable_segments on page 140.

Note: This call is not supported on non-segmented systems. Use the **get segmentation_settings** call to determine if your system supports segmentation. For more information, refer to get segmentation_settings on page 165.

REST Request URL: `POST /api/access/onguard/openaccess/user/{id}/segments?version=value`

REST Request Body Contents:

```
{
    "segment_list":
    [
        segment_id,
        ...
    ]
}
```

add user_segments

Name	Type	Required	Description
id	string	yes	ID of the user to which segment assignment will be added, as a string.
segment_list	list	yes	A list of segment IDs that indicate which segments to assign to the user. For more information, refer to Lnl_Segment on page 311.

delete user_segments

Deletes specific segments from the segments assigned to a user. Use the **get editable_segments** call to determine which segments can be deleted from a user. For more information, refer to get editable_segments on page 140.

Note: This call is not supported on non-segmented systems. Use the **get segmentation_settings** call to determine if your system supports segmentation. For more information, refer to get segmentation_settings on page 165.

REST Request URL: `DELETE /api/access/onguard/openaccess/user/{id}/segments?version=value`

REST Request Body Contents:

```
{
    "segment_list":
    [
        segment_id,
        ...
    ]
}
```

delete user_segments

Name	Type	Required	Description
id	string	yes	ID of user from which to remove segment assignment, as a string.
segment_list	list	yes	A list of segment IDs that indicate which segments to remove from the user. For more information, refer to Lnl_Segment on page 311.

get user preferences

Gets the user preferences of the logged in user.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/user_preferences?version=value&setting_type=value&preference_id=value&is_global=value`

get user preferences

Name	Type	Required	Description
setting_type	string	yes	The setting type refers to the category of settings to which the client wants to refer. For example, `setting_type="UI"`.
preference_id	int32	no	The unique ID of the preference.

Name	Type	Required	Description
is_global	boolean	no	Optional parameter. Get call returns all the preferences of the logged-in user, as well as global preferences. If TRUE, only the global preferences are returned. If FALSE, returns the preferences of that logged-in user only.
client_name	string	yes	The name of the client application making use of the user preferences (for example, Credentials, CSS, Access Manager, Monitor).
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get user preferences response

Name	Type	Description
preference_list	string	Refers to the list of preferences, in JSON format.
total_list	int32	The total number of user preferences retrieved.
client_name	string	The name of the client application making use of the user preferences (for example, Credentials, CSS, Access Manager, Monitor).
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put user preferences

Update the existing user preferences of the logged in user.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `PUT /api/access/onguard/openaccess/user_preferences?version=value`

put user preferences

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
preference_id	int32	yes	The unique identifier of the user preference.
preference_settings	string	no	The preference settings refers to the data the user wants to save, in json format. For example: `preference_settings: {"Address":{"Operator":"LIKE", "value":"NYC"}}` **Note:** The maximum length of this parameter is limited to 900 kB.
setting_type	string	yes	The setting type refers to the category of settings to which the client wants to refer. For example, `setting_type="UI"`.

put user preferences response

Name	Type	Description
preference_id	int32	The unique identifier of the user preference.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

post user preferences

Save the user preferences of the logged in user.

REST Request URL: `POST /api/access/onguard/openaccess/user_preferences?version=value`

post user preferences

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
preference_set-tings	string	no	The preference settings refers to the data the user wants to save, in json format. For example: `preference_settings: {"Address":{"Operator":"LIKE", "value":"NYC"}}` **Note:** The maximum length of this parameter is limited to 900 kB.
setting_type	string	yes	The setting type refers to the category of settings to which the client wants to refer. For example, `setting_type="UI"`.
is_global	boolean	no	If TRUE, the preference is visible to other users. If FALSE, the preference is visible only to the logged-in user.
client_name	string	yes	The name of the client application making use of the user preferences (for example, Credentials, CSS, Access Manager, Moni-tor).

post user preferences response

Name	Type	Description
preference_id	int32	The unique identifier of the user preference.
preference_ settings	json	The data the user wants to save in json format. For example: `preference_settings : {"Address":{"Opera-tor":"LIKE", "value":"NYC"}}`
setting_type	string	The category of settings to which the client refers. For example: `setting_type="UI"`
is_global	boolean	If "is global" is TRUE, the preference is visible to other users. If 'is_global" is FALSE, the preference is visible to only the logged in user.

post user preferences response

Name	Type	Description
user_id	int32	The owner of the preference. In case of global preference, the value of the user_id is `id0`.
client_name	string	The name of the client application making use of the user preferences (for example, Credentials, CSS, Access Manager, Monitor).
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete user preferences

Delete the existing user preferences of the logged in user, and current application type.

REST Request URL: `DELETE /api/access/onguard/openaccess/user_preferences?version=value`

delete user preferences

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
preference_id	int32	yes	The unique identifier of the user preferences to be removed.

delete user preferences response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Cardholders

get cardholder_from_directory

This is an authenticated method that returns the internal ID, equivalent to Lnl_cardholder.ID, of a cardholder in the system who has a linked directory account with the directory credentials that are passed in as parameters. For more information, refer to Lnl_Cardholder on page 255.

get cardholder_from_directory

Name	Type	Required	Description
user_name	string	yes	The user's user name, in plain text.
password	string	yes	The user's password, in plain text.
directory_id	string	yes	The cardholder's directory ID, as a string. To get a list of available directory IDs, use the **get directories** call. For more information, refer to get directories on page 68.

get cardholder_from_directory response

Name	Type	Description
cardholder_id	int32	The ID of the cardholder.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get directory_accounts

Gets directory accounts matching the provided filter.

Notes: Depending on the Active Directory Server configuration, number of users in the directory, and uniqueness of the search criteria, this method might time out. Consider using the queue parameter, which allows for an asynchronous response. For more information, refer to Task queuing: dealing with long running requests on page 59, and also refer to get queue on page 64.

Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/directory_accounts`

get directory_accounts

Name	Type	Required	Description
directory_id	string	yes	Directory ID of the directory containing the active directory accounts you want to find, as a string. To get a list of available directory IDs, use the **get directories** call. For more information, refer to get directories on page 68.
filter	string	yes	Filter, in the format <adattr> <condition> '<value>'. For example, `displayname has 'smith'` • Support Conditions: `eq, has`. One specific case is <adattr> <eq> '', which means AD attribute's value is empty. For example, `displayname eq ''` • Support negative conditions: not(<adattr.> <has> '<value>') means AD attribute's value does not contain the input value. For example, not(samaccountname has 'smith') not(<adattr.> <eq> '') means AD attribute's value is not empty.

get directory_accounts_matching_cardholders

Gets directory accounts matching the given cardholders, based on the property pairs specified by the filter.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/directory_accounts_matching_cardholders`

get directory_accounts_matching_cardholders

Name	Type	Required	Description
directory_id	string	yes	Directory ID of the directory containing the active directory accounts you want to find, as a string. To get a list of available directory IDs, use the **get directories** call. For more information, refer to get directories on page 68.
cardholder_ids	int32 array	yes	List of cardholder IDs in the format [1,2,3,...].
filter	string	yes	OData-formatted filter. Compares a directory account's attribute value with cardholder record attribute value.

Additional Filter Details

Filter format: `<adattr> <condition> '<cardholderattr>'`. For example, `displayname has 'firstname'`

Filter supports these comparison types: `eq, has`

Filter supports the negative condition: Therefore, `not(<adattr.> <has> '<cardholderattr>')` means the Active Directory attribute's value does not contain the Cardholder attribute's value. For example, `not(displayname has 'lastname')`.

get directory_accounts_matching_cardholders response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

The returned parameters are a list of matching cardholders or non-matching cardholders. For example:

```
name : type : required : description

version : string : yes : used by openaccess to maintain back... etc.

successful_list : object : contains a list of successfully matched cardholders' details

successful_list.total_items : int32 : count of successfully matched cardholders

successful_list.item_list: object array : list of successfully matched cardholders' details

successful_list.item_list.cardholder_id: int32 : cardholder id

successful_list.item_list.directory_account : object : contains details about the cardholder

successful_list.item_list.directory_account.SID : string : SID of the matched directory user

successful_list.item_list.directory_account.email : string : email of the matched directory user

successful_list.item_list.directory_account.user_name : string : username of the matched directory user

failure_list : contains a list of cardholders that could not be matched to directory accounts

failure_list.total_items : int32 : count of failed matches

failure_list.item_list : object : list of failed matched cardholders

failure_list.item_list.cardholder_id : int32 : id of an unmatched cardholder

failure_list.item_list.error_message : string : reason why the match failed for this cardholder
```

put update_cardholder_with_directory_account_property

Updates the given cardholder with the given directory account property.

REST Request URL: `PUT /api/access/onguard/openaccess/update_cardholder_with_directory_account_property`

put update_cardholder_with_directory_account_property

Name	Type	Required	Description
cardholder_id	integer	yes	The ID of the cardholder to update with a directory account property.

parameter_name	JSON body	yes	JSON, in the format:
			`{`
			`"directory_account_property": "string",`
			`"cardholder_property": "string",`
			`"can_overwrite": true`
			`}`

put update_cardholder_with_directory_account_property response

Name	Type	Description
updated	boolean	Indicates if the cardholder has been updated with the directory account property.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Console

post console cards

Adds a console card to all layouts, or modifies a console card in the system console layout.

Note: The maximum length of the layout description is limited to 900 kB.

REST Request URL: `POST /api/access/onguard/openaccess/console/cards?version=value`

post console cards

Name	Type	Required	Description
id	string	no	The ID of the console card.
group_id	string	no	The group ID to which the console card belongs.

post console cards

Name	Type	Required	Description
license	string	yes	The feature license ID.
display_name	string	yes	The console card display name.
color	string	yes	The color, in HEX.
icon	string	yes	The icon content, in base64. Should start with 'data:*/*;base64,'. **Note:** The maximum size of the image is limited to 50 kB.
application_type	string	yes	Options are 'web' or 'native'.
url	string	yes	The card URL.
extended_ properties	string	no	Currently empty, but in the future could contain a JSON-formatted text string to be used by the LenelS2 Console web application to define and store new properties to associate with a console card.
type	string	yes	The type of card. Options are 'system_default' or 'user'.

post console cards response

Name	Type	Description
Session-Token	string	The authentication token for the current user session.
Application-Id	string	A unique Application-Id, provided by LenelS2 OnGuard Technical Support.
id	string	The ID of the console card.
group_id	string	The group ID to which the console card belongs.
license	string	The feature license ID.
display_name	string	The console card display name.
color	string	The color, in HEX.
icon	string	The icon content, in base64. Should start with 'data:*/*;base64,'.
application_type	string	Options are 'web' or 'native'.
url	string	The card URL.
extended_ properties	string	Currently empty, but in the future could contain a JSON-formatted text string to be used by the LenelS2 Console web application to define and store new properties to associate with a console card.
type	string	The type of card. Options are 'system_default' or 'user'.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete console cards with id

Deletes the specified console card from all layouts.

Note: Version 1.1 and later of this method supports string encoding, which is helpful in preventing malicious code from being run by the client browser. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `DELETE /api/access/onguard/openaccess/console/cards?card_id=value&version=value`

delete console cards with id

Name	Type	Required	Description
card_id	string	yes	The ID of the console card.

delete console cards with id response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get console layouts

Returns the specific system console layout.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/console/layouts?layout_id=value&version=value`

get console layouts

Name	Type	Required	Description
layout_id	string	yes	The ID of the console layout.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get console layouts response

Name	Type	Description
id	string	The ID of the console layout.
display_name	string	The console layout display name.
groups	string	List of console card groups, in JSON format.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put console layouts

Modify the existing system console layout, or add the console layout if it does not exist already.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `PUT /api/access/onguard/openaccess/console/layouts?version=value`

put console layouts

Name	Type	Required	Description
id	string	no	The ID of the console layout. Add a new console layout if it is not provided.
display_name	string	yes	The console layout display name.
groups	string	yes	List of console card groups, in JSON format. **Note:** The maximum length of this parameter is limited to 900 kB.

put console layouts response

Name	Type	Description
console_layout_id	string	The unique ID of the console layout.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Settings

get authorization warning settings

Returns the settings for an authorization warning, as configured in System Administration.

Notes: You do not need to be logged in to make this call. A session-token and application-id are not required.

Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

If **Logon authorization warning** in System Administration is set to **None**, then the response to **get authorization_warning display_authorization_warning** is set to **false** and **authorization_warning_options** is not available.

Authorization_warning_options is a map which contains the values described in the Response table below. One property in the map is **font_properties**, which is a map of properties specific to the display font.

Some of the font properties are not directly selectable in the font dialog when setting up the font for the authorization warning in System Administration. For example, **escapement** cannot be set directly. Its value is based on other factors of the font selection. **height** is related to the font size selected, but does not map to it exactly; it often comes back negative. **weight** changes based on whether **bold** is selected or not. **face_name** is the name of the font selected. These properties come directly from the MFC LOGFONT structure. The purpose is to give a web client application all of the font information, and then let the client figure out how to convert this information to the appropriate HTML for the client to show.

REST Request URL: `GET /api/access/onguard/openaccess/settings/authorization_warning?version=value`

get authorization warning settings response

Name	Type	Description
display_ authorization_ warning	boolean	Indicates if the client should display the authorization warning.
authorization_ warning_options	map	Will not be present if display_authorization_warning is false. Contains information about how to display the warning.
authorization_ warning_text	string	Member of authorization_warning_options. The authorization warning text to display. Can include HTML hyperlinks.
yes_button_text	string	Member of authorization_warning_options. The text to display on the Yes button.
no_button_text	string	Member of authorization_warning_options. The text to display on the No button.
operation_ description	string	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
yes_is_default_button	boolean	Member of authorization_warning_options. If true, the Yes button is the default button in the authorization warning dialog.

get authorization warning settings response

Name	Type	Description
font_properties	map	Member of authorization_warning_options. Describes the display font for the authorization warning. • height (int32) • width (int32) • escapement (int32) • orientation (int32) • weight (int32) • italic (boolean) • underline (boolean) • strikeout (boolean) • character_set (string) • out_precision (string) • clip_precision (string) • quality (string) • pitch (string) • family (string) • face_name (string)
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get cardholder settings

Returns cardholder- and badge-related settings for the system as configured in System Administration.

REST Request URL: `GET /api/access/onguard/openaccess/settings/cardholder?segment_id=value&version=value`

get cardholder settings

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get cardholder settings

Name	Type	Required	Description
segment_id	int32	yes	Identifies the segment from which to retrieve cardholder options, and is required only if the system is segmented. For more information, refer to get segmentation_settings on page 165.

get cardholder settings response

Name	Type	Description
activate_ deactivate_ dates_use_time	boolean	Indicates whether or not both date and time are specified for badge activation/deactivation.
badge_pin_ properties	map	• **can_edit_pin_code (boolean):** If true, a user with the appropriate permissions can change PIN values. • **copy_pin_code (boolean):** If true, the Copy PIN check box on the Access Level and PIN Assignment dialog is selected by default. If false, the Copy PIN check box is not selected by default. For more information, refer to *Add or Replace a Badge Record* in the System Administration User Guide. • **digits (int32):** Indicates the number of digits the PIN contains. • **enforce_unique_pin_code (boolean):** If true, indicates that the cardholder badge record must have a unique PIN code. If false, duplicate PIN codes are allowed. • **generate_pin_code (boolean):** If true, indicates whether a PIN is randomly generated when a badge is created. If false, a PIN must be manually entered.
create_ photo_ thumbnails	boolean	Indicates whether or not thumbnail versions for all existing cardholder photos are saved in the database.
max_ accesslevels_ per_badge_ standard	int32	Indicates the maximum number of standard access levels that can be assigned to a badge at one time. For LenelS2 access panels, the maximum number is 128. Dependent on the segment_id property, if segmentation is enabled.
max_ accesslevels_ per_badge_ temporary	int32	Indicates the maximum number of temporary access levels that can be assigned to a badge at one time. For LenelS2 access panels, the maximum number is 128. Dependent on the segment_id property, if segmentation is enabled.
max_ accesslevels_ per_badge_total	int32	Indicates the maximum number of access levels that can be assigned to a badge at one time. This includes both standard and temporary access levels. For LenelS2 access panels, the maximum number is 128. Dependent on the segment_id property, if segmentation is enabled.
max_ active_badges	int32	Indicates the maximum number of active badges that are allowed for each cardholder.

get cardholder settings response

Name	Type	Description
max_badge_ id_length	int32	Indicates the maximum number of digits in a badge number. For LenelS2 access panels, the maximum length is 18 digits. Dependent on the segment_id property, if segmentation is enabled.
max_extended_ id_length	int32	Indicates the maximum extended ID length if extended identifiers are used (64 bits long). For LenelS2 access panels, the maximum length is 32 bytes. Dependent on the segment_id property, if segmentation is enabled.
temporary_ accesslevel_ granularity	int32	Indicates how frequently the Linkage Server examines and updates temporary access levels for date and time badge activation and deactivation purposes.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get enterprise settings

Returns enterprise-related settings for the system as configured in System Administration, if Enterprise support is enabled.

REST Request URL: `GET /api/access/onguard/openaccess/settings/enterprise?version=value`

get enterprise settings

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get enterprise settings response

Name	Type	Description
default_ cardholder_ replication	int32	The value in this property indicates where the cardholder record gets replicated. This property is not available on a Global Server. Returns a value that matches one of the items in the server_list property as the database_id.
default_user_ replication	int32	The value in this property indicates where a user record gets replicated. Returns a value that matches one of the items in the server_list property as the database_id.

get enterprise settings response

Name	Type	Description
default_visitor_ replication	int32	The value in this property indicates where the visitor record gets replicated. This property is not available on a Global Server. Returns a value that matches one of the items in the server_list property as the database_id.
is_enterprise_ system	boolean	Identifies whether or not this is an OnGuard Enterprise system.
is_global_server	boolean	Identifies whether or not this machine is the Global Server in an OnGuard Enterprise system.
local_database_ id	int32	Identifies the id of this Enterprise Global Server.
server_list	list	All Enterprise Global Servers of the Enterprise system. A list that will return database_id, display_name, and server_type of each server.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get password policy setting limits

Returns the minimum and maximum limits for the password policy.

REST Request URL: `GET /api/access/onguard/openaccess/settings/password_policy_limits?version=value`

get password policy limits

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get password policy limits response

Name	Type	Description
login_attempt_ threshold_min	int32	The minimum allowed number of invalid login attempts that will lock an internal account.

get password policy limits response

Name	Type	Description
login_attempt_ threshold_max	int32	The maximum allowed number of invalid login attempts that will lock an internal account.
login_attempt_ reset_interval_ in_minutes_min	int32	The minimum allowed number of minutes to wait before resetting the record of invalid logins.
login_attempt_ reset_interval_ in_minutes_max	int32	The maximum allowed number of minutes to wait before resetting the record of invalid logins.
lockout_interval_ in_minutes_min	int32	The minimum allowed number of minutes to lock an internal account after exceeding the invalid login attempt threshold.
lockout_interval_ in_minutes_max	int32	The maximum allowed number of minutes to lock an internal account after exceeding the invalid login attempt threshold.
expiration_days_ min	int32	The minimum allowed number of days the password will be expired.
expiration_days_ max	int32	The maximum allowed number of days the password will be expired.
expiration_ reminder_days_ min	int32	The minimum allowed number of days to start reminding the user, with each login, that the password is almost expired.
expiration_ reminder_days_ max	int32	The maximum allowed number of days to start reminding the user, with each login, that the password is almost expired.
minimum_ length_min	int32	The minimum allowed password length.
minimum_ length_max	int32	The maximum allowed password length.
history_ password_count_ min	int32	The minimum allowed number of previous passwords that will be prohibited when resetting the password.
history_ password_count_ max	int32	The maximum allowed number of previous passwords that will be prohibited when resetting the password.
minimum_ password_age_ min	int32	The minimum allowed number of days users must keep a password before they can change it.
minimum_ password_age_ max	int32	The maximum allowed number of days users must keep a password before they can change it.
inactivity_ timeout_in_ minutes_min	int32	The minimum number of minutes before authenticated token inactivity timeout.

get password policy limits response

Name	Type	Description
inactivity_ timeout_in_ minutes_max	int32	The maximum number of minutes before authenticated token inactivity timeout.
expiration_ first_reminder_ days_min	int32	The minimum number of days to first remind the user that the password is almost expired.
expiration_ first_reminder_ days_max	int32	The maximum number of days to first remind the user that the password is almost expired.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get password policy settings

Returns the password policy settings for the system.

REST Request URL: `GET /api/access/onguard/openaccess/settings/password_policy?version=value`

get password policy settings

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get password policy settings response

Name	Type	Description
is_lockout_ policy_enabled	boolean	A flag indicating whether the lockout policy is enabled.
login_attempt_ threshold	int32	The number of invalid login attempts that will lock an internal account.
login_attempt_ reset_interval_ in_minutes	int32	The number of minutes to wait before resetting the record of invalid logins.

get password policy settings response

Name	Type	Description
lockout_interval_in_minutes	int32	The number of minutes to lock an internal account after exceeding the invalid login attempt threshold.
disable_lockout_for_sa	boolean	Supports disabling the lockout policy for the SA user and SA delegate users.
is_expiration_policy_enabled	boolean	A flag indicating whether the expiration policy is enabled.
expiration_days	int32	The number of days the password will be expired.
is_expiration_reminders_enabled	boolean	A flag indicating whether to remind the user if the password is almost expired.
expiration_first_reminder_days	int32	The first day to remind the user that the password is almost expired.
expiration_reminder_days	int32	The day to start reminding the user with each login that the password is almost expired.
is_minimum_length_required	boolean	A flag indicating whether a minimum password length is required.
minimum_length	int32	The minimum password length.
is_numeric_characters_required	boolean	A flag indicating whether the password must contain a numeric character.
is_special_characters_required	boolean	A flag indicating whether the password must contain a non-alphanumeric character.
is_upper_and_lower_case_required	boolean	A flag indicating whether the password must contain an uppercase alphabetic and a lowercase alphabetic character.
is_history_policy_enabled	boolean	A flag indicating whether the password history policy is enabled.
history_password_count	int32	The number of previous passwords that will be prohibited when resetting the password.
minimum_password_age	int32	Determines how long users must keep a password before they can change it.
is_prohibited_password_policy_enabled	boolean	A flag indicating whether the prohibited password policy is enabled.
is_inactivity_timeout_policy_enabled	boolean	A flag indicating whether the inactivity timeout policy is enabled.
inactivity_timeout_in_minutes	int32	The authenticated token inactivity timeout, in minutes.

get password policy settings response

Name	Type	Description
can_be_same_as_user_name	boolean	A flag indicating whether the password can be the same as the user name.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put password policy settings

Updates the password policy settings for the system.

REST Request URL: `PUT /api/access/onguard/openaccess/settings/password_policy?version=value`

put password policy settings

Name	Type	Required	Description
can_be_same_as_user_name	boolean	no	A flag indicating whether the password can be the same as the user name. Default = FALSE
disable_lockout_for_sa	boolean	no	Supports disabling the lockout policy for the SA user and SA delegate users. Default = FALSE
expiration_days	int32	no	The number of days the password will be expired. Default = 90 Minimum = 0 Maximum = 730
expiration_first_reminder_days	int32	no	The first day to remind the user that the password is almost expired. Default = 15 Minimum = expiration_reminder_days Maximum = expiration_days
expiration_reminder_days	int32	no	The day to start reminding the user with each login that the password is almost expired. Default = 7 Minimum = 0 Maximum = expiration_days
history_password_count	int32	no	The number of previous passwords that will be prohibited when resetting the password. Default = 3 Minimum = 0 Maximum = 24

put password policy settings

Name	Type	Required	Description
inactivity_timeout_in_minutes	int32	no	The authenticated token inactivity timeout, in minutes. Default = 15 Minimum = 1 Maximum = authenticated_token_timeout configured in openaccess.ini
is_expiration_policy_enabled	boolean	no	A flag indicating whether the expiration policy is enabled. Default = TRUE
is_expiration_reminders_enabled	boolean	no	A flag indicating whether to remind the user if the password is almost expired. Default = FALSE
is_history_policy_enabled	boolean	no	A flag indicating whether the password history policy is enabled. Default = TRUE
is_inactivity_timeout_policy_enabled	boolean	no	A flag indicating whether the inactivity timeout policy is enabled. Default = TRUE
is_lockout_policy_enabled	boolean	no	A flag indicating whether the lockout policy is enabled. Default = TRUE
is_minimum_length_required	boolean	no	A flag indicating whether a minimum password length is required. Default = TRUE
is_numeric_characters_required	boolean	no	A flag indicating whether the password must contain a numeric character. Default = TRUE
is_prohibited_password_policy_enabled	boolean	no	A flag indicating whether the prohibited password policy is enabled. Default = TRUE
is_special_characters_required	boolean	no	A flag indicating whether the password must contain a non-alphanumeric character. Default = TRUE
is_upper_and_lower_case_required	boolean	no	A flag indicating whether the password must contain an uppercase alphabetic and a lowercase alphabetic character. Default = TRUE
lockout_interval_in_minutes	int32	no	The number of minutes to lock an internal account after exceeding the invalid login attempt threshold. Default = 5 Minimum = 1 Maximum = 99999

put password policy settings

Name	Type	Required	Description
login_attempt_threshold	int32	no	The number of invalid login attempts that will lock an internal account. Default = 3 Minimum = 1 Maximum = 999
login_attempt_reset_ interval_in_minutes	int32	no	The number of minutes to wait before resetting the record of invalid logins. Default = 60 Minimum = 1 Maximum = 99999
minimum_length	int32	no	The minimum password length. Default = 8 Minimum = 1 Maximum = 127
minimum_password_age	int32	no	Determines how many days a users must keep a password before they can change it. Default = 0 Minimum = 0 Maximum = 7
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

put password policy settings response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get segmentation_settings

Returns the segmentation-related settings of the system as configured in System Administration. The information returned in the response of this call identifies which SEGMENTID properties or classes are shown in OpenAccess. For more information, refer to Chapter 7: Data and Association Class Reference on page 223.

Note: For more information about segmentation settings, refer to “Segment Options Form” in the *System Administration User Guide*.

REST Request URL: `GET /api/access/onguard/openaccess/settings/segmentation?version=value`

get segmentation_settings

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Additional operation_guid Details on page 62.

get segmentation_settings response

Name	Type	Description
allow_access_ levels_to_be_ configured_as_ assignable_by_ other_segments	boolean	Identifies if users in other segments can configure this segment’s access levels.
allow_segment_ to_belong_to_ multiple_groups	boolean	Identifies if this segment can belong to more than one segment group.
extended_options	int32	Only in non-segmented systems. A number formed by combining bitwise flags. Bit designations: • 0x01 - Extended Options Access Levels • 0x02 - Extended Options - extended Cardholder storage • 0x04 - Extended Options - override Locked Reader Mode **Note:** This property is only returned when **get segmentation_settings** is called with version 1.2 or later.
segment_ badge_types	boolean	Identifies if badge type segmentation is enabled.
segment_card_ formats	boolean	Identifies if card format segmentation is enabled.
segment_ cardholders	boolean	Identifies if cardholders are segmented.
segment_non_ system_list_ builder_lists	boolean	Identifies if non-system List Builder entries are segmented.

get segmentation_settings response

Name	Type	Description
segment_visitors	boolean	Identifies if visitors are segmented.
segmentation_ enabled	boolean	Identifies if segmentation is enabled.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

For more information, refer to “Segment Options Form” in the *System Administration User Guide*.

get visit settings

Gets the visit settings of the system.

Note: Version 1.1 of this method supports string encoding. For more information, refer to Preventing Malicious Code in OpenAccess Responses on page 55.

REST Request URL: `GET /api/access/onguard/openaccess/settings/visit?version=value`

Note: All **get visit settings response** parameters except **default_visitor_badge_type_id**, **default_visitor_badge_type_name**, and **version** require version 1.2 or later of the **get visit settings** method.

get visit settings

Name	Type	Required	Description
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get visit settings response

Name	Type	Description
allow_access_ badge	boolean	Enables assigning an access control badge on visit sign-in. If enabled, the user can type in the badge ID of an existing visitor badge when signing in for a visit. The badge must already exist in the system, and its badge type must be of the class "Visitor".

get visit settings response

Name	Type	Description
allow_ disposable_ badge	boolean	If enabled, the user can print a disposable badge by selecting a disposable badge type that is assigned and printed when adding a visit.
allow_email	boolean	If enabled, e-mail notifications can be sent for visits. For this feature to work, an SMTP Server must be configured.
default_time_in	string	The time that visits begin by default. This time becomes the default time displayed in the **Time in** field on the Visit form. Format: "HH:MM:SS" (for example, `08:00:00`). The **default_time_in** value must be lower than the **default_time_out** value.
default_time_out	string	The time that visits end by default. This time becomes the default time displayed in the **Time out** field on the Visit form. Format: "HH:MM:SS" (for example, `17:00:00`). The **default_time_out** value must be higher than the **default_time_in** value.
default_visitor_ badge_type_id	string	The unique identifier of the default visitor badge type.
default_visitor_ badge_type_ name	string	The name of the default visitor badge type.
include_default_ recipients	boolean	If enabled, the **Default Recipients** checkbox on the E-mail sub-form in the Visits folder is selected by default when a new visit is added.
include_host_ email	boolean	If enabled, the **Cardholder for this visit** checkbox on the E-mail sub-form in the Visits folder is selected by default when a new visit is added. If a visit is added based on a currently-selected record, the setting for that record is used instead of the default.
include_ visitor_email	boolean	If selected, the **Visitor for this visit** checkbox on the E-mail sub-form in the Visits folder is selected by default when a new visit is added. If a visit is added based on a currently-selected record, the setting for that record is used instead of the default.
prompt_to_ update	boolean	If the **update_badges** and **prompt_to_update** properties are enabled, active badges are synchronized with active visits, and the logged-in user is prompted before the synchronization occurs. If **update_badges** is enabled but **prompt_to_update** is not enabled, all active badges and active visits are synchronized, but the logged-in user is not prompted when they are synchronized.
refresh_rate	int32	This refresh rate determines the default value for the **Refresh rate (in minutes)** field on the Status search sub-form in the Visits folder. The refresh rate is how often the database is queried to see if it has changed. Range is 0 to 9999 minutes.
sign_in_now_ by_default	boolean	If enabled, the **Sign in now** checkbox is selected by default when a visit is added, the visit will be automatically signed in, and the **Time in** field for the visit is updated with the current time.
signout_badge_ status	int32	Allows the user to choose the default badge status to which active badges are changed when a Visit is signed out.

get visit settings response

Name	Type	Description
update_badges	boolean	If enabled, synchronizes the badge activation and deactivation date with the visit time in and visit time out. A visitor's badge is activated at the beginning of the day for which the visit is scheduled, and deactivated at the end of the day for which the visit is scheduled.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put visit settings

Modifies the visit settings of the system.

REST Request URL: `PUT /api/access/onguard/openaccess/settings/visit?version=value`

put visit settings

Name	Type	Required	Description
allow_access_badge	boolean	no	Enables assigning an access control badge on visit sign-in. If enabled, the user can type in the badge ID of an existing visitor badge when signing in a visit. The badge must already exist in the system, and its badge type must be of the class "Visitor".
allow_disposable_badge	boolean	no	If enabled, the user can print a disposable badge by selecting a disposable badge type to be assigned and printed when adding a visit.
allow_email	boolean	no	If enabled, e-mail notifications can be sent for visits. For this feature to work, you must configure an SMTP Server.
default_time_in	string	no	Select the time that visits begin by default. This time becomes the default time shown in the **Time in** field on the Visit form. Format: "`HH:MM:SS`", for example "`08:00:00`". **default_time_in** value must be lower than **default_time_out**.
default_time_out	string	no	Select the time that visits end by default. This time becomes the default time that is displayed in the **Time out** field on the Visit form. Format: "`HH:MM:SS`", for example "**17:00:00**". **default_time_out** value must be higher than **default_time_in**.

put visit settings

Name	Type	Required	Description
default_visitor_ badge_type_id	int32	no	The unique identifier of the default badge type ID used for visit. Will return **0** if badge type ID was not assigned, or **-1** if user does not have enough privileges to see it.
include_default_ recipients	boolean	no	If enabled, the **Default Recipients** checkbox on the E-mail form in the Visits folder is selected by default when a new visit is added.
include_host_email	boolean	no	If enabled, the **Cardholder for this visit** checkbox on the E-mail form in the Visits folder is selected by default when a new visit is added. If a visit is added based on a currently-selected record, the setting for that record is used instead of the default.
include_visitor_ email	boolean	no	If selected, the **Visitor for this visit** checkbox on the E-mail form in the Visits folder is selected by default when a new visit is added. If a visit is added based on a currently-selected record, the setting for that record is used instead of the default.
operation_ description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.
prompt_to_update	boolean	no	If **update_badges** and **prompt_to_update** are enabled, active badges are synchronized with active visits, and the logged-in user is prompted before the synchronization occurs. If **update_badges** is enabled but **prompt_to_update** is not enabled, all active badges and active visits are synchronized, but the logged-in user is not prompted when they are synchronized.
refresh_rate	int32	no	Determines the default value for the **Refresh rate (in minutes)** field on the Status search sub-form in the Visits form. The refresh rate is how often the database is queried to see if it has changed. From 0 - 9999 minutes.

put visit settings

Name	Type	Required	Description
sign_in_now_by_default	boolean	no	If enabled, the **Sign in now** check box is selected by default when a visit is added, the visit is signed in automatically, and the **Time in** field for the visit is updated with the current time.
signout_badge_status	int32	no	Allows the user to choose the default badge status to which active badges are changed when the visit is signed out.
update_badges	boolean	no	If enabled, synchronizes the badge activation and deactivation date with the visit time in and visit time out. A visitor's badge is activated at the beginning of the day for which the visit is scheduled, and deactivated at the end of the day for which the visit is scheduled.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put visit settings response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get video settings

Retrieves video settings for the Video Tile.

REST Request URL: `GET /api/access/onguard/openaccess/settings/video?version=value`

get video settings

Name	Type	Required	Description
operation_description	string	no	A user-friendly free-form description for the operation being performed. For example, “Viewing a list of cameras.” Use this description to enhance the readability of the User Transaction Log reports.

get video settings

Name	Type	Required	Description
operation_guid	GUID (string)	no	An arbitrary GUID (formatted as 32-digits, separated by hyphens: “00000000-0000-0000-0000-000000000000”) to indicate a correlation among multiple API calls that will achieve a common purpose. For more information, refer to Operation GUID on page 54.

get video settings response

Name	Type	Description
instant_rewind_time_in_seconds	int32	Determines the amount of time from the current time, in seconds, to rewind the video for playback. This value is configured on the **System Adinistration > Administration > System Options > Video Options** tab. Default value = 30.
time_limited_playback_restriction_in_minutes	int32	Determines the amount of time from the current time, in minutes, that the user is allowed to play back recorded video. For more information, refer to “Limit Video Viewing to Time-limited Playback Only” in the *System Administration User Guide*. Default value = 480.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Maps

get maps

Retrieves a page of maps from the OnGuard database.

Note: Version 1.0 of this method supports string encoding.

REST Request URL: `GET /api/access/onguard/openaccess/maps?version=value`

get maps

Name	Type	Required	Description
map_filter	string	no	The filter based on map properties. The clause text used to count only those instances that match a given attribute. For example, `name like "%MAP%"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. This parameter supports filtering with the following properties: • map_id • name • width • height
map_device_ filter	string	no	The filter based on map device properties. The clause text used to count only those instances that match a given attribute. For example, `device_id=1 and panel_id=1`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. This parameter supports filtering with the following properties: • map_item_id • panel_id • device_id • input_device_id • pos_x • pos_y • pos_z • icon_id • icon_group_id • custom_text
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.

get maps

Name	Type	Required	Description
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
order_by	string	no	A field- or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by map_id. Fields must be valid properties of the requested object type. This parameter supports sorting with the following properties: • map_id • name • width • height

get maps response

Name	Type	Description
item_list	list	A list of maps. Each map in the list has following properties: • map_id • name • width • height • device_count
map_id	int32	The internal database ID of the map. Key field.
name	string	The display name of the map.
width	int32	The width of the map.
height	int32	The height of the map.
device_count	int32	The device count in the map.
count	int32	The number of maps returned.
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.

get maps response

Name	Type	Description
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get map devices

Retrieves a page of map devices from the OnGuard database.

Note: Version 1.0 of this method supports string encoding.

REST Request URL: `GET /api/access/onguard/openaccess/maps/{id}/devices?version=value`

get map devices

Name	Type	Required	Description
id	int32	yes	The ID of the map to retrieve its devices.
filter	string	no	The filter based on map device properties. The clause text used to count only those instances that match a given attribute. For example, `device_id=1 and panel_id=1`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. This parameter supports filtering with the following properties: • map_item_id • panel_id • device_id • input_device_id • pos_x • pos_y • pos_z • icon_id • icon_group_id • custom_text
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.

get map devices

Name	Type	Required	Description
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
order_by	string	no	A field- or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by map_item_id. Fields must be valid properties of the requested object type. This parameter supports sorting with the following properties: • map_item_id • panel_id • device_id • input_device_id • pos_x • pos_y • pos_z • icon_id • icon_group_id • custom_text

get map devices response

Name	Type	Description
item_list	list	A list of map devices. Each map device in the list has following properties: • map_item_id • panel_id • device_id • input_device_id • pos_x • pos_y • pos_z • icon_id • icon_group_id • custom_text • label_color • font_size • font_weight • font_face • is_italic • is_strikeout • is_underline • label_text • label_width • label_height • label_alignment
map_item_id	int32	The internal database ID of the map device. Key field.
panel_id	int32	The ID of the panel.
device_id	int32	The ID of the device.
input_device_id	int32	The ID of the input device.
pos_x	int32	The icon's X position.
pos_y	int32	The icon's Y position.
pos_z	int32	The icon's Z position.
map_item_type	int32	The type of map item.
icon_id	int32	The ID of the icon.
icon_group_id	int32	The ID of the icon group.
custom_text	string	The custom text.
label_color	string	The label color, in HEX format.
font_size	int32	The font size.
font_weight	int32	The font weight.
font_face	string	The font face name.
is_italic	boolean	Indicates if the label font style is italic.

get map devices response

Name	Type	Description
is_strikeout	boolean	Indicates if the label is shown with strikeout marks.
is_underline	boolean	Indicates if the label is shown underlined.
label_text	string	The label text. Only available when this map item is text.
label_width	int32	The label width. Only available when this map item is text.
label_height	int32	The label height. Only available when this map item is text.
label_alignment	int32	The label alignment. Only available when this map item is text.
count	int32	The number of map devices returned.
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get map icon groups

Retrieves a page of map icon groups from the OnGuard database.

Note: Version 1.0 of this method supports string encoding.

REST Request URL: `GET /api/access/onguard/openaccess/map_icon_groups?version=value`

get map icon groups

Name	Type	Required	Description
filter	string	no	The filter based on map icon group properties. The clause text used to count only those instances that match a given attribute. For example, `name like "%AAA%"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. This parameter supports filtering with the following properties: • icon_group_id • name • map_item_type
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
order_by	string	no	A field- or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by icon_group_id. Fields must be valid properties of the requested object type. This parameter supports sorting with the following properties: • icon_group_id • name • map_item_type

get map icon groups response

Name	Type	Description
item_list	list	A list of map icon groups. Each map icon group in the list has following properties: • icon_group_id • name • map_item_type • map_item_group_members
icon_group_id	int32	The internal database ID of the map icon group. Key field.
name	string	The display name of the map icon group.

get map icon groups response

Name	Type	Description
map_item_type	int32	The type of map item. Possible values: • Panel = 1 • Reader = 2 • Alarm panel = 3 • Input = 4 • Output = 5 • Map = 6 • Text = 7 • Alarm mask group = 8 • Function list = 9 • Reader auxiliary input = 10 • Reader auxiliary output = 11 • Reader group = 12 • Input group = 13 • Output group = 14 • CCTV panel = 15 • Fire panel = 16 • CCTV camera = 17 • CCTV monitor = 18 • Fire device = 19 • Intercom = 20 • Fire input/output = 21 • Intercom panel = 22 • Personal safety panel = 23 • Camera group = 24 • Off-line lock panel= 25 • Switcher panel = 26 • PC panel = 27 • Receiver = 28 • Receiver account = 29 • Account zone = 30 • Muster zone = 31 • Safe location = 32 • Normal anti-passback area = 33 • Intrusion panel = 34 • Intrusion zone = 35 • Intrusion on relay = 36 • Intrusion off relay = 37 • Intrusion door = 38 • Intrusion area = 39 • Intercom station input = 40 • Intercom station output = 41 • Intercom station door = 42 • Action group = 43 • SNMP manager = 44 • SNMP agent = 45

get map icon groups response

Name	Type	Description
map_item_type (continued)	int32	• Open Platform Communications (OPC) connection = 46 • OpenIT source= 47 • OPC source = 48 • Point of Sale (POS) panel = 52 • POS register = 53 • CCTV camera input = 56 • CCTV camera output = 57 • CCTV video monitor = 58 • Video monitor group = 59 • Elevator panel = 60 • Elevator keypad terminal = 61 • OpenIT device = 62 • OpenIT sub-device = 63 • Power supply = 64; • Intrusion reader = 67 • Intrusion RAS = 68 • Intrusion DGP = 69
map_item_ group_members	list	A list of map icon group members. Each map icon group member in the list has following properties: • map_item_state • icon_id See below for those properties.
map_item_state	int32	The map item state.
icon_id	int32	The map icon ID.
count	int32	The number of map icon groups returned.
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get map icons

Retrieves a page of map icons from the OnGuard database.

Note: Version 1.0 of this method supports string encoding.

REST Request URL: `GET /api/access/onguard/openaccess/map_icons?version=value`

get map icons

Name	Type	Required	Description
filter	string	no	The filter based on map icon properties. The clause text used to count only those instances that match a given attribute. For example, `name like "%AAA%"`. **Note:** You must use double-quotes around string delimiters when filtering. Single-quotes will result in an InvalidQuery error. This parameter supports filtering with the following properties: • icon_id • name
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	no	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
order_by	string	no	A field- or comma-separated list of fields to use for sorting the instances when performing paging. If not provided, results are ordered by icon_id. Fields must be valid properties of the requested object type. This parameter supports sorting with the following properties: • icon_id • name

get map icons response

Name	Type	Description
item_list	list	A list of map icons. Each map icon in the list has following properties: • icon_id • name • blob
icon_id	int32	The internal database ID of the map icon. Key field.
name	string	The display name of the map icon.
blob	string	The data of the map icon. Converted to base64-encoded string.
count	int32	The number of map icons returned.
page_number	int32	The page number to return when a subset (page) of instances is requested. Used in conjunction with page_size. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	The page size, or number of instances per page, to be returned when a subset (page) of instances is requested. Used in conjunction with page_number. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (page_size) that can be retrieved with a single request is 100.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Device Groups

get device groups

Retrieves a list of device groups. For more information, refer to Lnl_DeviceGroup on page 256.

REST Request URL: `GET /api/access/onguard/openaccess/device_groups?version=value&filter=value&page_number=value&page_size=value&order_by=value`

get device groups

Name	Type	Required	Description
filter	string	no	The filter, based on the device group properties. For more information refer to Searching for Objects on page 43 and Lnl_DeviceGroup on page 256.

get device groups

Name	Type	Required	Description
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used with **page_size**. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	no	The page size, or number of instances per page, to return when a subset (page) of instances is requested. Used with **page_number**. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (**page_size**) that can be retrieved with a single request is 100.
order_by	string	no	A field or comma-separated list of fields to use for sorting list of printers. If not provided, results are ordered by ID. Additional order_by Details on page 92.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get device groups response

Name	Type	Description
item_list	list	A list of items returned if instances exist. If a valid order_by parameter was provided in the request, then the list of items is sorted accordingly. If present, each item consists of the properties described below:
• ID	int32	The ID of the device group.
• Type	int32	The type of device group: 0: Reader Group 1: Input Group 2: Output Group 3: Camera Group 4: Monitor Group 5: Lockdown Group
• Name	string	The name of the device group.
• SegmentID	int32	The ID of the segment to which the device group belongs. This property is only available if segmentation is enabled.

get device groups response

Name	Type	Description
• DatabaseID	int32	The database identifier in an Enterprise system that identifies the server containing the device group.
• Devices	list	A list of items belonging to a given device group. If present, each item consists of the properties described below.
• PanelId	int32	The ID of the panel.
• PanelType	int32	The type of the panel.
• PanelTypeName	string	The name of the panel type.
• DeviceId	int32	The ID of the device.
• DeviceName	string	The name of the device.
• InputOutputId	int32	The ID of the input or output.
count	int32	The total number of records in the filter result.
page_number	int32	The page number of the requested subset (page) of instances returned. Same as corresponding input parameter, or the default value if not provided as input.
page_size	int32	The page size, or number of instances per page, returned when a subset (page) of instances is requested. Same as the corresponding input parameter, or the default value if not provided as input.
total_pages	int32	The total number of pages, given the existing number of instances (total_items) and the page_size being used.
total_items	int32	The total existing number of instances of the object being requested.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

post device group

Adds a device group. For more information, refer to Lnl_DeviceGroup on page 256.

REST Request URL: `POST /api/access/onguard/openaccess/device_group?version=value`

post device group

Name	Type	Required	Description
Type	int32	yes	The type of device group: 0: Reader Group 1: Input Group 2: Output Group 3: Camera Group 4: Monitor Group 5: Lockdown Group
Name	string	yes	The name of the device group.
SegmentID	int32	yes	The ID of the segment to which the device group belongs. This property is only available if segmentation is enabled.
Devices	list	no	A list of items belonging to a given device group. If present, each item consists of the properties described below:
• PanelId	int32	yes	The ID of the panel.
• DeviceId	int32	yes	The ID of the device.
• InputOutputId	int32	yes	The ID of the input or output.
operation_description	string	no	For more information, refer to Operation Description on page 55.
operation_guid	GUID (string)	no	For more information, refer to Operation GUID on page 54.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

post device group response

Name	Type	Description
ID	int32	The ID of the device group.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put device group

Update an existing device group. For more information, refer to Lnl_DeviceGroup on page 256.

REST Request URL: `PUT /api/access/onguard/openaccess/device_group?version=value`

put device group

Name	Type	Required	Description
ID	int32	yes	The ID of the device group.
Name	string	no	The name of the device group.
Devices	list	no	A list of items belonging to a given device group. If present, each item consists of the properties described below:
• PanelId	int32	yes	The ID of the panel.
• DeviceId	int32	yes	The ID of the device.
• InputOutputId	int32	yes	The ID of the input or output.
operation_description	string	no	For more information, refer to Operation Description on page 55.
operation_guid	GUID (string)	no	For more information, refer to Operation GUID on page 54.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

put device group response

Name	Type	Description
ID	int32	The ID of the device group.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete device group

Delete an existing device group. For more information, refer to Lnl_DeviceGroup on page 256.

REST Request URL: `DELETE /api/access/onguard/openaccess/device_group?version=value`

delete device group

Name	Type	Required	Description
ID	int32	yes	The ID of the device group.
operation_description	string	no	For more information, refer to Operation Description on page 55.

delete device group

Name	Type	Required	Description
operation_guid	GUID (string)	no	For more information, refer to Operation GUID on page 54.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

delete device group response

Name	Type	Description
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

Access Panel

get access panel with certs and users

Retrieves a list of access panels with TLS and Peer certificate information, and a list of panel users.

Note: To get the extended view of **Lnl_Panel** instances with certificate information and the controller users list, you must use Lnl_Panel attributes (version 1.9 or later) in the search and **order_by** parameters.

REST Request URL: `GET /api/access/onguard/openaccess/access_panel_with_certs_and_users?filter=value&page_number=value&`

```
page_size=value&order_by=value&version=value&queue=value&
do_not_reset_inactivity_timer=value
```

get access panel with certs and users

Name	Type	Required	Description
do_not_reset_ inactivity_timer	boolean	no	Default = false. If set to true, the OpenAccess call does not reset the inactivity timer to zero. If this parameter is not provided, it is assumed to be false and the call will reset the inactivity timer to zero. This parameter is available to all OpenAccess calls except the following: • get version • get directories • get identity_provider_url • get keepalive • post authentication • delete authentication
page_number	int32	no	The page number to return when a subset (page) of instances is requested. Used with **page_size**. Defaults to the first page (1) if not provided, and if provided, must be numeric.
page_size	int32	no	The page size, or number of instances per page, to return when a subset (page) of instances is requested. Used with **page_number**. Defaults to 20 if not provided, and if provided, must be numeric. For performance reasons, paging is always performed, and the maximum number of instances (**page_size**) that can be retrieved with a single request is 100.
queue	boolean	no	Queues the request as a task, and returns a response identical to `GET /queue/{id}`. Defaults to `false` if not provided.
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get access panel with certs and users response

Name	Type	Description
AccessPanelUsers	list	A list of items returned if Mercury access panel users exist. This information is only visible if the user has **Manage Panel Users** permissions. If present, each item consists of the properties described below:
• ID	int32	Access panel's user ID. *

get access panel with certs and users response

Name	Type	Description
• LastUpdate	string	The date of last user data download from the access panel.
• Level	int32	The user's level on the access panel. *
• Name	string	The name of the access panel user. *
• Type	int32	The user's type on the access panel. *
CommServer	string	The name of the communication server.
ID	int32	The ID of the access panel.
Name	string	The access panel name.
OnlineEnabled	boolean	If true, the controller is marked online.
PanelType	int32	The type of panel. Allowed types are stored in the PANELTYPE OnGuard database table (PANELTYPEID column).
PanelTypeName	string	The name of the panel type.
PeerCertEnabled	boolean	Is true if the peer certificate is enabled.
PeerCertIssuedBy	string	Peer certificate information. *
PeerCertIssuedExpired	string	The expiration date of the certificate. *
PeerCertIssuedStart	string	The start date of the certificate. *
PeerCertIssuedTo	string	Peer certificate information. *
SegmentID	int32	The ID of the segment to which the device group belongs.
TLSCertEnabled	boolean	If true, the TLS certificate is enabled.
TLSCertIssuedBy	string	TLS certificate information. *
TLSCertIssuedExpired	string	The expiration date of the TLS certificate. *
TLSCertIssedStart	string	The start date of the TLS certificate. *
TLSCertIssuedTo	string	TLS certificate information. *
WebConfigLink	string	URL of the access panel’s web configuration page.

* This data is retrieved from the hardware.

Marketplace

get marketplace

Retrieves information about the customer’s LenelS2 Marketplace configuration, based upon the customer’s privacy level configuration.

Notes: The number of items returned in the response depends on the PrivacyLevel parameter configured in the `LNLCONFIG` database table using the `LNLCONFIGID = 276` entry. Available PrivacyLevel values are 0 (default), 1,2,3. For example, to set PrivacyLevel = 1, use the following sql query:

```
update [LNLCONFIG] set [LNLVALUE] = 1 where [LNLCONFIGID] = 276
```

REST Request URL: `GET /api/access/onguard/openaccess/marketplace?version=value`

get marketplace

Name	Type	Required	Description
version	string	yes	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

get marketplace response

Name	Type	Description
URL	string	Marketplace URL. Returned if PrivacyLevel <=2.
product_name	string	Product name. Returned if PrivacyLevel <=1.
og_version	string	OnGuard version. Returned if PrivacyLevel <=1.
system_id	string	System ID with hash. Returned if PrivacyLevel <=0.
metadata	string	Metadata. Returned if PrivacyLevel <=0.
version	string	Used by OpenAccess to maintain backward compatibility as the API is updated. Required string, in the format `"version" : "1.0"`. For more information, refer to Version on page 54.

CHAPTER 6

Event API Reference

Events can be received using the Web Event Bridge. The Web Event Bridge is a SignalR Server running at **/api/access/onguard/openaccess/eventbridge**, which provides a hub named "Outbound". Because the Web Event Bridge is a SignalR Server, it is easiest to use one of the SignalR client APIs. There are SignalR client APIs for C# and JavaScript, and there are sample event subscriber applications provided for both. For help writing SignalR clients, refer to http://www.asp.net/signalr/overview/guide-to-the-api/hubs-api-guide-net-client and http://www.asp.net/signalr/overview/guide-to-the-api/hubs-api-guide-javascript-client.

Note: It is suggested that the server core and client components use SignalR version 2.4.3.

Web Event Bridge Operations

CreateSubscription

Creates a subscription and starts receiving events using the **OnBusinessEventReceived** event handler.

`Method Signature`

CreateSubscription(security : object, eventSubscription : object) : object

For a list of inputs and outputs, refer to add event_subscriptions on page 78.

Parameters

Name	Type	Required	Description
security	object	yes	An object containing the session token and application ID properties for the OpenAccess API.
security.SessionToken	string	yes	An authenticated OpenAccess session token. To authenticate with a session cookie ('OASession'), pass the SessionToken field as an empty string, and the session cookie must be submitted with the CreateSubscription request. For more information, refer to notes beneath this table. To authenticate with a SessionToken header, simply pass the session identifier in the SessionToken field. For more information, refer to notes beneath this table. **Note:** Depending upon the JavaScript API library that you use, the credentials might be included automatically. If not, you must explicitly include the cookie credentials with the OpenAccess API request.
security.ApplicationId	string	yes	An OpenAccess application ID.
eventSubscription	object	yes	An object containing the event subscription parameters.
eventSubscription.description	string	no	An optional description for the event subscription.
eventSubscription.filter	string	no	An optional filter for the event subscription.

Name	Type	Required	Description
eventSubscription.monitoring_zone_id	int32	no	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.

Notes: The message body associated with the CreateSubscription request takes the following form depending upon the authentication mechanism:

Cookie

```
{"H":"outbound","M":"CreateSubscription","A":[{"ApplicationId":
"<APPLICATION-ID>","SessionToken":""},{}],"I":0}
```

SessionToken header

```
{"H":"outbound","M":"CreateSubscription","A":[{"ApplicationId":
"<APPLICATION-ID>","SessionToken":"<SESSION-
IDENTIFER>"},{}],"I":0}
```

Return Value

The created event subscription.

Name	Type	Required	Description
Id	int32	yes	The unique subscription id.
user_id	string	yes	The ID of the user who owns the subscription.
name	string	yes	The unique name of the subscription.
description	string	yes	A description of the subscription.

Name	Type	Required	Description
monitoring_zone_id	int32	yes	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.
filter	string	yes	This optional parameter filters the events that will be received. If no filter is specified, all events will be forwarded to the subscriber. For more information, refer to Using Event Filters with Subscriptions on page 48.
is_durable	boolean	yes	Indicates if this is a durable subscription.
message_broker_hostname	string	yes	The hostname of the message broker where the events will be published.
message_broker_port	int32	yes	The port of the message broker where the events will be published.
requires_secure_connection	boolean	yes	Indicates if an SSL connection should be opened by the message broker where the events will be published.
exchange_name	string	yes	The exchange name on the message broker where the events will be published.
binding_key	string	yes	The unique binding key with which the events will be published on the exchange.
queue_name	string	yes	The unique queue name where the events will be published if the subscription is durable.

Name	Type	Required	Description
created_date	datetime (string)	yes	The time when the subscription was created.
last_updated_date	datetime (string)	yes	The time when the subscription was last updated.

ModifySubscription

Modifies a subscription and starts receiving events using the **OnBusinessEventReceived** event handler.

`Method Signature`

ModifySubscription(security : object, eventSubscription : object) : object

Parameters

Name	Type	Required	Description
security	object	yes	An object containing the session token and application ID properties for the OpenAccess API.
security.SessionToken	string	yes	An authenticated OpenAccess session token.
security.ApplicationId	string	yes	An OpenAccess application ID.
eventSubscription	object	yes	An object containing the event subscription parameters.
eventSubscription.description	string	no	An optional description for the event subscription.
eventSubscription.filter	string	no	An optional filter for the event subscription.
eventSubscription.monitoring_zone_id	int32	no	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.

Return Value

The modified event subscription.

Name	Type	Required	Description
Id	int32	yes	The unique subscription id.
user_id	string	yes	The ID of the user who owns the subscription.
name	string	yes	The unique name of the subscription.
description	string	yes	A description of the subscription.

Name	Type	Required	Description
monitoring_zone_id	int32	yes	This optional parameter allows filtering events and messages related to devices assigned to a specific monitoring zone ID. This parameter requires version 1.2 or later of the **<get/add/modify> event_subscriptions <with id>** method. For the **add** method, 0 is the default value (if the parameter is not specified). In this case, events and messages from all available monitoring zones are sent to the subscriber. Possible values: **id = 0:** all available monitoring zones **id > 0:** selected monitoring zone **Note:** An error is returned if the selected monitoring zone ID is not available, or the user does not have view permissions.
filter	string	yes	This optional parameter filters the events that will be received. If no filter is specified, all events will be forwarded to the subscriber. For more information, refer to Using Event Filters with Subscriptions on page 48.
is_durable	boolean	yes	Indicates if this is a durable subscription.
message_broker_hostname	string	yes	The hostname of the message broker where the events will be published.
message_broker_port	int32	yes	The port of the message broker where the events will be published.
requires_secure_connection	boolean	yes	Indicates if an SSL connection should be opened by the message broker where the events will be published.
exchange_name	string	yes	The exchange name on the message broker where the events will be published.
binding_key	string	yes	The unique binding key with which the events will be published on the exchange.
queue_name	string	yes	The unique queue name where the events will be published if the subscription is durable.

Name	Type	Required	Description
created_date	datetime (string)	yes	The time when the subscription was created.
last_updated_date	datetime (string)	yes	The time when the subscription was last updated.

StopSubscription

Stops receiving events using the **OnBusinessEventReceived** event handler. Also deletes the subscription if it is transient.

Method Signature

StopSubscription()

StartManaging

Starts receiving management messages using the **OnManagementEvent** event handler.

Method Signature

StartManaging(agentName : string)

Parameters

Name	Type	Required	Description
agentName	string	yes	A name to use for the management agent.

StopManaging

Stops receiving management messages using the **OnManagementEvent** event handler.

Method Signature

StopManaging()

ConnectionHeartbeat

Tests the connection, and re-subscribes events if LS Web Event Bridge service is restarted.

Method Signature

ConnectionHeartbeat(security : object, eventSubscription : object) : object

Parameters

Name	Type	Required	Description
security	object	yes	An object containing the session token and application ID properties for the OpenAccess API.
security.SessionToken	string	yes	An authenticated OpenAccess session token.
security.ApplicationId	string	yes	An OpenAccess application ID.
eventSubscription	object	yes	An object containing the event subscription parameters.
eventSubscription.id	int	yes	The unique subscription ID.
eventSubscription.description	string	no	An optional description for the event subscription.
eventSubscription.filter	string	no	An optional filter for the event subscription.

Web Event Bridge Client Event Handlers

Notes: If developing your own application, using WebSockets as the transport improves performance. To do this, target .NET Framework 4.6.1 or later instead of .NET Framework 4.0, as shown in this sample application. WebSockets functions correctly with any version of Windows supported by the OnGuard software.

When the LS Web Event Bridge service is restarted, it loses subscription details for all existing clients. Therefore, clients must re-subscribe to continue receiving events. New transient subscriptions must be created, but durable subscriptions can be re-established with the **ModifySubscription** call (ModifySubscription on page 197).

OnBusinessEventReceived

Called when an event is received.

Event Handler Signature

OnBusinessEventReceived(businessEvent : object)

Parameters

Name	Type	Required	Description
businessEvent	object	yes	The business event, with the properties specific to the given event type. For more information, refer to Hardware Event Reference on page 203, Alarm Acknowledgement Activity Event Reference on page 214, and Software Event Reference on page 215.

OnExceptionRaised

Called when an exception is raised.

Event Handler Signature

OnExceptionRaised(message : string)

Parameters

Name	Type	Required	Description
message	string	yes	The error message describing the exception.

OnConnectionFromMessageBusLost

Called when the connection to the message bus is lost.

Event Handler Signature

OnConnectionFromMessageBusLost()

OnConnectionToMessageBusEstablished

Called when the connection to the message bus is established.

Event Handler Signature

OnConnectionToMessageBusEstablished()

OnManagementEvent

Called when a management event is received.

Event Handler Signature

OnManagementEvent(message : string)

Parameters

Name	Type	Required	Description
message	string	yes	The management message. For example: “Updated Transient subscription 123. Client Id 7ffb8f0a-c38e-41c4-aaad-6e7eaa7f4d24”.

Hardware Event Reference

In OnGuard, events generally originate in the access control hardware and are displayed in Alarm Monitoring. An example is when a reader grants access to a cardholder.

This chapter includes the different categories of events, as well as properties that are common to all events, as included in the following table.

Notes: If an event contains an ID for an item that does not exist in the database, the fields relating to that item are not included in the event. For example, if an access denied event is received with a badge ID of 4, but there is no badge with an ID of 4 in the database, there will be no badge or cardholder properties included in that event.

For a complete list of event types and subtypes, perform a **get instances** call on Lnl_EventType and Lnl_EventSubtypeDefinition. For more information, refer to get instances on page 91, Lnl_EventType on page 262, and Lnl_EventSubtypeDefinition on page 261.

Common Properties for All Hardware Events

Property	Type	Description
alarm_ack_blue_channel	int16	The blue component of the RGB color for the alarm after it is acknowledged (0 to 255).
alarm_ack_green_channel	int16	The green component of the RGB color for the alarm after it is acknowledged (0 to 255).
alarm_ack_red_channel	int16	The red component of the RGB color for the alarm after it is acknowledged (0 to 255).
alarm_active_alarm	boolean	True if this alarm is configured as active, meaning that Alarm Monitoring clients should highlight alarms of this type when they occur.
alarm_aggregate_alarm	boolean	True if this alarm is to be aggregated, meaning that Alarm Monitoring clients should combine all alarms of this type into a single alarm for display purposes.
alarm_blue_channel	int16	The blue component of the RGB color for the alarm (0 to 255).
alarm_change_response	boolean	True if the operator is allowed to change the information provided when acknowledging this alarm type.
alarm_display_alarm	boolean	True if this alarm should be displayed.

Common Properties for All Hardware Events (Continued)

Property	Type	Description
alarm_display_map	boolean	True if a map containing the location of this alarm should be displayed automatically.
alarm_do_not_delete_on_acknowledge	boolean	True if this alarm should not be deleted from the client view after it is acknowledged.
alarm_green_channel	int16	The green component of the RGB color for the alarm (0 to 255).
alarm_login_required_-for_acknowledge	boolean	True if the operator is required to log in when acknowledging this type of alarm.
alarm_must_acknowledge	boolean	True if this alarm must be acknowledged before it can be deleted.
alarm_must_mark_in_progress	boolean	True if this alarm must be marked `In Progress` before it can be deleted.
alarm_print_alarm	boolean	True if this alarm should be printed.
alarm_priority	int16	Alarm priority (0 to 255).
alarm_red_channel	int16	The red component of the RGB color for the alarm (0 to 255).
alarm_response_required	boolean	True if notes are required when acknowledging this alarm.
alarm_show_cardholder	boolean	True if the cardholder view should be shown for this type of alarm.
alarm_video_verify	boolean	True if the video verification view should be shown for this type of alarm.
alarm_visual_notification	boolean	True if the occurrence of this alarm type should be highlighted by, for example, bringing the main alarm monitor window to the foreground.
associated_text	string	Optional text that provides additional information about an event.
business_event_class	string	Type of event. Will always be hardware_event.
device_name	string	Name of the device that is the source of the event.
domain	string	The source domain of an event.
event_parameter	uint32	A parameter that provides additional information about an event.
event_subtype	uint16	A subtype of a class of events defined in the system.
event_type	uint8	A class of events defined in the system and reported by the API that can be further broken down into subtypes. For example, 0 indicates an access granted event and 1 indicates an access denied event.
initiating_event_id	int32	The ID of a previous event that caused the event.

Common Properties for All Hardware Events (Continued)

Property	Type	Description
segment_id	uint32	The segment ID of the source of an event, if segmentation is enabled in the system. Otherwise, the value is null.
source	string	The source of the event encoded in a domain-specific manner as a URI string. For example, a source defined as a UUID should be encoded as *urn:uuid:7673868d-231e-490d-9c4f-19288e7e668d*. Fore more examples, visit: http://example.org/absolute/URI/with/absolute/path/to/resource.txt
timestamp	int64	The time when the event occurred at its source, following the AMQP standard of milliseconds since January 1, 1970 in UTC time.
version	string	The version of this specific event message type. This is a period-delimited string in the format <*major*>.<*minor*>. • A *minor* version change is one in which only fields were added, and a parser that ignores unrecognized fields can still process the message. • A *major* version change is one in which the message structure has changed in a manner that is not backwards compatible with the previous structure. Version is managed on a per event type basis, **not** the version of the application that sent the message. A specific event type is uniquely identified using the ordered list of domain, event type, and version.

The following properties are delivered for controller-based events, which are events for devices that are either controllers or have a root parent device that is a controller:

Properties for Controller-Based Events

Property	Type	Description
alarm_id	int32	ID for the alarm.
alarm_name	string	Name of the alarm.
controller_id	uint16	The ID of the controller for the device that is the source of an event.
controller_name	string	Name of the controller to which the device or subdevice is connected. May also refer to the controller itself.
device_id	uint16	The ID of the device that is the source of an event. A value of 0 indicates that the source of the event is a controller.
device_type	int8	The type of device that generated an event.
event_parameter_description	string	The description of the event parameter. **Note:** This value may be included for events that convey additional information.

Properties for Controller-Based Events

Property	Type	Description
event_source_name	string	The name of the device that generated the event.
controller_time_zone_id	uint16	The time zone where the controller is located.
serial_number	int32	The serial number of the event, as specified by the controller.
subdevice_id	uint16	The ID of the subdevice of a device that is the source of the event. A value of 0 indicates that the source is a device or a controller.
timestamp_processed	int64	The time when the event was processed by the Communication Server, following the AMQP standard of milliseconds since January 1, 1970 in UTC time.

Access Granted Events

When an Access Granted event occurs, subscribers with proper authorization receive the following properties and their values:

Properties for Access Granted Events

Property	Type	Description
access_granted_entry_made	boolean	Indicates if entry was made through the door. Value Range: True, False
area_entering_id	int32	The ID of the area that a cardholder entered, if the corresponding reader is defined to detect when an area is entered.
area_entering_name	string	The name of the area that a cardholder entered.
area_exiting_id	int32	The ID of the area that a cardholder exited, if the corresponding reader is defined to detect when an area is exited.
area_exiting_name	string	The name of the area that a cardholder exited.
badge_extended_id	string	The full Federal Agency Smart Credential Number (FASC-N) or full UUID from a Personal Identity Verification (PIV)-based card or other Federal Information Processing Standard (FIPS) 201-based card.
badge_id	int64	The ID encoded on a badge.
badge_id_str	string	A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
badge_issue_code	uint32	The issue code of the badge.
badge_key	int64	The database record ID of the badge.

Properties for Access Granted Events

Property	Type	Description
badge_key_str	string	A string representation of the badge key. To accurately display badge key, web clients should use this property instead of the badge_key property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
badge_status_name	string	The status of the badge, which must be “Active” if access was granted.
badge_type_name	string	The cardholder’s badge type, as configured in System Administration.
cardholder_first_name	string	The cardholder’s first name, as configured in System Administration.
cardholder_key	int32	The database record ID, which is not displayed in System Administration, but which can be useful when developing custom scripts.
cardholder_last_name	string	The cardholder’s last name, as configured in System Administration.
controller_segment_id	int32	The ID of the controller segment.
event_parameter	int32	A parameter that provides additional information about an event.
event_parameter_description	string	The description of the event parameter. **Note:** This value may be included for events that convey additional information.

Access Denied Events

When an Access Denied event occurs, subscribers with proper authorization receive the following properties and their values:

Properties for Access Denied Events

Property	Type	Description
badge_id	int64	The ID encoded on a badge.
badge_id_str	string	A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
badge_issue_code	uint32	The issue code of the badge.
badge_key	int64	The database record ID of the badge.

Properties for Access Denied Events

Property	Type	Description
badge_key_str	string	A string representation of the badge key. To accurately display badge key, web clients should use this property instead of the badge_key property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
badge_status_name	string	The status of the badge.
badge_type_name	string	The cardholder's badge type, as configured in System Administration.
cardholder_first_name	string	The cardholder's first name, as configured in System Administration.
cardholder_key	int32	The database record ID, which is not displayed in System Administration, but which can be useful when developing custom scripts.
cardholder_last_name	string	The cardholder's last name, as configured in System Administration.

Area Control Events

When an Area Control event occurs, subscribers with proper authorization receive the following properties and their values:

Property for Area Control Events

Property	Type	Description
area_apb_id	int32	The name of an APB area where an event occurred.

Asset Events

When an Asset event occurs, subscribers with proper authorization receive the following properties and their values:

Properties for Asset Events

Property	Type	Description
asset_id	string	The ID of the asset that caused the event.
asset_event_type	int32	The event type of the event associated with the asset event.
asset_event_subtype	int32	The event subtype of the event associated with the asset event.
badge_key	int64	The database record ID of the badge.

Properties for Asset Events

Property	Type	Description
badge_key_str	string	A string representation of the badge key. To accurately display badge key, web clients should use this property instead of the badge_key property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
badge_status_name	string	The status of the badge.
badge_type_name	string	The cardholder's badge type, as configured in System Administration.
cardholder_first_name	string	The cardholder's first name, as configured in System Administration.
cardholder_key	int32	The database ID, which is not displayed in System Administration, but which can be useful when developing custom scripts.
cardholder_last_name	string	The cardholder's last name, as configured in System Administration.

Biometric Events

Properties for Biometric Events

Property	Type	Description
badge_id	int64	The ID encoded on a badge.
badge_id_str	string	A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off.
badge_issue_code	uint32	Issue code associated with the card.
biometric_score	uint32	The biometric score for a biometric card event.

Intercom Events

When an Intercom event occurs, subscribers with proper authorization receive the following properties and their values:

Properties for Intercom Events

Property	Type	Description
intercom_data	uint32	Special intercom data associated with the event.
intercom_line_number	int32	The line number used by special intercom events.

Intrusion Events

When an Intrusion event occurs, subscribers with proper authorization receive the following properties and their values:

Properties for Intrusion Events

Property	Type	Description
intrusion_area_id	uint16	The ID of the area where an intrusion was detected.
intrusion_user_id	string	The ID of the user who will receive information about an intrusion event.
receiver_area_id	uint16	The ID of the area where the receiver is located.
receiver_controller_id	uint16	The ID of the receiver that generated the event.
receiver_line_number	uint16	The line number used by the receiver that generated the event.

Transmitter Events

When a Transmitter event occurs, subscribers with proper authorization receive the following properties and their values:

Properties for Transmitter Events

Property	Type	Description
transmitter_id	int32	The ID of the device transmitting the event.
transmitter_input_id	int32	The ID of the input on the transmitter associated with the event.

Video Events

Properties for Video Events

Property	Type	Description
video_channel	int32	The physical channel to which the camera is connected.
video_start_time	uint32	The start time of the video associated with an event.
video_end_time	uint32	The end time of the video associated with an event.

Status Events

All events are examined, regardless of their message type, to determine if the information indicates a status change. If that is the case, additional information specifying the status change is appended to the event before it is distributed to subscribing clients. The appended information follows the same key/value pair methodology but uses specific keys to indicate that the data specifies status information.

The presence of the key **status_count** indicates that status information is contained in the event and the value is an integer count of the number of status change items that have been appended. In most cases, the count value will be one, but there are cases where the count value can be higher indicating that the source event contained information indicating that multiple state changes have occurred.

For each status change item, there are four key/value pairs that convey the information about that particular status change, as summarized below.

Status Information Key/Value Pairs

Key structure	Type	Value description
status_<*n*>_name	string	The name of the status item that changed, where <*n*> is an integer index specifying which status item the data is for, with 0 for the first status item, 1 for the second, etc.
status_<*n*>_name_text	string	The language translated display text for the name.
status_<*n*>_value	string	The new value for the status item.
status_<*n*>_value_text	string	The language translated display text for the value of the status item.
status_count	int32	An integer specifying the number of status change items appended to the event.

Here is an example of status change information that can be appended to an event:

status_0_name	ReaderMode
status_0_name_text	Reader Mode
status_0_value	ReaderModePinOrCard
status_0_value_text	Pin or Card
status_count	1

Here is an example of status change information where the status item conveys a value and the range of values is not fixed or predefined. For these status items, both the value and value_text elements contain the data.

status_0_name	PanelCardCapacity
status_0_name_text	Panel Card Capacity
status_0_value	500
status_0_value_text	500
status_count	1

Here is an example of status change information containing multiple status items that can be appended to an event:

status_0_name = ReaderAuxInputLineStatus

status_0_name_text = Reader Auxiliary Input Line Status

status_0_value = Alarm

status_0_value_text = Alarm

status_1_name = ReaderAuxInputMasking

status_1_name_text = Reader Auxiliary Input Masking

status_1_value = Unmasked

status_1_value_text = Unmasked

status_count = 2

The table below identifies the status change items currently supported through the OpenAccess API.

Status Change Items

Name	Description
Device-independent status items	
OnlineStatus	The communication status of the device. Values: Online, Offline
FirmwareRevision	The firmware revision of the device. Value: A text string
SerialNumber	The serial number of the device. Value: An integer
Panel status items	
PanelPowerInputStatus	The power input status for a panel. Values: Secure, Alarm
PanelCabinetStatus	The cabinet status for a panel. Values: Secure, Alarm
PanelFirmwareDownloadStatus	The firmware download status for a panel. Values: Completed, In Progress
PanelDownloadStatus	The download status for a panel. Values: Completed, In Progress
PanelEventPollingStatus	The event polling status for a panel. Values: Normal, Stopped
PanelCardCapacity	The maximum number of cards supported by the panel. Value: An integer
PanelCardCount	The current number of cards downloaded to the panel. Value: An integer
Reader status items	
ReaderAuxInputMasking	The masking state of a reader auxiliary input. Values: Masked, Unmasked
ReaderAuxOutputActivation	The activation state of a reader auxiliary output. Values: Activated, Deactivated

Status Change Items (Continued)

Name	Description
ReaderMode	The mode of a reader. Values: Facility Code Only, Card Only, Pin Only, First Card Unlock, Card Unlocked, Locked, Unlocked, Pin or Card, Card and Pin, Cipher or Card, Dual Custody, Escort, Blocked, Secured, Unsecured, Normal
ReaderAuxInputLineStatus	The reader auxiliary input physical line status. Values: Secure, Alarm, Shorted, Open, Grounded, Error
ReaderPowerfailStatus	The power status for a reader. Values: Active, Inactive
ReaderCabinetTamperStatus	The cabinet tamper status for a reader. Values: Active, Inactive
ReaderExternalTamperStatus	The external tamper status for a reader. Values: Active, Inactive
ReaderExtraPowerfailStatus	The extra powerfail status for a reader. Values: Active, Inactive

Example Access Denied Event

```
badge_id: 1
controller_id: 1
device_id: 1
device_type: 0
domain: access
event_subtype: 65
event_type: 1
initiating_event_id: 0
intelligent_video: 0
segment_id: 0
serial_number: 1460010837
source: CommServer@TEST105-248
subdevice_id: 0
timestamp: 1460011160000
timestamp_processed: 1460011160684
transmitter_id: 0
transmitter_input_id: 0
version: 1.0
controller_name: Panel-3300
controller_segment_id: 0
controller_time_zone_id: 16
event_source_name: Reader-AAA
alarm_id: 4100
alarm_name: Denied Access
badge_key: 1
badge_extended_id:
badge_type_name: Employee
badge_status_name: Active
cardholder_first_name: Lisa
cardholder_last_name: Lake
cardholder_key: 1
```

```
32  business_event_class: hardware_event
```

Alarm Acknowledgement Activity Event Reference

The Alarm acknowledgement Activity event is published when an alarm is acknowledged by a user. Subscribers with proper authorization receive the following properties and their values:

Properties for Alarm Acknowledgement Activity Events

Property	Type	Description
controller_id	int16	The ID of the access panel that generated the alarm.
serial_number	int32	The serial number of the alarm.
user_id	string	The ID of the user that submitted the acknowledgement.
acknowledge_notes	string	Optional notes submitted with the acknowledgement.
acknowledge_status	lint32	The status of the acknowledgement that can be one of the following: • 0 Update • 1 Acknowledged without notes • 2 Acknowledged with notes • 3 In Progress
device_id	uint16	The ID of the device that is the source of an event. A value of 0 indicates that the source of the event is a controller.
subdevice_id	uint16	The ID of the subdevice of a device that is the source of the event. A value of 0 indicates that the source is a device or a controller.
event_type	uint8	A class of events defined in the system and reported by the API that can be further broken down into subtypes. For example, 0 indicates an access granted event and 1 indicates an access denied event.
event_id	int32	The ID of the event.
domain	string	The source domain of an event.
source	string	The source of the event encoded in a domain-specific manner as a URI string. For example, a source defined as a UUID should be encoded as *urn:uuid:7673868d-231e-490d-9c4f-19288e7e668d*. Fore more examples, visit: http://example.org/absolute/URI/with/absolute/path/to/resource.txt
timestamp	int64	The time when the event occurred at its source, following the AMQP standard of milliseconds since January 1, 1970 in UTC time.

Properties for Alarm Acknowledgement Activity Events

Property	Type	Description
version	string	The version of this specific event message type. This is a period-delimited string in the format <*major*>.<*minor*>. • A *minor* version change is one in which only fields were added, and a parser that ignores unrecognized fields can still process the message. • A *major* version change is one in which the message structure has changed in a manner that is not backwards compatible with the previous structure. Version is managed on a per event type basis, **not** the version of the application that sent the message. A specific event type is uniquely identified using the ordered list of domain, event type, and version.
business_event_class	string	Type of event. Will always be Acknowledgement Event.

Software Event Reference

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

Users with *all segments* and *view all permissions* can register to receive software events that they have permission to receive. In general, users can view a software event for an object if they could view that object normally. For example, if users do not have permission to view visitors, then they cannot receive software events indicating that a visitor was created, modified, or deleted. Furthermore, if users do not have view permissions for each property of a class, then they can't receive software events for instances of that class. For example, if users can't view the visitor address field (set through the field/page permission groups in System Administration), then they can't view visitor software events.

Note: For all **Add** events, each object property name is prefixed with **new_**. For all **Delete** events, each object property name is prefixed with **old_**. All **Modify** events include both the **new_** and **_old** prefixes.

Common Properties for All Software Events

Property	Type	Description
business_event_class	string	Type of event. Will always be software_event.
object_id	int32	The unique identifier of the software event.
software_event_object_type	string	The software event's object type, such as Cardholder, Visitor, Badge, Visit, VisitEvent, or Account.
software_event_operation_type	string	The software event's operation type, such as Add, Modify, or Delete.
timestamp	int64	The time when the event occurred at its source, following the AMQP standard of milliseconds since January 1, 1970 in UTC time.

Person Directory Account Events

When a Person Directory Account event occurs, subscribers with proper authorization receive the following properties and their values. For more information, refer to Lnl_Account on page 232.

Properties for Person Directory Account Events

Property	Type	Description
AccountID	string	ID of the entry in the external directory.
DirectoryID	string	Internal ID of the directory to which this account belongs.
ID	int32	ID that uniquely identifies this directory account.
PersonID	int32	Internal ID of the person who owns this account.

Badge Events

When a Badge event occurs, subscribers with proper authorization receive the following properties and their values. For more information, refer to Lnl_Badge on page 242.

Properties for Badge Events

Property	Type	Description
ACTIVATE	datetime (string)	Badge activate date in the format `1968-07-31T12:34`. The default is the current date and time.
APBEXEMPT	boolean	Whether the badge is APB exempt.
BADGEKEY	int32	ID that uniquely identifies the badge.
DEACTIVATE	datetime (string)	Badge deactivate date in the format `1968-07-31T12:34`.
DEADBOLT_ OVERRIDE	boolean	If true, the selected cardholder will have deadbolt override privileges, which allows the cardholder to access a door with a deadbolt function mortise lock even when the deadbolt is thrown.
DEFAULT_DOOR	int32	Indicates which elevator door (front or rear) is opened at the Default floor when the badge is presented to a reader associated with the DEC (elevator terminal).
DEFAULT_FLOOR	int32	Indicates the floor number that is called by default when the badge is presented to a reader associated with the DEC (elevator terminal). Configure the Default floor from -128 to 127.
DESCRIPTOR_FLAG	int32	Custom objects that are sent to an elevator dispatch system.
DEST_EXEMPT	boolean	When selected, the badge will not be included in the destination assurance processing and no alarms will be generated if the cardholder violates any of the destination assurance settings.

Properties for Badge Events

Property	Type	Description
EMBOSSED	int32	Any numbers or characters that are embossed on the card. Typically this applies to Proximity cards, which are embossed by the manufacturer prior to delivery.
EXTEND_STRIKE_ HELD	boolean	Use extended strike/held times.
EXTENDED_ID	string	Extended length string identifier that refers to a PIV-based badge in the OnGuard database that generated the event.
ID	int64	The ID of the badge.
ID_str	string	A string representation of the badge ID.
ISSUECODE	int32	Issue code of the badge.
LASTCHANGED	datetime (string)	Date the badge was last changed.
LASTPRINT	datetime (string)	Date the badge was last printed.
PASSAGE_MODE	boolean	If true, the cardholder is allowed to use the card twice (within the lock's unlock duration) to place the lock in an unlock mode for an indefinite duration.
PERSONID	int32	Internal ID of the person who owns this badge.
PRINTS	int32	Number of times badge has been printed.
STATUS	int32	Badge status ID. 1 = Active.
TWO_MAN_TYPE	int32	Specifies the two-man rule designation of the card-holder (either Supervisor or Team Member).
TYPE	int32	Badge type ID.
USELIMIT	int32	Imposes a restriction on the number of times a card-holder can use his/her badge at readers marked with the Enforce Use Limit option. A use limit value of zero (0) indicates that a badge has no uses at readers that enforce a use limit. A use limit value of 255 or that is left empty indicates that the badge has unlimited uses.

Cardholder Events

When a Cardholder event occurs, subscribers with proper authorization receive the following properties and their values. For more information, refer to Lnl_Cardholder on page 255.

Properties for Cardholder Events

Property	Type	Description
ADDR1	string	Cardholder’s address.

Properties for Cardholder Events

Property	Type	Description
ALLOWEDVISITORS	boolean	Whether the **Allowed visitors** checkbox is selected on the Cardholders folder in System Administration.
ASSET_GROUPID	int32	ID of the Asset Group.
BDATE	datetime (string)	Cardholder's birth date, in the format `1968-07-31`.
BUILDING	int32	Cardholder's building.
CITY	string	Cardholder's city.
DATABASEID	int32	The database identifier in an Enterprise system that identifies the system containing the reader to which the badge was last presented.
DEPT	int32	Cardholder's department.
DIVISION	int32	Cardholder's division.
EMAIL	string	Cardholder's email address.
EXT	string	Cardholder's extension.
FIRSTNAME	string	Cardholder's first name.
FLOOR	string	Cardholder's floor.
GUARD	int16	Indicates that the cardholder can be assigned to perform guard tours (1 = guard can perform tours).
ID	int32	Unique cardholder ID.
LASTCHANGED	datetime (string)	Date the record was last changed.
LASTNAME	string	Cardholder's last name.
LOCATION	int32	Cardholder's location.
MIDNAME	string	Cardholder's middle name.
OPHONE	string	Cardholder's office phone number.
PHONE	string	Cardholder's phone number.
PRIMARYSEGMENTID	int32	This property is only visible when cardholders are segmented.
SSNO	string	Cardholder's social security number.
STATE	string	Cardholder's state.
TITLE	int32	Cardholder's title.
VISITOR	boolean	Whether the cardholder is a visitor in the system.
ZIP	string	Cardholder's zip code.

Visitor Events

When a Visitor event occurs, subscribers with proper authorization receive the following properties and their values. For more information, refer to Lnl_Visitor on page 324.

Properties for Visitor Events

Property	Type	Description
ADDRESS	string	Visitor's address.
ASSET_GROUPID	int32	ID of the Asset Group.
CITY	string	Visitor's city.
DATABASEID	int32	The database identifier in an Enterprise system that identifies the system containing the reader to which the badge was last presented.
EMAIL	string	Visitor's email address.
EXT	string	Visitor's extension.
FIRSTNAME	string	Visitor's first name.
GUARD	int16	Indicates that the visitor can be assigned to perform guard tours (1 = guard can perform tours).
ID	int32	Unique visitor ID.
LASTCHANGED	datetime (string)	Date the record was last changed.
LASTNAME	string	Visitor's last name.
MIDNAME	string	Visitor's middle name.
OPHONE	string	Visitor's office phone number.
ORGANIZATION	string	Visitor's organization.
PRIMARYSEGMENTID	int32	This property is only visible when visitors are segmented.
SSNO	string	Visitor's social security number.
STATE	string	Visitor's state.
TITLE	string	Visitor's title.
VISITOR	boolean	Whether the visitor is a visitor in the system.
ZIP	string	Visitor's zip code.

Visit Events

When a Visit event occurs, subscribers with proper authorization receive the following properties and their values. For more information, refer to Lnl_Visit on page 320.

Properties for Visit Events

Property	Type	Description
CARDHOLDERID	int32	The ID for the visitor's host.
ID	int32	Unique visit ID.
LASTCHANGED	datetime (string)	The date and time the visit was last changed, in UTC time.
PURPOSE	string	The purpose of the visit.
SCHEDULED_TIMEIN	datetime (string)	The scheduled time the visitor will arrive for the visit.
SCHEDULED_ TIMEOUT	datetime (string)	The scheduled time the visitor will leave from the visit.
STATUS	int16	The status of the visit.
TIMEIN	datetime (string)	The actual time the visitor arrived for the visit, in UTC time.
TIMEOUT	datetime (string)	The actual time the visitor left the visit, in UTC time.
TYPE	int32	System field.
VISIT_EVENTID	int32	The ID of the visit event.
VISIT_KEY	string	A unique identifier assigned to a scheduled visit, used to sign visitors in or out.
VISITORID	int32	The ID of the visitor.

VisitEvent Events

When a VisitEvent event occurs, subscribers with proper authorization receive the following properties and their values. For more information, refer to Lnl_VisitEvent on page 323.

Properties for VisitEvent Events

Property	Type	Description
CardholderID	int32	The host of the visit event.
DatabaseID	int32	The database identifier in an Enterprise system that identifies the system containing the event data.
DelegateID	int32	The person who schedules or maintains the event instead of the host.
ID	int32	Unique visitor event ID.
LastChanged	datetime (string)	The last time the properties of the visit event changed, in UTC time.

Properties for VisitEvent Events

Property	Type	Description
Name	string	The user-friendly name of this object.
Scheduled_TimeIn	datetime (string)	The time the visit event is scheduled to start.
Scheduled_TimeOut	datetime (string)	The time the visit event is scheduled to complete.
SignInLocationID	int32	The ID of the visitor sign in location.

Example Add Cardholder Event

```
business_event_class: software_event
object_id: 2
software_event_object_type: Cardholder
software_event_operation_type: Add
timestamp: 1460011160000
new_ADDR1: 1212 Pittsford-Victor Rd.
new_ALLOWEDVISITORS: 1
new_ASSET_GROUPID: 0
new_BDATE: 01/01/1965
new_BUILDING: 0
new_CITY: Rochester
new_DATABASEID: 1
new_DEPT: 0
new_DIVISION: 0
new_EMAIL: user@abc.com
new_EXT: 5555
new_FIRSTNAME: William
new_FLOOR: 1
new_GUARD: 0
new_ID: 2
new_LASTCHANGED: 1477928433000
new_LASTNAME: Smith
new_LOCATION: 0
new_MIDNAME: Thomas
new_OPHONE: 555-555-5555
new_PHONE: 555-555-1212
new_PRIMARYSEGMENTID: 0
new_SSNO: 555-55-5555
new_STATE: NY
new_TITLE: 0
new_VISITOR: 0
new_ZIP: 14534
```

CHAPTER 7

Data and Association Class Reference

Data Classes

For more information about each data class, execute a get type call. For more information, refer to get type on page 88.

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

The **Access** attribute is not configurable. It describes if a class or property can be modified by the user or is processed internally by OnGuard.

For example such properties as **internal ID** and the **LASTCHANGED** timestamp are processed internally and cannot be modified by user.

In the following tables, the **Access** attribute might have the following values:

- **view** indicates that the property is view only and not editable. A write attempt of any value will return an error.
- **read** indicates that the property is editable on **Add only**.
- **edit** indicates that the property is always editable.

In addition to the Access attribute, User-Defined Field (UDF) properties include a configurable display **permission** attribute. You can configure this attribute in **System Administration > Users > Field/Page Permission Groups**.

For more information, refer to “Field/Page Permission Groups Form” in the *System Administration User Guide*.

The **permission** attribute might have the following values:

- **none** indicates that the user has no permission to access this property. A **null** value is always returned in response to a GET request.
- **view** indicates that the user can read the property by issuing a GET request.
- **edit** indicates that the user can read the property by issuing a GET request, and can also edit by issuing PUT or POST requests.

DatabaseID only appears as a property when the OnGuard system is an Enterprise system. For more information, refer to get enterprise settings on page 158.

SEGMENTID only appears as a property in data classes that support segmentation when segmentation for that class is enabled. For more information, refer to get segmentation_settings on page 165 and Lnl_Segment on page 311. Restarting the LS OpenAccess service is required when making segmentation changes.

Lnl_AccessGroup

Description: An access group defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	SEGMENTID	Segment to which the access group belongs.	View
string	NAME	Display name.	View

Methods:

void AssignGroup([in]int32 badgeKey);

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

Parameters:

badgeKey - int32 internal ID of the badge to which the access levels are assigned.

Lnl_AccessLevel

Description: An access level defined in the security system.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
boolean	AvailableForRequest	The access level is available to be requested.	Edit

Type	Name	Description	Access
boolean	DoubleCardAuthority	Grants authority to a badge to toggle a reader between a secured and unsecured state when the badge is presented twice, with the second presentation within 5 seconds of the first.	Edit
boolean	DownloadToIntelligentReaders	Level is download to Intelligent Readers	Edit
int32	EscortMode	If **Enable extended options** is selected in **System Administration > Administration > System Options > Access Levels/ Assets**, then EscortMode can be updated using a POST/PUT Lnl_AccessLevel call. Possible values are: • **0** - Not an escort and does not require an escort • **1** - Is an escort • **2** - Requires an escort **Note:** This property is returned in the response regardless of whether a particular access level has extended options enabled, as long as at least one access level in the system has extended options enabled.	Edit
boolean	FirstCardUnlock	First Card Unlocks the reader	Edit
boolean	GlobalIntrusionCommand-Authority	Users with this access level have global intrusion command authority.	Edit
boolean	HasCommandAuthority	Command authority is enabled for the access level	Edit
int32	ID	Internal database ID. Key field.	View
string	Name	Display name.	Edit
int32	SegmentID	Segment to which the access level belongs.	Read

Lnl_AccessLevelAssignment

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

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ACCESSLEVELID	Lnl_AccessLevel.ID - ID of the access level. Key field.	Read
int32	BADGEKEY	Lnl_Badge.BADGEKEY - BadgeKey of the badge. Key field.	Read
datetime (string)	ACTIVATE	Date and time when this assignment will become active.	Read
datetime (string)	DEACTIVATE	Date and time when this assignment will become inactive.	Read

Notes: A successful response indicates that the badge and access level assignment have reached the database. The successful response does not indicate that the assignment has reached the access panel. There might be a delay before the assignment reaches the panel.

Version 1.3 or later of this call: If the **ACTIVATE** or **DEACTIVATE** properties include a timezone offset or seconds (for example, the "22-04:00" in "`2022-09-25T08:11:22-04:00`"), the offset and seconds are ignored, and the date/time is stored in the database without an offset or seconds (for the previous example, "`2022-09-25T08:11:00`" is stored in the database). A warning message is returned that the timezone offset or the seconds were ignored.

The following table describes how OpenAccess uses cardholder permissions and Area Access Manager levels to determine which access levels the authenticated OpenAccess user who is making the call can assign.

Does authenticated OpenAccess user have permission group, badge, and "Modify Access Level Assignment" permissions?	Does authenticated OpenAccess user have Area Access Manager levels defined?	The authenticated OpenAccess user can assign these access levels
Yes	Yes	All
Yes	No	All
No	Yes	Only Area Access Manager access levels

Does authenticated OpenAccess user have permission group, badge, and "Modify Access Level Assignment" permissions?	Does authenticated OpenAccess user have Area Access Manager levels defined?	The authenticated OpenAccess user can assign these access levels
No	No	None

Note: If the authenticated OpenAccess user only has Area Access Manager access levels defined, all access levels in the AssignLevel array must be contained within the authenticated OpenAccess user's Area Access Manager access levels. For example, if the authenticated OpenAccess user has access levels 1 and 2, then the authenticated OpenAccess user cannot assign access levels 1, 2, and 3, and the entire access level assignment attempt will fail.

Lnl_AccessLevelManaged

Description: View all access levels that can be managed by Access Manager users.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Access level ID.	View
int32	SegmentID	Segment ID to which the access level belongs.	View
string	Name	Access level name.	View
boolean	AvailableForRequest	True if this access level can be requested.	View

Lnl_AccessLevelReaderAssignment

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

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	AccessLevelID	Access level to which the link belongs. Key field.	View
int32	PanelID	Lnl_Panel which is linked to this level. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderID	Lnl_Reader ID which is linked to this level. Key field.	View
string	AccessLevelName	Name of the Lnl_AccessLevel.	View
boolean	AvailableForRequest	True if this access level can be requested.	View
string	ReaderFriendlyName	The descriptive name for the Lnl_Reader.	View
string	ReaderName	The display name of the reader.	View
int32	TimezoneID	Lnl_Timezone in which this level is active	View
string	TimezoneName	Name of the Lnl_Timezone.	View

Lnl_AccessPanelUser

Description: Web users of the access panel.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	PanelID	The controller's ID.	View
int32	UserID	The user's ID on the controller.	View
string	Login	The user's login on the controller.	View
int32	Level	The user's level on the controller.	View
string	Password	The user's password on the controller.	View
string	LastUpdate	The date the controller was last updated.	View

Methods:

void UpdateAccessPanelUser()

Sets the access panel web user's password, in the following format:

{

"property_value_map": {"PanelID":2,"UserID":1}

,"in_parameter_value_map": { "Login":"test2","Level":1,"Password":"testpassword"}

}

Lnl_AccessRequest

Description: A request raised by a person for accessing access levels and readers.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Name of the associated access level or reader.	View
int32	PersonID	Internal ID of the person who requested access to the access level or reader. See Lnl_Person.ID.	View
int32	Type	Request type ID: 0: Reader 1: AccessLevel	View
int32	Status	Request status ID: 0: Submitted 1: Approved 2: OnHold 3: Denied	View
datetime (string)	StartDate	Start date the cardholder requests for access level or reader.	View
datetime (string)	EndDate	End date the cardholder requests for access level or reader.	View
int32	SubmittedByUserID	The user ID of the user who submits the request.	View
int32	ApprovedByUserID	The user ID of the user who approves the request.	View

Type	Name	Description	Access
int32	DeniedByUserID	The user ID of the user who denied the request.	View
int32	OnHoldByUserID	The user ID of the user who put the request on hold.	View
string	SubmittedNote	Notes entered when submitting this request.	View
string	ApprovedNote	Notes entered when approving this request.	View
string	DeniedNote	Notes entered when denying this request.	View
string	OnHoldNote	Notes entered when putting this request on hold.	View
datetime (string)	SubmittedDate	The date and time when the request was submitted.	View
datetime (string)	ApprovedDate	The date and time when the request was approved.	View
datetime (string)	DeniedDate	The date and time when the request was denied.	View
datetime (string)	OnHoldDate	The date and time when the request was put on hold.	View
boolean	EmailCardholder	Whether the cardholder is notified.	View
boolean	EmailAccessManager	Whether the approver is notified.	View

Lnl_AccessLevelRequest

Description: A request raised by a person for accessing access levels.

Abstract: No

Access: View/Add

Superclass: Lnl_AccessRequest

Platforms: OnGuard

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Name of the associated access level.	View
int32	AccessLevelID	Access level to which access request should be submitted. Key field.	Read

Type	Name	Description	Access
int32	PersonID	Internal ID of the person who requested access for AccessLevel. Key field. See Lnl_Person.ID.	Read
int32	Type	Request type ID: 1: AccessLevel	View
int32	Status	Request status ID: 0: Submitted 1: Approved 2: OnHold 3: Denied	View
datetime (string)	StartDate	Start date the cardholder requests for Accesslevel.	Read
datetime (string)	EndDate	End date the cardholder requests for Accesslevel.	Read
int32	SubmittedByUserID	The user ID of the user who submits the request.	View
int32	ApprovedByUserID	The user ID of the user who approves the request.	View
int32	DeniedByUserID	The user ID of the user who denied the request.	View
int32	OnHoldByUserID	The user ID of the user who put the request on hold.	View
string	SubmittedNote	Notes entered when submitting this request.	Read
string	ApprovedNote	Notes entered when approving this request.	View
string	DeniedNote	Notes entered when denying this request.	View
string	OnHoldNote	Notes entered when putting this request on hold.	View
datetime (string)	SubmittedDate	The date and time when the request was submitted.	View
datetime (string)	ApprovedDate	The date and time when the request was approved.	View
datetime (string)	DeniedDate	The date and time when the request was denied.	View
datetime (string)	OnHoldDate	The date and time when the request was put on hold.	View

Type	Name	Description	Access
boolean	EmailCardholder	Whether the cardholder is notified.	Read
boolean	EmailAccessManager	Whether the approver is notified.	Read

Methods:

void Approve([in] string Note, [in] boolean EmailCardholder);

Approves the AccessLevel Request. setting ApprovedDate to current date/time.

void Deny([in] string Note, [in] boolean EmailCardholder);

Denies the AccessLevel Request. setting DeniedDate to current date/time.

void Hold([in] string Note, [in] boolean EmailCardholder);

Holds the AccessLevel Request. setting OnHoldDate to current date/time.

Parameters:

Note : Notes when the request is approved, denied and put on hold.

EmailCardholder : Whether the cardholder should be notified.

Lnl_Account

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

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	AccountID	ID of the entry in the external directory. For example, with Microsoft directories, this property would contain the account's security identifier (SID).	Read
string	DirectoryID	Internal ID of the directory to which this account belongs.	Read
int32	PersonID	Internal ID of the person who owns this account. See Lnl_Person.ID.	Read

Lnl_AlarmAckHistory

Description: Records a change in the acknowledgement status of an OnGuard alarm.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	AckNote	The text entered by the operator when acknowledging an alarm.	View
int32	AckStatus	The status of the alarm, with possible values: 0: Not acknowledged 1: Acknowledged 2: Acknowledged with note 3: Marked in-progress	View
int32	AckTimeUTC	The date and time when the acknowledgement occurred, in the format `YYYY-MM-DDTHH:MM:SS[+-]HH:00`.	View
int32	ID	The internal ID of the acknowledgement entry.	View
int32	PanelID	The ID if the access panel with which the alarm is associated.	View
int32	SerialNumber	The serial number of the acknowledged alarm.	View
int32	UserID	the user ID of the user who acknowledged the alarm.	View

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 with Lnl_Alarm instances, which come in with all security events that come through the Communication Server.

Note: If **get instances** is called with an earlier version than 1.7, then text instructions are required in order for an instance from this alarm class to appear in OpenAccess. Text instructions are created using the **System Administration > Monitoring > Alarms > Alarm Configuration** form.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
boolean	AckNotesRequired	True if notes are required when acknowledging this alarm type.	View
boolean	Active	True if the alarm type is configured as Active, meaning the alarm monitoring clients should highlight alarms of this type when they occur.	View
boolean	Aggregate	True if alarms of this type will be aggregated, meaning that alarm monitoring clients should combine all alarms of this type into a single alarm for display purposes.	View
boolean	ChangeResponse	True if it should be allowed for the operator to change the information provided when acknowledging this alarm type.	View
string	Description	Parameter description.	View
boolean	DisplayAlarm	True if this alarm should be displayed.	View
boolean	DisplayMap	True if a map containing the location of this alarm should be shown automatically.	View
boolean	DoNotDeleteOn-Acknowledge	True if alarms of this type should not be deleted from the client view when they are acknowledged.	View
int32	Flags	An integer value representing the combined values of all of the above boolean values.	View
int32	ID	Internal database ID. Key field.	View

Type	Name	Description	Access
boolean	LoginRequiredFor-Acknowledge	True if the operator is required to log in when acknowledging this alarm type.	View
boolean	MustAcknowledge	True if alarms of this type must be acknowledged before they can be deleted.	View
boolean	MustMarkInProgress	True if alarms of this type must be marked “In Progress” before they can be deleted.	View
boolean	PrintAlarm	True if this alarm should be printed.	View
int32	Priority	Alarm priority (0-255)	View
int32	SegmentID	Segment to which the alarm definition belongs.	View
boolean	ShowCardholder	True if the cardholder view should be shown for this alarm type.	View
string	TextInstructionName	Text instruction name. Starting with version 1.7 of **get instances**, a null value is returned if a **text instruction** is not assigned to a defined alarm.	View
string	TextInstructionData	Text instruction. Starting with version 1.7 of **get instances**, a null value is returned if a **text instruction** is not assigned to a defined alarm.	View
boolean	VideoVerify	True if the video verification view should be shown for this alarm type.	View
boolean	VisualNotification	True if the occurrence of this alarm type should be highlighted by, for example, bringing the main alarm monitor window to the foreground.	View

Lnl_AlarmInput

Description: Retrieves the hardware status for the device. Inherits from Lnl_Input, described below. Implements the input control methods and represents an alarm input found on an input control module.

Abstract: No

Access: View

Superclass: Lnl_Input

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	AlarmPanelID	The ID of the associated alarm panel.	View
string	AlarmPanelName	Name of the alarm panel to which this input belongs. Reference to Lnl_AlarmPanel.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	HostName	The name of the workstation where the Communication Server associated with the alarm input's panel is running.	View
int32	ID	Internal database ID. Key field.	View
int32	InputID	The input number configured for this input.	View
string	Name	The name of the alarm input.	View
int32	PanelID	The ID of the associated access panel. Reference to Lnl_Panel.ID.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

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)

Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:	
ALRM_STATUS_SECURE	0
ALRM_STATUS_ACTIVE	1
ALRM_STATUS_GND_FLT	2
ALRM_STATUS_SHRT_FLT	3
ALRM_STATUS_OPEN_FLT	4
ALRM_STATUS_GEN_FLT	5

Lnl_AlarmOutput

Description: Retrieves the hardware status for the device. Inherits from Lnl_Output, described below. 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 LenelS2 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.

Abstract: No

Access: View

Superclass: Lnl_Output

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	AlarmPanelID	The ID number of the associated alarm panel.	View

Type	Name	Description	Access
string	AlarmPanelName	Name of the alarm panel to which this output belongs. Reference to Lnl_AlarmPanel.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	Duration	The duration of the alarm, in seconds.	View
string	HostName	The name of the workstation where the Communication Server associated with the alarm output's panel is running.	View
int32	ID	Internal database ID. Key field.	View
string	Name	The name of the associated alarm output.	View
int32	OutputID	The ID number of the associated alarm output.	View
int32	PanelID	The ID number of the associated access panel. Reference to Lnl_Panel.ID.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Methods:

void Activate()

Sends a command to activate a specific alarm output.

void Deactivate()

Sends a command to deactivate a specific alarm output.

void Pulse()

Sends a momentary pulse command to a specific alarm output.

void GetHardwareStatus([out] uint32 Status)

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_AlarmPanel

Description: Retrieves the hardware status for the device. This class represents the Alarm input or output control module.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	PanelId	The ID of the associated access panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ControlType	The type of alarm panel.	View
string	Name	The name of the associated alarm panel.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Methods:

void GetHardwareStatus([out] uint32 Status)

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	1
OPTIONS_MISMATCH_STATUS	Options Mismatch	2
CABINET_TAMPER	Cabinet Tamper	4
POWER_FAIL	Power Failure	8

Lnl_AlarmVideoConfiguration

Description: Retrieves the alarm video configuration for the device.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard, Magic Monitor

Properties:

Type	Name	Description	Access
boolean	AcknowledgeOnClose	A video alarm event will be acknowledged when the video is closed. 0 = no, 1 = yes. Default = 0.	View
int32	AlarmID	ID for the alarm.	View
int32	AutoLaunchVideo	Automatically launch video on alarm. 0= no, 1 = live video only, 2 = recorded video only, 3 = live and recorded video. Default = 0.	View
int32	CloseOnAcknowledge	Close the video player when the alarm is acknowledged. 0 = no, 1 = yes. Default = 0.	View
boolean	DisplayOverlay	Show the IV overlay. 0 = no, 1 = yes. Default = 0.	View
boolean	PlayPrerollFirst	When the video player is launched on alarm automatically, begin the playback with the video pre-roll. 0 = no, 1 = yes. Default = 0.	View
boolean	PlayPrerollOnPlay	Begin the video playback with the video pre-roll. 0 = no, 1 = yes. Default = 0.	View

Type	Name	Description	Access
int32	RecordedPlaybackOptions	Recorded playback options. 0 = play from pre-roll, 1 = pause at alarm time, 2 = pause at pre-roll start time. Default = 0.	View
int32	StartInMatrixMode	Start the video player in matrix mode. 0 = no, 1 = yes. Default = 0.	View
int32	UnattendedPlayerTimeout	Automatically close an unattended video player after the configured number of seconds. Default = 60.	View
int32	VideoEventLocking	Video event locking. 0 = none, 1 = begin, 2 = end. Default = 0.	View
int32	VideoEventRecording	Video event recording. 0 = none, 1 = begin, 2 = end. Default = 0.	View
int32	VideoEventTimeout	Video event timeout in seconds. Default = 0.	View

Lnl_Area

Description: An APB area defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	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.

void MoveBadge([in] int32 *areaID*, [in] int64 *badgeID*, [in] string *badgeID_str,* [in] int32 *panelID*, [in] int32 *readerID*, [in] int32 *segmentID*, [in] datetime **UTCTime**);

Parameters:

- areaID - This is ID of the area to move the badge to.
- badgeID - This is the 64-bit badge ID of the badge you want to move.
- badgeID_str - A string representation of the badgeID. You cannot provide both badgeID and badgeID_str in the same call.
- 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 authenticated readers that specify the authentication mechanism used by the reader to authenticate a cardholder.
For pivCLASS authenticated readers: 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 294.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Name of the authentication mode.	View

Lnl_Badge

Description: A badge in the security system.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	BADGEKEY	Internal database ID. Key field.	View
datetime (string)	ACTIVATE	Badge activate date in the format `1968-07-31T12:34`. **Note:** Default for ACTIVATE is the current date and time.	Edit
boolean	APBEXEMPT	Whether the badge is APB exempt	Edit
string	COUNTRYCODE	The country code for the mobile phone number associated with this badge. For the Cumulus Credential Factory solution, this field is appended to the Mobile Phone Number sent to Cumulus when this badge is issued as a mobile credential. **Note:** The country code and mobile phone number must be in the ITU E.164 format. To support this format, a "+" is added to beginning of the country code automatically, and cannot be deleted. You should *not* include the "+" in your Lnl_Badge call.	Edit
datetime (string)	DEACTIVATE	Badge deactivate date in the format `1968-07-31T12:34`. **Note:** Default for DEACTIVATE is determined by the configuration for the badge type in System Administration.	Edit
boolean	DEADBOLT_OVERRIDE	If true, the selected cardholder will have deadbolt override privileges, which allows the cardholder to access a door with a deadbolt function mortise lock even when the deadbolt is thrown.	Edit

Type	Name	Description	Access
boolean	DEST_EXEMPT	If true, the badge will not be included in the destination assurance processing and no alarms will be generated if the cardholder violates any of the destination assurance settings.	Edit
int32	EMBOSSED	Embossed	Edit
boolean	EXTEND_STRIKE_HELD	Use extended strike/held times	Edit
int64	ID	ID of the badge.	Edit
string	ID_Str	A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off. **Note:** This property is only returned when **get instances** is called with Version 1.2 or later. **Note:** When adding or modifying an Lnl_Badge, you cannot provide both an ID and an ID_Str.	View
int32	ISSUECODE	Issue code. **Note:** Default for ISSUECODE is determined by the First Issue Code configured for the badge type in System Administration.	Edit
datetime (string)	LASTCHANGED	Badge last changed	View
datetime (string)	LASTPRINT	Badge last printed	View

Type	Name	Description	Access
string	MOBILEPHONENUMBER	The mobile phone number to be associated with this badge. For the Cumulus Credential Factory solution, this number (along with the Country Code) is sent to Cumulus when this badge is issued as a mobile credential. **Note:** The country code and mobile phone number must be in the ITU E.164 format. To support this format, a “+” is added to beginning of the country code automatically, and cannot be deleted. You should *not* include the “+” in your Lnl_Badge call.	Edit
boolean	PASSAGE_MODE	If true, the cardholder is allowed to use the card twice (within the lock's unlock duration) to place the lock in an unlock mode for an indefinite duration.	Edit
int32	PERSONID	Internal ID of the person who owns this badge. If you modify Lnl_Badge.PERSONID, all other properties are ignored. For example, if you modify Lnl_Badge.PERSONID and Lnl_Badge.STATUS in the same "PUT /instances" call, the Lnl_Badge.STATUS is ignored. Lnl_Badge.PERSONID cannot be modified if the class of the badge type is “Standard” or “Guest”.	Edit
string	PIN	PIN code. **Note:** You cannot view or search the contents of this property.	Edit
int32	PRINTS	Number of times badge has been printed	View
int32	STATUS	Badge status ID. 1 = “Active”. For more information, refer to User-Defined Value Lists on page 330.	Edit

Type	Name	Description	Access
int32	TYPE	Badge type ID. For more information, refer to Lnl_BadgeType on page 250.	Edit
int32	USELIMIT	Use limit	Edit

Notes: A successful response indicates that the badge and access level assignment have reached the database. The successful response does not indicate that the assignment has reached the access panel. There might be a delay before the assignment reaches the panel.

When adding badges of badge types associated with a Cumulus Universal bundle type, data about the cardholder and credential is sent to Cumulus at the time of badge creation using OpenAccess, and the mobile credential is issued at that time automatically. Therefore, a separate issue operation is not necessary. However, if the badge creation fails in Cumulus, you can use the **issue_mobile_credential** option to retry the operation.

When modifying badges of badge types associated with a Cumulus bundle using OpenAccess to change any attributes that are configured to be sent to Cumulus, a call is made to Cumulus to provide the updated data. For more information, refer to "Send to Cumulus Form" in the *System Administration User Guide*.

Methods:

- void AssignAccessLevel([in] level[] LevelIn);

  Assigns the access level(s) of a badge. The following table describes how OpenAccess uses cardholder permissions and Area Access Manager levels to determine which access levels the authenticated OpenAccess user who is making the call can assign.

Does authenticated OpenAccess user have permission group, badge, and "Modify Access Level Assignment" permissions?	Does authenticated OpenAccess user have Area Access Manager levels defined?	The authenticated OpenAccess user can assign these access levels
Yes	Yes	All
Yes	No	All
No	Yes	Only Area Access Manager access levels
No	No	None

Note: If the authenticated OpenAccess user only has Area Access Manager access levels defined, all access levels in the AssignLevel array must be contained within the authenticated OpenAccess user's Area Access Manager access levels. For example, if the authenticated OpenAccess user has access levels 1 and 2, then the authenticated OpenAccess user cannot assign access levels 1, 2, and 3, and the entire access level assignment attempt will fail.

Parameters:

LevelIn - Array that includes all the access level IDs the badge needs to be assigned with, in the format:

– `[1, 2, 3]`

Note: This method supports the optional **Operation Description** and **Operation GUID** parameters. For more information, refer to execute_method on page 111.

- void ReplaceAccessLevels([in] int32 SourceBadgekey);

  Replaces the access levels assigned to the badge instance with the access levels belonging to the badge with the supplied badge key. This method also copies the activation and deactivation dates for all access levels copied from the badge identified by the input badge key.

  If no input parameter is provided, this method removes all access level assignments of the badge. This is the recommended approach for deleting all access level assignments from a badge.

  Parameters:

  SourceBadgekey - The badgekey of the badge from which to copy the access levels.

Note: This method supports the optional **Operation Description** and **Operation GUID** parameters. For more information, refer to execute_method on page 111.

- void ModifyAccessLevelAssignment([in] level[] LevelIn);

  Allows you to add, modify, or remove access levels, including temporary access levels with activation and deactivation dates.

  If an input access level is provided with **add_level=true**, and that level already exists for the badge, the operation succeeds and the provided **activation_date** and **deactivation_date** , if any, replaces the existing values.

  If the request contains two or more levels with matching **access_level_id**, the operation fails.

  The **level** array consists of the parameters shown below, in the format
  `[{"access_level_id":1,"activation_date":"2022-10-21T20:22","deactivation_date":"2023-10-21T20:22"},{"access_level_id":2,"add_level":false}]`

Parameter	Type	Description
access_level_id	int32	The ID of the access level.
activation_date	datetime	The date and time when the access level will be activated, in the access panel's local time. For example: `2022-09-25T08:00`
deactivation_date	datetime	The date and time when the access level will be deactivated, in the access panel's local time. For example: `2022-10-25T17:00`
add_level	boolean	If true, the level will be added. If false, the level will be removed.

Notes: If the **activation_date** or **deactivation_date** properties include a timezone offset or seconds (for example, the "22-04:00" in "`2022-09-25T08:11:22-04:00`"), the offset and seconds are ignored, and the date/time is stored in the database without an offset or seconds (for the previous example, "`2022-09-25T08:11:00`" is stored in

the database). A warning message is returned that the timezone offset or the seconds were ignored.

This method supports the optional **Operation Description** and **Operation GUID** parameters. For more information, refer to execute_method on page 111.

- void ReplacePIN([in] int32 SourceBadgekey);

  Replaces the PIN assigned to the current badge instance with the PIN belonging to the badge with the supplied badgekey.

  Parameters:

  SourceBadgekey - The badgekey of the badge from which to copy the PIN.

Note: This method supports the optional **Operation Description** and **Operation GUID** parameters. For more information, refer to execute_method on page 111.

Lnl_BadgeFIPS201

Description: Holds the data imported from FIPS 201 credentials.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	BADGEKEY	Internal database ID of the associated badge record. Key field.	Read
string(hex)	FASCN	Federal Agency Smart Credential Number.	Edit
binary	TWICPrivacyKey	TWIC Privacy Key. The key used to encrypt/decrypt the fingerprints on TWICs.	Edit
int32	TPKAlgorithmId	TWIC Privacy Key algorithm identifier. The algorithm used for encrypting/decrypting the fingerprints on TWICs. Paired with the TWIC Privacy Key.	Edit
string(hex)	UUID	Cardholder's globally unique identifier.	Edit

Type	Name	Description	Access
int32	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	Edit

Lnl_BadgeLastLocation

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

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int64	BadgeID	Badge ID. Key field.	View
string	BadgeID_str	A string representation of the badge ID. To accurately display badge ID, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off. **Note:** This property is only returned when **get instances** is called with Version 1.2 or later.	View
int32	AccessFlag	Shows whether the access was granted. Key field.	View
int32	DatabaseID	The database identifier in an Enterprise system that identifies the system containing the reader to which the badge was last presented. Key field.	View
int32	PanelID	Panel ID where access event occurred. Reference to Lnl_Panel.ID.	View
int32	ReaderID	Reader ID at which access occurred	View

Type	Name	Description	Access
datetime (string)	EventTime	Time at which access occurred	View
int32	EventID	ID of the event associated with the access.	View
int32	EventType	Type of the event associated with access	View
int32	PersonID	Lnl_Person for which access occurred	View
int32	IsFromReplication	Shows whether badge last location came over for other region in the system.	View

Lnl_BadgeStatus

Description: The status of a badge in the security system.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	NAME	Name of the list value.	Edit

Lnl_BadgeType

Description: A badge type in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	NAME	Name of the badgetype.	View

Type	Name	Description	Access
int32	BadgeIDAllocationType	Indicates the method by which the Badge ID field on the Badge Form is automatically filled in when adding a new badge. 1: Automatic 2: From Cardholder ID 3: Manual entry 5: Internal Cardholder ID 7: FASC-N 8: Import from card	View
int32	BadgeTypeClass	Class of the badgetype Possible values: 0: Standard 1: Temporary 2: Visitor 3: Guest 4: Special Purpose	View
int32	DefaultAccessGroup	A group of access levels to be associated with this badge type.	View
string	DefaultDeactivationDate	Indicates the date on which badges of the specified type will expire.	View
int32	DefaultDeactivationDateType	Indicates the type, or class, assigned to this badge. 0: None 1: Exact 2: After	View
int32	FirstIssueCode	Indicates the first issue code, if used, for the badge (0 or user-specififed).	View
boolean	IsDisposable	If true, indicates that the visitor's badge will be a disposable badge.	View
int32	SegmentID	Segment to which the badge type belongs.	View
boolean	AnySegmentCanAssign	Returns true if badge type is made available to any user and any person (no segment restrictions).	View
boolean	BadgeIDAllowEdit	Returns true if badge type allows editing of the badge ID of this type.	View
boolean	UseLatestBadgeDeactivation Date	Indicates whether or not the latest deactivation date of existing badges is used.	View
boolean	UseMobileCredential	Indicates whether or not mobile credentialing is enabled.	View

Methods:

- void GetRequiredFields([out] string[] RequiredFields);

  Returns a list of field names that this badge type requires a cardholder to have in order to possess a badge of this type.

Lnl_Camera

Description: A camera defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	CameraID	Recorder-specific unique camera ID. **Note:** This property is only returned when **get instances** is called with Version 1.5 or later.	View
string	CameraTypeName	Camera Type Name	View
int32	Channel	LenelS2 NVR Channel	View
int32	FrameRate	Frame rate	View
int32	HorizontalResolution	Horizontal resolution	View
int32	ID	Internal database ID. Key field.	View
int32	IPAddress	IP address of the camera.	View
boolean	IsOnline	If true, the camera is online.	View
int32	MotionBitRate	Motion Bit Rate	View
string	Name	Camera name.	View
int32	NonMotionBitRate	Non-motion Bit Rate	View
int32	PanelID	LenelS2 NVR ID. Reference to Lnl_Panel.ID. Key field.	View
int32	PlaybackPreroll	Camera-specific preroll value. **Note:** This property is only returned when **get instances** is called with Version 1.7 or later.	View

Type	Name	Description	Access
int32	PlaybackPostroll	Camera-specific postroll value. **Note:** This property is only returned when **get instances** is called with Version 1.7 or later.	View
int32	Port	Port of the camera.	View
string	RecorderName	Name of recorder to which camera is connected.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	VerticalResolution	Vertical Resolution	View
string	VideoStandard	Video Standard (Ex.: NTSC).	View
string	Workstation	Workstation of the host LenelS2 NVR.	View

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.

Note: This method supports the optional **Operation Description** and **Operation GUID** parameters. For more information, refer to execute_method on page 111.

Lnl_CameraDeviceLink

Description: Shows the relationship between a camera and a device (such as a reader). Used for determining if event video is available for the specified device.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	CameraID	The ID of the camera.	View
int32	DeviceID	The ID of the device.	View

Type	Name	Description	Access
int32	DevicePanelID	The ID of the panel to which the device is associated.	View
int32	InputOutputID	The ID of the input or output for this association, if any.	View
int32	VideoRecorderID	The ID of the video recorder to which the camera is associated.	View
int32	ViewOrder	The order, or priority, to be used by clients when displaying video associated with an event, if there are multiple cameras associated with a single device.	View

Lnl_CameraGroup

Description: Camera group definition.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Group name.	View
int32	SegmentID	Segment to which the camera group belongs.	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
int32	CameraGroupID	Camera group for this link. Lnl_CameraGroup.ID. Key field.	View
int32	PanelID	Panel ID for the camera. Reference to Lnl_Panel.ID. Key field.	View
int32	CameraID	Camera ID. Key field. See Lnl_Camera.ID.	View

Lnl_Cardholder

Description: A cardholder in the security system.

Abstract: No

Access: 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	Edit
string	ADDR1	The cardholder's address.	Edit
datetime (string)	BDATE	The cardholder's birth date in the format `1968-07-31`.	Edit
int32	BUILDING	Reference to Lnl_BUILDING. For more information, refer to User-Defined Value Lists on page 330.	Edit
string	CITY	The cardholder's city.	Edit
int32	DEPT	Reference to Lnl_DEPT. For more information, refer to User-Defined Value Lists on page 330.	Edit
int32	DIVISION	Reference to Lnl_DIVISION. For more information, refer to User-Defined Value Lists on page 330.	Edit

Type	Name	Description	Access
string	EMAIL	The cardholder's email address.	Edit
string	EXT	The cardholder's extension.	Edit
string	FLOOR	The cardholder's floor.	Edit
int32	LOCATION	Reference to Lnl_LOCATION. For more information, refer to User-Defined Value Lists on page 330.	Edit
string	OPHONE	The cardholder's office phone number.	Edit
string	PHONE	The cardholder's phone number.	Edit
int32	PRIMARYSEGMENTID	This property is only visible when cardholders are segmented.	Read
string	SSNO	Person's identification number.	Edit
string	STATE	The cardholder's state.	Edit
int32	TITLE	Reference to Lnl_TITLE. For more information, refer to User-Defined Value Lists on page 330.	Edit
string	ZIP	The cardholder's zip code.	Edit

Note: When using OpenAccess to modify a cardholder with a badge of a badge type associated with a Cumulus bundle to change any attributes that are configured to be sent to Cumulus, a call is made to Cumulus to provide the updated data. For more information, refer to "Send to Cumulus Form" in the *System Administration User Guide*.

Lnl_DeviceGroup

Description: A group consisting of one or more readers, inputs, outputs, cameras, or remote monitoring devices. A group can contain devices from more than one access panel, and a device can belong to more than one group. In a segmented system, a device group can belong either to one segment or to all segments.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Methods:

void LockdownGroupActivate();

Activate the lockdown group with the specified ID, in the format
`{"property_value_map": {"ID":7}}`

void LockdownGroupDeactivate();

Deactivate the lockdown group with the specified ID, in the format
`{"property_value_map": {"ID":6}}`

void GetLockdownGroupState();

Get the current state of the lockdown group with the specified ID, in the format
`{"property_value_map": {"ID":4}}`

An example response is:

```
{
"LockdownState": 2,
"TotalReaders": 5,
"LockedReaders": 2,
"UnlockedReaders": 1,
"FailedReaders": 2
}
```

Possible **LockdownState** returned values are:

- 0: Deactivated
- 1: Activating
- 2: Activated
- 3: Deactivating

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	The name of the device group.	View
int32	SegmentID	The ID of the segment to which the device group belongs (when segmentation is enabled).	View
int32	Type	The type of device group: 0: Reader Group 1: Input Group 2: Output Group 3: Camera Group 4: Monitor Group 5: Lockdown Group	View

Lnl_Directory

Description: A directory defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	ID	Internal database ID. Key field.	View
string	ACCOUNTCATEGORY	Account category.	View
string	ACCOUNTCLASS	Account class.	View
string	ACCOUNTDISPLAYNAMEATTR	Account display name attribute.	View
string	ACCOUNTIDATTR	Account ID attribute.	View
string	ACCOUNTUSERNAMEATTR	Account user name 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: Windows Local Accounts 4: OpenID Connect	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: None

Superclass: None

Platforms: OnGuard

Properties: None

Lnl_ElevatorTerminal

Description: An elevator terminal defined in the security system. Retrieves the hardware status for the device.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	IPAddress	The IP address of the associated elevator terminal. Key field	View
int32	PanelId	Reference to Lnl_Panel.ID. Key field.	View
string	Hostname	Host name or domain.	View
int32	Name	The name of the associated elevator terminal.	View

Methods:

void GetHardwareStatus();

Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

Possible returned values are:

- 1 = default floor only
- 2 = Access to authorized floors
- 3 = User entry of destination floor
- 4 = Default floor or user entry of destination floor

void SetAllowedFloors();

Sends a command to update which floors and doors are accessible via the elevator terminal without supplying security credentials. This method takes a single parameter named AllowedFloorListID which corresponds to a Floor List in the OnGuard software. Returns Pass or Fail.

void SetTerminalMode();

Sends a command to update the elevator terminal's operational mode for interacting with the cardholder. This method takes the numerical value of a single parameter named Mode. Possible values are:

- 1 = Default floor only. When the cardholder presents a valid badge to the elevator reader, or enters a valid PIN code or floor number on the elevator terminal, the system calls the default floor.

- 2 = Access to authorized floors. When the cardholder presents a valid badge to the elevator reader, and then selects an authorized floor, the system calls the authorized floor.
- 3 = User entry of destination floor. The cardholder has the option to select a floor with or without presenting a valid badge to the elevator reader. If the selected floor is an allowed floor, the system calls the floor. If the floor is a non-allowed floor, the cardholder is requested to present a valid badge.
- 4 = Default floor or user entry of destination floor. When the cardholder presents a valid badge to the elevator reader, the system calls the cardholder's default floor. Within a configurable timeout period, the cardholder can override the default floor call by entering another floor number.

Lnl_EventAlarmDefinitionLink

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

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	DeviceID	Device ID (ex.: Alarm panel, Reader). Key field.	View
int32	EventParameterID	Event parameter ID. Key field. See Lnl_EventParameter.ID.	View
int32	EventSubtypeDefinitionID	Event Subtype. Key field. See Lnl_EventSubtypeDefinition.ID.	View
string	EventText	Text used to filter events with matching **associated_text** while mapping events to alarms. This parameter is compared to the first row of the text associated with the event. For more information, refer to get logged_events on page 82. **Note:** This property is only returned when **get instances** is called with version 1.7 or later.	View
int32	EventTypeID	Event Type. Key field. See Lnl_EventType.ID.	View
int32	PanelID	Panel ID (ex.: ISC). Key field. Reference to Lnl_Panel.ID.	View
int32	SecondaryDeviceID	Secondary device ID (ex.: Input, Output). Key field.	View

Type	Name	Description	Access
int32	AlarmDefinitionID	Alarm Definition. See Lnl_AlarmDefinition SubtypeID.	View

Lnl_EventParameter

Description: An event parameter.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Description	Parameter description.	View
int32	Value	Parameter value	View

Lnl_EventSubtypeDefinition

Description: An event subtype defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	TypeID	Event Type ID, see Lnl_EventType.ID.	View
int32	SubTypeID	ID within the subtype.	View
string	Description	Sub type description.	View
int32	SupportParameters	Supporting Parameter ID	View
int32	Category	Event subtype category	View

Lnl_EventSubtypeParameterLink

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

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	EventParameterID	Key field. See Lnl_EventParameter.ID.	View
int32	EventSubtypeDefinitionID	Key field. See Lnl_EventSubtypeDefinition.ID.	View

Lnl_EventType

Description: An event type defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Description	Event type description.	View

Lnl_GuardTour

Description: A guard tour provides a security guard with a defined set of tasks that must be performed within a specified period of time.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View

Type	Name	Description	Access
string	Name	Guard tour name.	View

Methods:

- **void** LaunchTour([in] **int64** BadgeId, [in] **string** badgeID_str, [in] **int32** MonStationId, [out] **int32** ReturnValue);

  Parameters:

  - BadgeId - This is the 64-bit badge ID of the badge you want to move.
  - badgeID_str - A string representation of the badgeID. You cannot provide both badgeID and badgeID_str in the same call.
  - MonStationID - Monitoring station (workstation) ID
  - ReturnValue - Result of the guard tour. Possible values:
    - 0: Success
    - 1: Tour already in progress
    - 2: Tour not in progress
    - 3: Invalid tour ID
    - 4: Invalid tour status
    - 5: Invalid badge ID
    - 6: Invalid monitoring station
    - 7: Communication error

Lnl_Holiday

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

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Note: **Add** and **Modify** permissions are available when **post instances** is called with version 1.4 and later. **Delete** permission is available when **delete instances** is called with version 1.2 and later. Starting with version 1.9 of **get instances** and version 1.4 of **put/post instances**, the time format was changed from `YYYY-MM-DDTHH:MM:SS+00:00` format to `YYYY-MM-DD`.

Properties:

Type	Name	Description	Access
int32	ExtentDays	How many days the holiday lasts. Allowed values are 0 through 127.	Edit
int32	ID	Internal database ID. Key field.	View
string	Name	Holiday name. Maximum length is 64 characters.	Edit

Type	Name	Description	Access
boolean	RepeatYearly	If true, the holiday repeats every year. If false, the holiday does not repeat. This property is only returned when `get instances` is called with Version 1.9 or later.	Edit
int32	SegmentID	Segment to which the holiday belongs.	View
string	SegmentName	Name of the segment to which the holiday is assigned. This property is only returned when `get instances` is called with Version 1.9 or later.	View
datetime (string)	StartDate	Date the holiday starts, in the format `YYYY-MM-DD`.	Edit

Lnl_HolidayType

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

Abstract: No

Access: View/Modify

Superclass: Lnl_Element

Platforms: OnGuard

Note: **Modify** permissions are available when **post instances** is called with version 1.4 or later.

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Holiday name. Maximum length is 64 characters.	Edit
int32	SegmentID	Segment to which the holiday belongs.	View

Lnl_HolidayTypeLink

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

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Note: **Add** permissions are available when **post instances** is called with version 1.4 or later. **Delete** permission is available when **delete instances** is called with version 1.2 or later.

Properties:

Type	Name	Description	Access
int32	HolidayID	Holiday. Key field.	Read
int32	HolidayTypeID	Holiday type. Key field. Allowed values are 1 through 8.	Read

Lnl_IncomingEvent

Description: A data class that supports sending incoming events via OpenAccess. This object has no properties; it only has the methods listed below.

Abstract: No

Superclass: Lnl_Element

Platforms: OnGuard

Properties: None

Methods:

- **void** SendIncomingEvent([in] **string** Source, [in] **string** Device, [in] **string** SubDevice, [in] **string** Description, [in] **datetime** Time, [in] **boolean** IsAccessGrant, [in] **boolean** IsAccessDeny, [in] **int64** BadgeID, [in] **string** BadgeID_str, [in] **string (hex)** ExtendedID);

  Parameters:

  - Source - text representation of the object/device that generated the event
    Variable-length Unicode string. This parameter is required. The source must be defined in the OpenAccess Sources folder (in the System Administration application) prior to using the Lnl_IncomingEvent::SendIncomingEvent method. For more information, refer to Add a Logical Source on page 342.
  - Device - text representation of a device associated with a OpenAccess Source that generated the event
    Variable-length Unicode string. This parameter is optional. The device must be defined in the OpenAccess Sources folder > OpenAccess Devices tab (in System Administration) prior to using the Lnl_IncomingEvent::SendIncomingEvent method.
  - SubDevice - text representation of a sub device associated with a OpenAccess Device that generated the event.
    Variable-length Unicode string. This parameter is optional. The device must be defined in the OpenAccess Sources folder > OpenAccess Sub-Devices tab (in System Administration) prior to using the Lnl_IncomingEvent::SendIncomingEvent method.
  - Description - text that describes the event
    Variable-length Unicode string.
  - 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 OpenAccess 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 OpenAccess

Source, Device or Sub-Device must be defined in the OpenAccess Sources folder > OpenAccess 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 269.

- IsAccessDeny - boolean value that specifies whether the event reported for the OpenAccess 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 OpenAccess Source, Device or SubDevice must be defined in the OpenAccess Sources folder > OpenAccess 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 269.
- BadgeID - The 64-bit badge ID of the badge in the OnGuard database that generated the event. This parameter is optional and is used in association with all badge related events.
- BadgeID_str - A string representation of the BadgeID. You cannot provide both BadgeID and BadgeID_str in the same call.

Note: The maximum character limit for both the BadgeID and BadgeID_str fields is 18 characters.

- ExtendedID - Extended length string identifier that refers to a PIV-based badge in the OnGuard database that generated the event. Specifies the 128-bit UUID or 200-bit FASC-N. This parameter is optional and is used in association with all badge-related events. This parameter must be in hexadecimal string format. The FASCN or UUID needs to be converted to a binary value that begins with "0x" and includes the values of the FASCN/UUID.

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

- **int32** AcknowledgeAlarm([in] **int32** CurrentAckStatus, [in] **int32** SerialNumber, [in] **string** CommServerHostName, [in] **int32** PanelID, [in] **int32** AlarmID, [in] **datetime** AlarmTime, [in] **int32** AckStatus, [in] **string** AckNotes, [object] AckUdfItems, [out] **int32** SimultaneousAckStatus);

Note: For Version 1.2 or later, the request will be rejected if this method is called for an alarm that has **Require login on acknowledge** set to **true**.

Description:

Allows acknowledgement of alarms received from the system. Most of the parameters can be extracted from the Lnl_LoggedEvent.

Return:

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

1 - If acknowledgement succeeds.

Parameters:

- CurrentAckStatus - Current acknowledgement status of the alarm to ensure that simultaneous acknowledgement 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 acknowledgement request.
- AlarmID - Alarm ID associated with the event to ensure the integrity of the acknowledgement request.
- AlarmTime - Time the event occurred to ensure the integrity of the acknowledgement request.
- AckStatus - Acknowledgment status to set. See the CurrentAckStatus parameter description for possible values.
- AckNotes - Acknowledgment notes to set. AckStatus must be 2.
- AckUdfItems - Parameter for UDF fields. User can put the UDF fields and their values into this parameter, in the format:

```
{
"property_value_map": {},
"in_parameter_value_map":
{"SerialNumber":1667439811,"PanelID":1,"CurrentAckStatus":3,"
AckStatus":3,"AckNotes":"abc","AckUdfItems":
   {"TXT01":"aaa", "TXT02":"bbb"}
}
}
```

  In this example, **TXT01** and **TXT02** are the UDF names configured in FormsDesigner. You can get the UDF names and their types using **Lnl_AlarmAckHistory**.
- SimultaneousAckStatus - Value greater than 0 if alarm had been acknowledged by other means. Contains the new acknowledgement status if that was the case. Refer to the CurrentAckStatus parameter description for possible values.

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

- **int32** AcknowledgeAlarmWithAuth([in] **int32** CurrentAckStatus, [in] **int32** SerialNumber, [in] **string** CommServerHostName, [in] **int32** PanelID, [in] **int32** AlarmID, [in] **datetime** AlarmTime, [in] **int32** AckStatus, [in] **string** AckNotes, [in] **int32** DirectoryID, [in] **string** UserName, [in] **string** Password, [object] AckUdfItems, [out] **int32** SimultaneousAckStatus;

  Description:

  Allows acknowledgement of alarms received from the system. Most of the parameters can be extracted from the Lnl_LoggedEvent.

  If this method is called for an alarm with **Require login on acknowledge** set to **true**, then the credentials are validated and:

  - If the credentials are valid:
    - The provided credentials are used to perform the acknowledgement operation. Any transaction logging will show the provided username as the user who performed the acknowledgement operation.
    - If the provided credentials do not have sufficient permissions to perform the requested operation, the request is rejected with an appropriate error message.

    - A new OpenAccess session is **not** established. The credentials are used only for this single operation.
- If the credentials are not valid, the operation is rejected with an error message.

If the **UserName** and **Password** input parameters are not provided, the request is rejected with an error message.

If this method is called for an alarm with **Require login on acknowledge** set to **false**, then the request is rejected.

Return:

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

1 - If acknowledgement succeeds.

Parameters:

- CurrentAckStatus - current acknowledgement status of the alarm to ensure that simultaneous acknowledgement 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 acknowledgement request
- AlarmID - Event type ID associated with the event to ensure the integrity of the acknowledgement request
- AlarmTime - Time the event occurred to ensure the integrity of the acknowledgement 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 acknowledgement status if that was the case. See the CurrentAckStatus parameter description for possible values.
- AckUdfItems - Parameter for UDF fields. User can put the UDF fields and their values into this parameter, in the format:

  ```
  {
  "property_value_map": {},
  "in_parameter_value_map":
  {"SerialNumber":1667439811,"PanelID":1,"CurrentAckStatus":3,"
  AckStatus":3,"AckNotes":"abc","AckUdfItems":
     {"TXT01":"aaa", "TXT02":"bbb"}
  }
  }
  ```

  In this example, **TXT01** and **TXT02** are the UDF names configured in FormsDesigner. You can get the UDF names and their types using **Lnl_AlarmAckHistory**.

- UserName - The user's OnGuard username.
- Password - The user's OnGuard password.

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

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

Note: The OpenAccess 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.

Lnl_Input

Description: Abstract class that represents any kind of input.

Abstract: Yes

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the input's panel is running.	View
string	Name	The name of the input.	View
int32	PanelId	The ID of the associated access panel. Reference to Lnl_Panel.ID.	View

Lnl_IntrusionArea

Description: Implements the control methods for the Intrusion Area. Retrieves the hardware status for the device.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	AreaNumber	The number of the associated intrusion area.	View
int32	AreaType	The type of the associated intrusion area.	View
string	HostName	The name of the workstation where the Communication Server associated with the intrusion panel is running.	View
string	Name	The name of the associated intrusion area.	View
int32	PanelId	The ID of the associated intrusion panel. Reference to Lnl_Panel.ID.	View

Methods:

void Arm([in] int32 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)

Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:	
OFFLINE_STATUS	0
ONLINE_STATUS	1

Lnl_IntrusionDoor

Description: Implements the control methods for the Intrusion Door.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	DeviceId	The ID of the intrusion door. Key field.	View
int32	ID	Internal database ID. Key field.	View
int32	PanelId	The ID of the associated intrusion panel. Key field. Reference to Lnl_Panel.ID.	View
string	HostName	The name of the workstation where the Communication Server associated with the intrusion panel is running.	View
string	Name	The name of the associated intrusion door.	View

Methods:

void Open()

Sends a command to open the intrusion door.

void SetMode([in] int32 Mode);

Sends a command to change the intrusion door mode.

Parameters:

int32 Mode: Intrusion door mode to be set. Allowed values are:	
INTRUSION_DOOR_MODE_LOCKED	1
INTRUSION_DOOR_MODE_UNLOCKED	2
INTRUSION_DOOR_MODE_SECURED	3
INTRUSION_DOOR_MODE_ENABLED	4
INTRUSION_DOOR_MODE_OFFLINE	5

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 – door status:		
uint32 Status	**Description**	**Device status**
INTRUSION_DOOR_BIT_LEARN_MODE	Door bit learn mode	1
INTRUSION_DOOR_BIT_DIAGNOSTIC_MODE	Door bit diagnostic mode	2

uint32 Status – door status:		
uint32 Status	**Description**	**Device status**
INTRUSION_DOOR_BIT_NOT_INSTALLED	Door bit not installed	4
INTRUSION_DOOR_BIT_SDI_FAILURE	Door bit SDI failure	8
INTRUSION_DOOR_BIT_HELD	Door bit held	10
INTRUSION_DOOR_BIT_FORCED_OPEN	Door bit forced open	20
INTRUSION_DOOR_BIT_UNBLOCKED	Door bit unblocked	40

Lnl_IntrusionOutput

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

Abstract: Yes

Access: View

Superclass: Lnl_Output

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	DeviceId	The ID of the intrusion output. Key field.	View
int32	PanelId	The ID of the associated intrusion panel. Key field. Reference to Lnl_Panel.ID.	View
string	HostName	The name of the workstation where the Communication Server associated with the intrusion panel is running.	View
string	Name	The name of the intrusion output.	View

Lnl_IntrusionZone

Description: Implements the control methods for the Intrusion Zone.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	DeviceID	The ID of the intrusion zone. Key field.	View
int32	ID	Internal database ID. Key field.	View
int32	PanelID	The ID of the associated intrusion panel. Key field. Reference to Lnl_Panel.ID.	View
string	HostName	The name of the workstation where the Communication Server associated with the intrusion panel is running.	View
string	Name	The name of the associated intrusion zone.	View

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	0
ONLINE_STATUS	1

Lnl_LoggedEvent

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

Notes: The primary way of obtaining hardware events is through subscribing to the event, not pulling the event. Only use this call when you have no other choice because the system has gone out of synchronization.

When requesting events using the **get logged_events** call, a filter is required due to the large number of instances this class usually returns. Also, be careful what you specify as the order_by value. If left blank, the key values (PanelID, SerialNumber) are used, which works well.

You can also specify **Timestamp** as the **order_by** value. If you filter by Time, you will improve performance if you also order_by Time. Filters can be used on other columns as long as there is an index. However, it is not recommended to use any other

combination without an index in place on the EVENTS table, as doing so might generate a timeout error. For more information, refer to Error Messages on page 349.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	SerialNumber	Serial number of the event. Key field.	View
int32	PanelID	Panel at which the event occurred. Key field. Reference to Lnl_Panel.ID.	View
datetime (string)	Time	Time when event occurred.	View
string	Description	Description of the event.	View
int32	DeviceID	Device ID at which event occurred (Lnl_Reader, Lnl_AlarmPanel, and so on.)	View
string(hex)	ExtendedID	Extended identifier of the card (where available) which caused the event.	View
int32	SecondaryDeviceID	Secondary device ID at which event occurred (ex. Lnl_Input).	View
int32	SegmentID	Segment where event occurred.	View
int32	Type	Event type i.e., “duress”, “system”, etc. Corresponds to Lnl_EventSubtypeDefinition.TypeID and Lnl_EventType.ID.	View
int32	SubType	Event sub-type i.e., “granted”, “door forced open”, etc. Corresponds to Lnl_EventSubtypeDefinition. SubTypeID.	View
string	EventText	Text associated with event.	View
int64	CardNumber	Card (where available) which caused the event.	View

Type	Name	Description	Access
string	CardNumber_str	A string representation of the Card Number. To accurately display Card Number, web clients should use this property instead of the **ID** property, since there is a JavaScript limitation in which integer values with 17 digits or more are rounded off. **Note:** This property is only returned when **get instances** is called with Version 1.2 or later.	View
int32	IssueCode	Issue code of the card.	View
int32	AssetID	Asset (where available) which caused the event.	View
int32	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 **Note:** You cannot filter using the AccessResult property. For more information, refer to Searching for Objects on page 43.	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
int32	PersonID	Internal ID of the person who is assigned the badge at the time of the access event. See Lnl_Person.ID.	View
int32	Priority	Alarm priority (0 to 255).	View
int32	PriorityColorRed-Value	The red component of the RGB color for the alarm (0 to 255).	View
int32	PriorityColorGreen-Value	The green component of the RGB color for the alarm after it is acknowledged (0 to 255).	View

Type	Name	Description	Access
int32	PriorityColorBlue-Value	The blue component of the RGB color for the alarm (0 to 255).	View
int32	PriorityColorAckRed-Value	The red component of the RGB color for the alarm after it is acknowledged (0 to 255).	View
int32	PriorityColorAck-GreenValue	The green component of the RGB color for the alarm after it is acknowledged (0 to 255).	View
int32	PriorityColorAck-BlueValue	The blue component of the RGB color for the alarm after it is acknowledged (0 to 255).	View

Lnl_LogicalDevice

Description: A third-party logical device.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	SourceID	ID of the logical source to which this logical device belongs (Lnl_LogicalSource.ID). Key field.	Read
string	Name	Name of the logical device	Edit

Lnl_LogicalSource

Description: A third-party logical source.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View

Type	Name	Description	Access
boolean	IsDaylightSaving	Identifies if the logical source follows Daylight Saving Time rules. True = Follows Daylight Saving Time rules	Edit
boolean	IsOnline	Identifies if the logical source is online. True = Is online	Edit
string	Name	Name of the logical source.	Edit
int32	SegmentID	Segment to which the logical source belongs.	Read
int32	WorldTimezoneID	Reference to Lnl_WorldTimezone.ID	Edit

Lnl_LogicalSubDevice

Description: A third-party logical sub-device.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	DeviceID	ID of the logical device to which this logical sub-device belongs (Lnl_LogicalDevice.ID). Key field.	Read
int32	ID	Internal database ID. Key field.	View
int32	SourceID	Reference to Lnl_LogicalSource.ID. Key field.	Read
string	Name	Name of the logical sub-device.	Edit

Lnl_MonitoringZone

Description: A Monitoring zone defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Monitoring zone name.	View
int32	DefaultMapID	Map ID assigned as the default map to the monitoring zone. Set to 0 when no map is assigned.	View
int32	SegmentID	Segment to which the monitoring zone belongs.	View

Lnl_MonitoringZoneCameraLink

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

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Type	Name	Description	Access
int32	CameraID	Camera ID. Key field. See Lnl_Camera.ID.	View
int32	MonitoringZoneID	Monitoring Zone ID. Key field. See Lnl_MonitoringZone.ID.	View
int32	PanelID	Panel ID for the camera. Key field. Reference to Lnl_Panel.ID.	View

Lnl_MonitoringZoneDeviceLink

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

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Type	Name	Description	Access
int32	MonitoringZoneID	Monitoring Zone ID. Key field. Required field. See Lnl_MonitoringZone.ID.	Read

Type	Name	Description	Access
int32	PanelID	Panel ID for the device. Key field. Required field. Reference to Lnl_Panel.ID.	Read
int32	DeviceID	Device ID. Key field. Set to 0 when the device being linked to a monitoring zone is a panel type. Required.	Read
int32	InputOutputID	Input or output ID. Key field. Set to 0 when the device being linked to a monitoring zone is not an input or output. Required.	Read
boolean	AllDevicesOnPanel	Required. True if all sub-devices are included in this monitoring zone. False if individual sub-devices are to be specified. Required.	Read

Lnl_MonitoringZoneRecorderLink

Description: Defines what LenelS2 NVR Video Recorders are associated with a given monitoring zone.

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_MonitoringZoneDeviceLink

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	MonitoringZoneID	Monitoring Zone ID. Key field. Required field. See Lnl_MonitoringZone.ID.	Read
int32	PanelID	Panel ID for the device. Key field. Required field. Reference to Lnl_Panel.ID.	Read
int32	DeviceID	Device ID. Key field. Set to 0 when the recorder itself is being linked to the monitoring zone and not one of its sub-devices. Required.	Read
int32	InputOutputID	Input or Output ID. Key field. Set to 0 when the device being linked to a monitoring zone is not an input or output. Required.	Read

Type	Name	Description	Access
string	Name	The name of the panel.	View
boolean	AllDevicesOnPanel	Required. True if all sub-devices are included in this monitoring zone. False if individual sub-devices are to be specified.	Read

Lnl_MultimediaObject

Description: An image, signature, document, or biometric template belonging to a person in the security system.

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	AcceptanceThreshold	Biometric verification acceptance threshold. Possible values: • 0 = No security • 1 = Very high • 2 = High • 3 = Medium • 4 = Low • 5 = Very low • 6 = Default to global **Note:** This property is available in **get instances** version 1.9 and later, and in **put instances** and **post instances** version 1.4 and later.	Edit

Type	Name	Description	Access
int32	BiometricBodyPart	Type of biometric body part. Possible values: • 0 = None • 1 = Finger right index • 2 = Finger left index • 3 = Finger right middle • 4 = Finger left middle • 5 = Finger right ring • 6 = Finger left ring • 7 = Finger right little • 8 = Finger left little • 9 = Finger right thumb • 10 = Finger left thumb • 11 = Finger unnamed • 100 = Hand right • 101 = Hand right inverted • 102 = Hand left • 103 = Hand left inverted • 200 = Iris right • 201 = Iris left • 202 = Iris right left **Note:** This property is available in **get instances** version 1.9 and later, and in **put instances** and **post instances** version 1.4 and later.	Edit
sint32	DataType	Data type. Key field. For possible values, refer to DataType and ObjectType Pairings on page 283.	Read
sint32	ObjectType	Object type. Key field. For possible values, refer to DataType and ObjectType Pairings on page 283.	Read
sint32	PersonId	Internal ID of the person who owns this object. See Lnl_Person.ID.	Read
binary	Data	Array of image data.	Read
datetime (string)	LastChanged	Image last changed.	View

Notes: DataType and ObjectType properties must remain paired as shown in DataType and ObjectType Pairings on page 283.

The maximum file size that Lnl_MultimediaObject can send by default is less than 5 MB. To allow for larger file sizes, add the **client_max_body** line to the http block in the **nginx.conf** file (located by default in **C:\ProgramData\Lnl\nginx\conf**):

```
http {
     ...
     client_max_body_size 10M;
}
```

This example uses 10 MB as the maximum file size, but you can use any value that you feel is appropriate.

DataType and ObjectType Pairings

Multimedia Object Type	DataType	ObjectType
Photo Image	0	1
Photo Image Mask	1	1
Thumbnail	2	1
Signature	0	8
Hand Geometry (RSI)	4	16
LG Iris Code (right eye)	6	64
LG Iris Code (left eye)	7	64
LG Iris Image (right eye)	8	64
LG Iris Image (left eye)	9	64
Bioscrypt Fingerprint Template (primary)	3	32
Bioscrypt Fingerprint Template (secondary)	3	96
Bioscrypt Fingerprint Image (primary)	0	32
Bioscrypt Fingerprint Image (secondary)	0	96
ANSI INCITS 378 Template (primary)	11	112
ANSI INCITS 378 Template (secondary)	12	112
PK_COMP Template (primary)	11	128
PK_COMP Template (secondary)	12	128
Biometric PIN	-1	512
Visitor PDF Document	13	513

Lnl_OffBoardRelay

Description: Inherits from Lnl_Output, and therefore has the same properties. Implements the relay control methods and represents an Off-Board relay connected to the Intrusion Panel. Retrieves the hardware status for the device.

Abstract: No

Access: View

Superclass: Lnl_IntrusionOutput

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	DeviceId	The ID of the intrusion output. Key field.	View
int32	PanelId	The ID of the associated intrusion panel. Key field. Reference to Lnl_Panel.ID.	View
string	HostName	The name of the workstation where the Communication Server associated with the intrusion panel is running.	View
string	Name	The name of the intrusion output.	View

Methods:

void Activate()

Sends a command to activate a specific alarm relay.

void Deactivate()

Sends a command to deactivate a specific alarm relay.

void Toggle();

Toggles the state of the specific alarm relay.

void GetHardwareStatus([out] uint32 Status)

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_OnBoardRelay

Description: Inherits from Lnl_Output, and therefore has the same properties. Implements the relay control methods and represents an On-Board relay of the Intrusion Panel. Retrieves the hardware status for the device.

Abstract: No

Access: View

Superclass: Lnl_IntrusionOutput

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	DeviceId	The ID of the on-board relay. Key field.	View
int32	PanelId	The ID of the associated intrusion panel. Key field. Reference to Lnl_Panel.ID.	View
string	HostName	The name of the workstation where the Communication Server associated with the intrusion panel is running.	View
string	Name	The name.	View

Methods:

void Activate()

Sends a command to activate a specific alarm relay.

void Deactivate()

Sends a command to deactivate a specific alarm relay.

void GetHardwareStatus([out] uint32 Status)

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_Output

Description: Abstract class that represents any kind of output.

Abstract: Yes

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	PanelId	The ID number of the associated access panel. Reference to Lnl_Panel.ID. Key field.	View
string	HostName	The name of the workstation where the Communication Server associated with the output's panel is running.	View
string	Name	The name of the associated output.	View

Lnl_Panel

Description: A panel defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	ComputerName	The hostname of the panel, for panels that are specified using a hostname rather than an IP address.	View
int32	ControllerFlags	Information flags: • TLS encryption enabled = 0x08 • Peer encryption enabled = 0x40 **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	ID	Internal database ID. Key field.	View
boolean	IsDaylightSaving	Whether or not this panel observes Daylight Saving Time	View

Type	Name	Description	Access
boolean	IsOnline	The panel is online.	View
string	NAME	Display name.	View
int32	PanelClass	The class of the panel: 1 = access control panel 2 = fire panel 3 = intercom panel 4 = CCTV (video recorder) 5 = offline lock 6 = personal safety device 7 = intrusion panel 8 = switcher 9 = PC 10 = receiver 11 = account 12 = SNMP manager 13 = SNMP agent 14 = OPC connection 15 = OpenIT source 17 = point of sale device 18 = Intelligent Video Server 19 = video monitor 20 = elevator dispatching device 21 = Video Aux Server 22 = caching status proxy	View
string	PANELTYPE	Panel type name.	View
int32	PanelTypeID	The type of panel. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	PeerCertIssuedBy	Peer certificate information. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Type	Name	Description	Access
string	PeerCertIssuedExpired	The expiration date of the certificate. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	PeerCertIssuedStart	The start date of the certificate. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	PeerCertIssuedTo	Peer certificate information. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	PrimaryDialupHost-Number	The primary phone number to use when connecting to a server with dial-up access.	View
int32	PrimaryIPAddress	The primary IP address to use when connecting to a server with network access.	View
string	SecondaryDialupHost-Number	The back-up phone number to use when connecting to a server with dial-up access.	View
int32	SEGMENTID	Segment to which the panel belongs.	View

Type	Name	Description	Access
string	TLSCertIssuedBy	TLS certificate information. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	TLSCertIssuedExpired	The expiration date of the certificate. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	TLSCertIssuedStart	The start date of the certificate. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	TLSCertIssuedTo	TLS certificate information. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	WorldTimezoneID	Time zone of the panel (reference to Lnl_WorldTimezone.ID)	View
string	WORKSTATION	Panel workstation name.	View

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 DownloadPanelCertificates()

Sends a command to load the TLS and/or Peer certificate to the control panel. Also allows the enabling and disabling of the certificate attributes. The control panel will restart at the last step to apply the changes.

DownloadPanelCertificates		
Parameter	**Description**	**Type**
ID	The control panel's ID.	int32
TlsCertPath	The full path of the source certificate file on the local communication server or network file system. If the TLS paths (TlsCertPath and TlsKeyPath) are empty, then the TLS certificate loading step is skipped.	string
TlsKeyPath	The full path of the source key file on the local communication server or network file system. Required if the TlsCertPath is not empty.	string
TlsEnabled	Boolean flag to enable or disable the certificate. Can be used alone.	boolean
PeerCertPath	The full path of the source certificate file on the local communication server or network file system. If empty, the Peer certificate loading step is skipped.	string
PeerIssuedTo	The string using to validate the peer certificate.	string
PeerEnabled	The flag to enable or disable the peer certificate. Can be used alone.	boolean

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 UpdateHardwareStatus is called on the parent ISC. If the device is offline, the status is returned with a value of "0".

uint32 Status – device status:		
uint32 Status	**Description**	**Device status**
ONLINE_STATUS	Online	1
OPTIONS_MISMATCH_STATUS	Options Mismatch	2
CABINET_TAMPER	Cabinet Tamper	4
POWER_FAIL	Power Failure	8
DOWNLOADING_FIRMWARE	Downloading Firmware	10

Lnl_Person

Description: A cardholder or visitor in the security system.

Abstract: Yes

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Note: The properties listed below with Edit access are editable only through instances of Lnl_Cardholder and Lnl_Visitor.

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	FIRSTNAME	First name.	Edit
datetime (string)	LASTCHANGED	Person last changed	View
string	LASTNAME	Last name.	Edit
string	MIDNAME	Middle name.	Edit

Type	Name	Description	Access
int32	DATABASEID	The database identifier in an Enterprise system that identifies the system containing the cardholder data.	View

Lnl_PersonSecondarySegments

Description: An association between a person and that person's assigned secondary segments. Present only in segmented systems where cardholder or visitor segmentation is enabled.

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	PERSONID	Reference to Lnl_Person.ID. Cardholder or Visitor ID. Key field.	Read
int32	SEGMENTID	Secondary segment to which the person belongs. Key field.	Read

Lnl_PrecisionAccessGroup

Description: A defined set of unique access privileges for assignment to individual cardholders. Only present if the system is configured to use precision access. For more information, refer to "Precision Access Form" in the *System Administration User Guide*.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	The ID of the precision access group. Key field.	View
string	Name	The name of the precision access group.	View
int32	Type	The type of precision access group. Possible values: 1 (Inclusion), 2 (Exclusion)	View

Type	Name	Description	Access
int32	SegmentID	The ID of the segment associated with the precision access group.	View

Lnl_PrecisionAccessGroupAssignment

Description: An assignment relationship between a badge and a precision access group. Only present if the system is configured to use precision access. For more information, refer to "Precision Access Form" in the *System Administration User Guide*.

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	BadgeKey	A key value uniquely identifying a badge. Key field.	Read
int32	PrecisionAccessGroupID	The ID of the precision access group assigned to the badge. Key field.	Read

Lnl_ProhibitedPassword

Description: The prohibited password list defined in the system.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Password	The prohibited password list.	Edit

Lnl_PTZPreset

Description: PTZ presets configured by the OnGuard software.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	PresetID	Preset ID. Key field.	View
int32	CameraPreset	Preset number stored on the camera.	View
int32	Channel	Channel ID of the recorder.	View
int32	Duration	Number of seconds, applicable to continuous preset (PresetType=3).	View
Float	Focus	Value of the focus.	View
Float	Iris	Value of the iris.	View
string	Name	Name of the preset.	View
Float	Pan	Value of the pan.	View
int32	PanelID	Value of the recorder.	View
int32	PresetType	Type of PTZ preset. 1 = Absolute 2. = Relative 3 = Continuous 4 = Camera preset	View
Float	Tilt	Value of the tilt.	View
Float	Zoom	Value of the zoom.	View

Lnl_Reader

Description: A reader defined in the security system.

Abstract: No

Access: View/Modify

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	AccessMode	The reader mode setting for when the reader is online and communicating with the access panel. For more information, refer to the Reader Mode table below.	View

Type	Name	Description	Access
int32	Address	The address of the reader (0 to 31).	View
string	Aux1Name	The name of the first auxiliary input.	View
string	Aux2Name	The name of the second auxiliary input.	View
string	Aux3Name	The name of the third auxiliary input.	View
int32	ControlType	The type of reader.	View
int32	ExtendedOpenTime	For LenelS2 hardware only. Specifies the held open time for badges with the extended strike/held times feature enabled. This field is intended for anyone who needs extra time to proceed through a doorway. Values range from 1 to 131070 seconds.	View
int32	ExtendedStrikeTime	For LenelS2 hardware only. Specifies the reader strike time for badges with the extended strike/held times feature enabled. This field is intended for anyone who needs extra time to proceed through a doorway. Values range from 1 to 255 seconds.	View
string	FriendlyName	A descriptive name for the reader. **Note:** If an Lnl_Reader instance is updated to change the FriendlyName property, this change also affects the name of the reader shown in Cardholder Self Service and in Cumulus.	Edit
int32	GatewayAddress	Address of the SimonsVoss gateway to which the reader belongs.	View
string	GatewayHostName	Hostname of the SimonsVoss gateway to which the reader belongs.	View
int32	GatewayIPPort	The port number of the SimonsVoss Gateway to which the reader belongs.	View
string	HostName	The name of the workstation where the Communication Server associated with this reader's panel is running.	View

Type	Name	Description	Access
bool	IsPairedPrimary	If true, indicates that the reader is the primary reader of a paired set of readers.	View
bool	IsPairedSecondary	If true, indicates that the reader is the secondary reader of a paired set of readers.	View
int32	MobileReaderType	Indicates the mobile reader type: 1 = Not mobile 2 = BlueDiamond	Edit
string	Name	Display name.	View
int32	OfflineMode	The reader mode setting for when communication is lost between the reader and the access panel. For more information, refer to the Reader Mode table below.	View
int32	OpenTime	The number of seconds the door can be held open before an alarm is generated. For LenelS2 hardware, values range from 1 to 131070 seconds. For other types of hardware, values range from 1 to 255 seconds.	View
string	Out1Name	The name of the first reader output.	View
string	Out2Name	The name of the second reader output.	View
int32	PanelID	ID of the panel to which this reader belongs. Key field. Reference to Lnl_Panel.ID.	View
string	PanelName	Name of the panel to which this reader belongs. Reference to Lnl_Panel.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	PanelTypeName	The panel type name.	View
int32	PortNumber	The number of the port on the access panel to which the reader is attached.	View
int32	ReaderID	Internal database ID. Key field.	View
int32	ReaderNumber	A number that differentiates this reader from other readers using the same port and address. Values typically range from 0 to 7, but may vary depending on reader type.	View

Type	Name	Description	Access
int32	SecondaryID	If IsPairedPrimary is true, this is the ID of the associated secondary reader of the paired set of readers. Reference to Lnl_Reader.ReaderID.	View
int32	SegmentID	Segment to which this reader belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	StrikeTime	The number of seconds a strike or lock is open (activated) when access is granted. Typically, this is set from 5 to 10 seconds, but possible values range from 1 to 255 seconds.	View
int32	TimeAttendanceType	The time and attendance reader configuration. not used = 0 (or <empty>) Entrance Reader = 1 Exit Reader = 2	View

Methods:

void OpenDoor()

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

void SetMode([in] int32 Mode)

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

void GetMode ([out] int32 Mode)

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

Parameters:

int32 Mode: Reader mode to be set. Allowed values are:	
MODE_LOCKED	0
MODE_CARDONLY	1
MODE_PIN_OR_CARD	2
MODE_PIN_AND_CARD	3
MODE_UNLOCKED	4
MODE_FACCODE_ONLY	5

int32 Mode: Reader mode to be set. Allowed values are:	
MODE_CYPHERLOCK	6
MODE_AUTOMATIC	7
MODE_UNLOCK_NEXT_EXIT	37
MODE_TEMPORARY_LOCKED	40
MODE_TEMPORARY_LOCKED_CANCEL	41
MODE_DEFAULT	100

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.

Before placing a reader into the **ACC_MODE_TEMPORARY_LOCKED** state, the reader's **MNT_CTRL_LOCKDOWN_ACTIVATION** permission is checked.

Before placing a reader into the **ACC_MODE_TEMPORARY_LOCKED_CANCEL** state, the reader's **MNT_CTRL_LOCKDOWN_DEACTIVATION** permission is checked.

void SetBiometricVerifyMode([in] boolean Value)

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

Note: Using this method requires that you configure at least one biometric type for the reader's controller. You must also configure the desired biometric template type to greater than `0` on the **System Options > Biometrics** tab.

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.

Note: Using this method requires that you enable the First Card Unlock option on the reader's controller.

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	1
RDRSTATUS_OPTION_MISMATCH	Options Mismatch	2
RDRSTATUS_CNTTAMPER	Cabinet Tamper	4
RDRSTATUS_PWR_FAIL	Power Failure	8
RDRSTATUS_TAMPER	Reader Tamper	10
RDRSTATUS_FORCED	Door Forced Open	20
RDRSTATUS_HELD	Door Held Open	40
RDRSTATUS_AUX	Auxiliary Input 1	80
RDRSTATUS_AUX2	Auxiliary Input 2	100
RDRSTATUS_AUX3	Auxiliary Input 3	400
RDRSTATUS_BIO_VERIFY	Bio Verify	800
RDRSTATUS_DC_GND_FLT	DC Ground Fault	1000
RDRSTATUS_DC_SHRT_FLT	DC Short Fault	2000
RDRSTATUS_DC_OPEN_FLT	DC Open Fault	4000
RDRSTATUS_DC_GEN_FLT	DC Generic Fault	8000
RDRSTATUS_RX_GND_FLT	RX Ground Fault	10000
RDRSTATUS_RX_SHRT_FLT	RX Short Fault	20000
RDRSTATUS_RX_OPEN_FLT	RX Open Fault	40000
RDRSTATUS_RX_GEN_FLT	RX Generic Fault	80000
RDRSTATUS_FIRST_CARD_UNLOCK	First Card Unlock Mode	100000
RDRSTATUS_EXTENDED_HELD_MODE	Extended Held Mode	200000
RDRSTATUS_CIPHER_MODE	Cipher Mode	400000
RDRSTATUS_LOW_BATTERY	Low Battery	800000

uint32 Status – device status:		
uint32 Status	**Description**	**Device status**
RDRSTATUS_MOTOR_STALLED	Motor Stalled	1000000
RDRSTATUS_READHEAD_OFFLINE	Read Head Offline	2000000
RDRSTATUS_MRDT_OFFLINE	MRDT Offline	4000000
RDRSTATUS_DOOR_CONTACT_ OFFLINE	Door Contact Offline	8000000
RDRSTATUS_LOCKED	Reader's status is locked	80000000
RDRSTATUS_UNLOCKED	Reader's status is unlocked	100000000
RDRSTATUS_UNLOCK_NEXT_EXIT	Reader's status will be unlocked after the next exit	200000000
RDRSTATUS_KAR_PENDING	The reader status is pending. You can now acknowledge a door forced open or door held open alarm.	800000000

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.

Abstract: Yes

Access: View

Superclass: Lnl_Input

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the reader's access panel is running.	View
int32	InputID	The input number. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Type	Name	Description	Access
string	Name	The name of the associated reader input.	View
int32	PanelId	The ID of the associated panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderId	The ID of the associated reader. Key field.	View
string	ReaderName	The name of the associated reader. Reference to Lnl_Reader.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Lnl_ReaderInput1

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

Abstract: No

Access: View

Superclass: Lnl_ReaderInput

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the reader's access panel is running.	View

Type	Name	Description	Access
int32	InputID	The input number. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	Name	The name of the associated reader input.	View
int32	PanelId	The ID of the associated panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderId	The ID of the associated reader. Key field.	View
string	ReaderName	The name of the associated reader. Reference to Lnl_Reader.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Methods:

void Mask();

Sends a command to mask a specific reader input.

void Unmask();

Sends a command to unmask a specific reader input.

void GetHardwareStatus([out] uint32 Status)

Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:	
ALRM_STATUS_SECURE	0
ALRM_STATUS_ACTIVE	1
ALRM_STATUS_GND_FLT	2

uint32 Status – device status:	
ALRM_STATUS_SHRT_FLT	3
ALRM_STATUS_OPEN_FLT	4
ALRM_STATUS_GEN_FLT	5
OA_HW_STATUS_MASK_INPUT_MASKED	100

Lnl_ReaderInput2

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

Abstract: No

Access: View

Superclass: Lnl_ReaderInput

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the reader's access panel is running.	View
int32	InputID	The input number. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
string	Name	The name of the associated reader input.	View
int32	PanelId	The ID of the associated panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderId	The ID of the associated reader. Key field.	View

Type	Name	Description	Access
string	ReaderName	The name of the associated reader. Reference to Lnl_Reader.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Methods:

void Mask();

Sends a command to mask a specific reader input.

void Unmask();

Sends a command to unmask a specific reader input.

void GetHardwareStatus([out] uint32 Status)

Status is only retrieved from the hardware when the UpdateHardwareStatus is called on the parent ISC.

uint32 Status – device status:	
ALRM_STATUS_SECURE	0
ALRM_STATUS_ACTIVE	1
ALRM_STATUS_GND_FLT	2
ALRM_STATUS_SHRT_FLT	3
ALRM_STATUS_OPEN_FLT	4
ALRM_STATUS_GEN_FLT	5
OA_HW_STATUS_MASK_INPUT_MASKED	100

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.

Abstract: Yes

Access: View

Superclass: Lnl_Output

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the reader's access panel is running.	View
string	Name	The name of the associated reader output.	View
int32	OutputID	The output number. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	PanelId	The ID of the associated panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderId	The ID of the associated reader. Key field.	View
string	ReaderName	The name of the associated reader. Reference to Lnl_Reader.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

Lnl_ReaderOutput1

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

Abstract: No

Access: View

Superclass: Lnl_ReaderOutput

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the reader's access panel is running.	View
string	Name	The name of the associated reader output.	View
int32	OutputID	The output number. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	PanelId	The ID of the associated panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderId	The ID of the associated reader. Key field.	View
string	ReaderName	The name of the associated reader. Reference to Lnl_Reader.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

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.

void GetHardwareStatus([out] uint32 Status)

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_ReaderOutput2

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

Abstract: No

Access: View

Superclass: Lnl_ReaderOutput

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	HostName	The name of the workstation where the Communication Server associated with the reader's access panel is running.	View
string	Name	The name of the associated reader output.	View
int32	OutputID	The output number. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	PanelId	The ID of the associated panel. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderId	The ID of the associated reader. Key field.	View

Type	Name	Description	Access
string	ReaderName	The name of the associated reader. Reference to Lnl_Reader.Name. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	SegmentID	Segment to which the parent panel belongs. Reference to Lnl_Panel.SegmentID. **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View

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.

void GetHardwareStatus([out] uint32 Status)

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_ReaderRequest

Description: A request raised by a person for accessing readers.

Abstract: No

Access: View/Add

Superclass: Lnl_AccessRequest

Platforms: OnGuard

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	Name	Name of the associated reader.	View
string	ReaderFriendlyName	The descriptive name of the reader for which access is being requested.	View
int32	PanelID	Panel to which access request should be submitted. Key field.	Read
int32	ReaderID	Reader to which access request should be submitted. Key field.	Read
int32	PersonID	Internal ID of the person who requested access to the reader. Key field. See Lnl_Person.ID.	Read
int32	Type	Request type ID: 0: Reader	View
int32	Status	Request status ID: 0: Submitted 1: Approved 2: On Hold 3: Denied	View
int32	EventSerialNumber	The serial number of the associated access denied event, if this request was initiated by an access denied event.	Read
datetime (string)	StartDate	Start date the cardholder requests for the reader.	Read
datetime (string)	EndDate	End date the cardholder requests for the reader.	Read
int32	SubmittedByUserID	The user ID of the user who submitted the request.	View
int32	ApprovedByUserID	The user ID of the user who approved the request.	View
int32	DeniedByUserID	The user ID of the user who denied the request.	View
int32	OnHoldByUserID	The user ID of the user who put the request on hold.	View
string	SubmittedNote	Notes entered when submitting this request.	Read
string	ApprovedNote	Notes entered when approving this request.	View

Type	Name	Description	Access
string	DeniedNote	Notes entered when denying this request.	View
string	OnHoldNote	Notes entered when putting this request on hold.	View
datetime (string)	SubmittedDate	The date and time when the request was submitted.	View
datetime (string)	ApprovedDate	The date and time when the request was approved.	View
datetime (string)	DeniedDate	The date and time when the request was denied.	View
datetime (string)	OnHoldDate	The date and time when the request was put on hold.	View
boolean	EmailCardholder	Whether the cardholder is notified.	Read
boolean	EmailAccessManager	Whether the approver is notified.	Read

Methods:

void Approve([in] string Note, [in] boolean EmailCardholder);

Approves the Reader Request. setting ApprovedDate to current date/time.

void Deny([in] string Note, [in] boolean EmailCardholder);

Denies the Reader Request. setting DeniedDate to current date/time.

void Hold([in] string Note, [in] boolean EmailCardholder);

holds the Reader Request. setting OnHoldDate to current date/time.

Parameters:

Note: Notes when the request is approved, denied and put on hold.

EmailCardholder: Whether the cardholder should be notified.

Lnl_RequestableReader

Description: A reader associated with an access level to which cardholders can request access. For more information, refer to the AvailableForRequest property in Lnl_AccessLevel on page 224.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	PanelID	ID of the panel to which this reader belongs. Key field. Reference to Lnl_Panel.ID.	View
int32	ReaderID	Internal database ID for the requestable reader. Key field.	View
string	Name	Display name for the reader.	View
string	FriendlyName	Descriptive name for the reader.	View

Lnl_Segment

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

Abstract: Yes

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	EXTENDEDOPTIONS	Only in segmented systems. A number formed by combining bitwise flags. Bit designations: • 0x01 - Extended Options Access Levels • 0x02 - Extended Options - extended Cardholder storage • 0x04 - Extended Options - override Locked Reader Mode **Note:** This property is only returned when **get instances** is called with version 1.9 or later.	View
int32	ID	Internal database ID. Key field.	View
string	NAME	Display name.	View
string	TYPE	The segment type.	View
string	SERVERNAME	The Regional Server's name. This field is filterable.	View

Lnl_SegmentGroup

Description: A segment group in the security system. Present in segmented systems only. Refer to Lnl_SegmentGroupMember on page 334 to determine which segments make up a segment group.

Abstract: No

Access: View

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

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/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Note: **Add** and **Modify** permissions are available when **post instances** is called with version 1.4 or later. **Delete** permission is available when **delete instances** is called with version 1.2 or later. Built-in system timezones with ID = 0, 1, or 2 are not editable.

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	SegmentID	Segment ID to which the time zone belongs. Segment groups and dynamic segments are not allowed for timezones.	Read
string	SegmentName	Name of the segment to which the timezone is assigned. This property is only returned when `get instances` is called with Version 1.9 or later.	View

Type	Name	Description	Access
string	Name	Name of the timezone. Name is unique in the selected segment.	Edit

Lnl_TimezoneInterval

Description: A time zone interval used by instances of Lnl_Timezone.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Notes: **Add** and **Modify** permissions are available when **post instances** is called with version 1.4 or later. **Delete** permission is available when **delete instances** is called with version 1.2 or later. Built-in system timezones with ID = 0, 1, or 2 are not editable.

Starting with version 1.9 of **get instances** and version 1.4 of **put/post instances**, the military time format: `HH:MM` (for example, `23:59`) is expected for the **StartTime** and **EndTime** properties.

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	TimezoneID	Lnl_Timezone of which this interval is a part of. Key field. Allowed value range: **0** to **5**.	Read
datetime (string)	StartTime	Time of day when interval becomes active.	Edit
datetime (string)	EndTime	Time of day when interval stops being active. **EndTime** cannot be earlier than **StartTime**.	Edit
boolean	Monday - Sunday	Day of the week when interval is active. There are seven individual boolean properties, one for each day of the week.	Edit
boolean	HolidayType1 - HolidayType8	Holiday type during which the interval is active. There are eight individual boolean properties, one for each holiday type.	Edit

Lnl_User

Description: A user defined in the system.

Abstract: No

Access: View/Add /Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Note: When modifying an Lnl_User instance, if the **password** property provided in the modify request is null or an empty string, the existing password is unchanged. For all other properties, providing null or an empty string in the modify request will set that property to null.

Type	Name	Description	Access
string	ID	Internal database ID. Key field.	View
string	LogonID	Internal Account User name.	Edit
string	Password	Internal Account Password. This property cannot be viewed.	Edit
string	FirstName	First Name.	Edit
string	LastName	Last Name.	Edit
boolean	Enabled	Determines whether user is enabled	Edit
boolean	HasInternalAccount	If true, indicates that the user has an internal account.	Edit
boolean	IgnorePasswordExpiration	If true, indicates that this user's password never expires. The sa account and SA delegate accounts are exceptions: this property is always false for the sa user and SA delegate users, and cannot be set to true.	Edit
boolean	SAPowers	If true, indicates that the user is an SA Delegate with all permissions enabled. A user can only be given SA Delegate permissions at the time the user is created by the Root SA or by another SA Delegate user; existing users cannot be converted into SA Delegate users. For security, the first SA Delegate user created should log into the system and disable the Root SA user.	Read
sint32	SystemPermissionGroupID	System User Permission Group. See Lnl_UserPermissionGroup.ID.	Edit
sint32	MonitoringPermissionGroupID	Monitor User Permission Group. See Lnl_UserPermissionGroup.ID.	Edit

Type	Name	Description	Access
sint32	CardPermissionGroupID	Cardholder User Permission Group. See Lnl_UserPermissionGroup.ID.	Edit
sint32	ReportPermissionGroupID	Indicates the Report Permission Group ID. This is a required field, but defaults to 0 which provides no report permissions.	Edit
sint32	FieldPermissionID	Field/Page Access Group. Reference to Lnl_UserFieldPermissionGroup.ID.	Edit
sint32	SegmentID	User's Segment ID This property cannot be viewed. Use Lnl_UserSecondarySegments to see a full list of the user's segments.	Read
list	SegmentID_list	A list of the User's Segment ID. For example: [1,2,3]. This property cannot be viewed. Use **Lnl_UserSecondarySegments** to see a full list of the user's segments. This parameter requires version 1.2 or later of the **add/ modify instances** requests, and version 1.7 or later of the **get instances** request.	Read
sint32	MonitoringZoneID	Monitoring Zone ID. Reference to Lnl_MonitoringZone.ID.	Edit
datetime (string)	Created	Date user was created	View
datetime (string)	LastChanged	Date user was modified	View
string	Notes	Notes associated with the user. This field is not filterable.	Edit
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
boolean	PasswordChangeRequired	Determines if the user is forced to change the password at the next login.	Edit
boolean	IsPasswordCaseSensitive	Determines if the user's password is case sensitive.	View

Type	Name	Description	Access
sint32	DatabaseID	The database identifier in an Enterprise system that identifies the replication setting for the User. The value has a default value of 'Local System Only' which matches the default through the OnGuard software.	Edit

Lnl_UserAccount

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

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	UserID	Internal ID of the user who owns this account. See Lnl_User.ID. Key field.	Read
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
string	DirectoryID	Internal ID of the directory to which this account belongs. See Lnl_Directory.ID.	View/Edit

Lnl_UserPermissionGroup

Description: A user permission group defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

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	Segment to which the user permission group belongs	View
sint32	PTZPriority	PTZ Priority for the users belonging to this group	View
boolean	CanLoginToDataConduIT	Shows if the users in this group can login to DataConduIT	View
boolean	CanViewLiveVideo	Shows if the users in this group can view live video	View
boolean	CanViewRecordedVideo	Shows if the users in this group can view recorded video	View
boolean	CanSearchVideo	Shows if the users in this group can search video	View
boolean	DevicesExcluded	Shows if the devices in the associated group are excluded	View

Lnl_UserFieldPermissionGroup

Description: A user field permission group defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Permission Group name.	View
sint32	SegmentID	Segment to which the user field permission group belongs.	View

Lnl_UserPermissionDeviceGroupLink

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

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	UserPermissionGroupID	User permission group. See Lnl_UserPermissionGroup.ID. Key field.	View
sint32	DeviceGroupID	Device Group ID. See Lnl_DeviceGroup.ID. Key field.	View

Lnl_UserReportPermissionGroup

Description: A user report permission group defined in the system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
sint32	ID	Internal database ID. Key field.	View
string	Name	Permission Group name.	View
sint32	SegmentID	Segment to which the user report permission group belongs.	View
sint32	DatabaseID	The database identifier in an Enterprise system that identifies the replication setting for the group. 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 all assigned segments.

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
string	UserID	Internal ID of the user Lnl_User.ID.	Read
sint32	SegmentID	A segment to which the user belongs.	Read

Lnl_VideoLayout

Description: Configuration of the matrix view for displaying video channels.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	VideoLayoutID	Video layout ID.	View
string	LayoutName	Name of the video layout.	View
int32	VideoTemplateID	Template ID.	View
string	UserID	User ID.	View
int32	WorkstationID	Workstation ID.	View

Lnl_VideoLayoutSource

Description: Source details for the cells in the video layout.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	VideoSourceID	Unique ID of the video source.	View

Type	Name	Description	Access
int32	PanelID	VideoRecorderID	View
int32	CameraID	The ID of the camera connected to the video recorder.	View
string	CameraName	Name of the camera.	View
int32	Channel	The channel of the camera.	View
int32	LayoutID	The layout ID.	View
int32	LayoutCellID	The specific cell in the layout.	View

Lnl_VideoTemplate

Description: A video template for the matrix view of the player window.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	TemplateID	Video template ID.	View
string	TemplateName	Video template name.	View
string	TemplateXml	The structure of the template, described in XML.	View

Lnl_Visit

Description: A visit in the security system.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
int32	CARDHOLDERID	LNL_CARDHOLDER.ID - the host	Read

Type	Name	Description	Access
int32	DELEGATEID	The person who schedules or maintains the event on behalf of the host. Optional property.	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
datetime (string)	LASTCHANGED	Visit last changed	View
string	NAME	The user-friendly name of this object. Optional property.	Edit
string	PURPOSE	Visit purpose.	Edit
datetime (string)	SCHEDULED_TIMEIN	Scheduled start time	Edit
datetime (string)	SCHEDULED_TIMEOUT	Scheduled end time	Edit
int32	SIGNINLOCATIONID	The ID of the visitor sign-in location. This parameter is required starting with version 1.1 of the **get type** request and version 1.2 of the **add/ modify instances** requests.	Edit
datetime (string)	TIMEIN	Actual start time	View
datetime (string)	TIMEOUT	Actual end time	View
int32	TYPE	Visit type, values are user-defined	Edit
int32	VISIT_EVENTID	The ID of the visit event. Reference to Lnl_VisitEvent.ID. If this property is empty when calling **post Lnl_Visit**, a new visit event is created. If a valid Visit_EventID is passed, an additional visitor is added to the event.	Edit
string	VISIT_KEY	A unique identifier assigned to a scheduled visit, used to sign visitors in or out.	View
int32	VISITORID	Lnl_Visitor.ID - the visitor.	Read

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]**int32** BadgeTypeID, [in]**string** PrinterName, [in]**int64** AssignedBadgeID, [in]**string** AssignedBadgeID_str);

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 64-bit badge ID you want to assign the visitor, a badge already in the system.
- AssignedBadgeID_str - A string representation of the AssignedBadgeID. You cannot provide both AssignedBadgeID and AssignedBadgeID_str in the same call.
- 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/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	RECIPIENTNUMBER	Internal database ID. Key field.	Read
int32	VISITID	Lnl_Visit.ID - ID of the visit. Key field.	Read
string	ACCOUNTID	ID of the entry in the external directory. For example, with Microsoft directories, this property would contain the account's security identifier (SID).	Read

Type	Name	Description	Access
string	DIRECTORYID	Internal ID of the directory to which this account belongs.	Read
string	EMAILADDRESS	Recipient e-mail address.	Read
boolean	INCLUDEDEFAULTRECIPIENTS	Whether the default recipients are notified	Read
boolean	INCLUDEHOST	Whether the visit host is notified	Read
boolean	INCLUDEVISITOR	Whether the visitor is notified	Read
int32	PERSONID	Lnl_Person.ID - ID of the person receiving the e-mail	Read
int32	SEGMENTID	Segment to which the visit email recipient belongs.	Read

Lnl_VisitEvent

Description: A hosted event with visits and visitors.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	The internal database ID	View
string	Name	The user-friendly name of this object.	Edit
int32	CardholderID	The host of the visit event. Reference to Lnl_Cardholder.ID.	Edit
int32	DelegateID	The person who schedules or maintains the event instead of the host.	Edit
int32	DatabaseID	The database identifier in an Enterprise system that identifies the system containing the event data.	Edit
datetime (string)	Scheduled_TimeIn	The time the visit event is scheduled to start.	Edit
datetime (string)	Scheduled_TimeOut	The time the visit event is scheduled to complete.	Edit

Type	Name	Description	Access
datetime (string)	LastChanged	The last time the properties of the visit event changed.	View
int32	SignInLocationID	The ID of the visitor sign in location. This parameter is required starting with version 1.1 of the **get type** request and version 1.2 of the **add/modify instances** requests.	Edit

Method:

HRESULT SendEmail([in] int32 ID, [in] Boolean UseSystemDefaults, [in] string Action, [in] Boolean, IncludeHost, [in] Boolean IncludeVisitor, [in] Boolean IncludeDefRecipents, [in] string EmailList);

Sends an email to the host, co-hosts, default recipients (if configured), delegate (if visit event is created by the delegate), and individual mails to visitors when a visit event is scheduled with multiple visitors.

Parameters:

- ID - Visit_EventID passed as 'property_value_map'.
- UseSystemDefaults - If true, then emails will be sent as configured in System Administation settings. All other parameters passed to this method are ignored. If false, then emails will be sent as configured by the parameters.
- Action - Add/Modify. 'Add' when visit event is added and 'Modify' when visit event is updated.
- IncludeHost - Whether the host is notified.
- IncludeVisitor - Whether the visitor is notified.
- IncludeDefRecipients - Whether the default recipients are notified.
- EmailList - A list of semi-colon separated e-mail recipients (other than the visitor, host, or defaults).

Lnl_Visitor

Description: A visitor in the security system.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Person

Platforms: OnGuard

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

Type	Name	Description	Access
string	ADDRESS	The visitor’s address.	Edit
string	CITY	The visitor’s city.	Edit

Type	Name	Description	Access
string	EMAIL	The visitor's email address.	Edit
boolean	EMAIL_DOCUMENTS	When true, the email sent to a visitor when the visitor is signed in will contain as an attachment the Visitor PDF Document (Lnl_MultimediaObject. Datatype = 13, Lnl_MultimediaObject. Objecttype = 513) associated with the visitor's Lnl_MultimediaObject PERSONID = X.	Edit
string	EXT	The visitor's extension.	Edit
string	OPHONE	The visitor's office phone number.	Edit
string	ORGANIZATION	The visitor's organization.	Edit
int32	PRIMARYSEGMENTID	This property is only available when visitors are segmented.	Read
string	STATE	The visitor's state.	Edit
string	TITLE	The visitor's title.	Edit
string	ZIP	The visitor's zip code.	Edit

Note: When using OpenAccess to modify a visitor with a badge of a badge type associated with a Cumulus bundle to change any attributes that are configured to be sent to Cumulus, a call is made to Cumulus to provide the updated data. For more information, refer to "Send to Cumulus Form" in the *System Administration User Guide*.

Lnl_VisitDelegateAssignment

Description: A visit delegate assignment in the system.

Abstract: No

Access: View/Add/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	HostID	The host. Reference Lnl_Cardholder.ID.	Read

Type	Name	Description	Access
int32	DelegateID	The delegate. Reference Lnl_Cardholder.ID.	Read

Lnl_VisitSelfServiceStation

Description: The Visitor Self-Service station associated with a sign-in location.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	The internal database ID.	View
string	Name	The Visitor Self Service station name.	Edit
string	UniqueIdentifier	A unique identifier representing the Visitor Self Service device.	Edit
int32	VisitSignInLocationID	A reference to the Lnl_VisitSignInLocation instance corresponding to this station.	Edit

Lnl_VisitSignInLocation

Description: The sign-in location for visits.

Note: The sign-in location assigned to a visit event cannot be changed, except for the sign-in location's Name. To delete a sign-in location, you must first unassign that sign-in location from all visit events, or delete the visit events with that sign-in location.

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	The internal database ID.	View
string	Name	The visit sign-in location name.	Edit

Type	Name	Description	Access
int32	SegmentID	The ID of the segment to which the sign-in location belongs. This property is only available if segmentation is enabled.	Read
int32	WorldTimezoneID	The time zone of the sign-in location. Reference to Lnl_WorldTimeZone.ID.	Edit

Lnl_Workstation

Description: The workstation used to configure the Monitor Zones used on monitoring stations.

Abstract: No

Access: View

Superclass: None

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	WorkstationID	The ID of the workstation.	View
string	WorkstationName	The name of the workstation.	View
int32	DatabaseID	The database identifier in an Enterprise system that identifies the system containing the workstation data. For more information, refer to Settings on page 154.	View

Lnl_WorldTimezone

Description: A world time zone defined in the security system.

Abstract: No

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View

Type	Name	Description	Access
int32	Bias	The current bias for local time translation on this computer, in minutes.	View
int32	DaylightBias	A bias value that is used during local time translations that occur during daylight time.	View
int32	DaylightDay	DaylightDayOfWeek of the DaylightMonth when the transition from standard time to daylight saving time occurs on this operating system. Example: If the transition day (DaylightDayOfWeek) occurs on a Sunday, then the value "1" indicates the first Sunday of the DaylightMonth, "2" indicates the second Sunday, and so on. The value "5" indicates the last DaylightDayOfWeek in the month.	View
int32	DaylightHour	Hour of the day when the transition from standard time to daylight saving time occurs on an operating system.	View
int32	DaylightMinute	Minute of the DaylightHour when the transition from standard time to daylight saving time occurs on an operating system.	View
int32	DaylightMonth	Minute of the DaylightHour when the transition from standard time to daylight saving time occurs on an operating system. For example, “1” is January, “2” is February, and so on.	View
int32	DaylightSecond	Second of the DaylightMinute when the transition from standard time to daylight saving time occurs on an operating system.	View
int32	DaylightWeek	Week of the DaylightMonth when the transition from standard time to daylight saving time occurs on an operating system.	View
string	DisplayName	The user-friendly name, and how the timezone appears.	View

Type	Name	Description	Access
int32	GMTOffset	In areas of the United States that observe daylight saving time, local residents move their clocks ahead one hour when daylight saving time begins. As a result, their GMT offset would change from GMT - 5h to GMT - 4h. In places not observing daylight saving time, the local GMT offset remains the same all year. Arizona, Puerto Rico, Hawaii, U.S. Virgin Islands, and American Samoa do not observe daylight saving time.	View
boolean	IsDaylightSaving	True if in an area of the United States that observes daylight saving time.	View
int32	StandardBias	Bias value to use when daylight saving time is not in effect. This property is ignored if a value for StandardDay is not supplied. The value of this property is added to the Bias property to form the bias during standard time.	View
int32	StandardDay	StandardDayOfWeek of the StandardMonth when the transition from daylight saving time to standard time occurs on an operating system. If the transition day (StandardDayOfWeek) occurs on a Sunday, then the value "1" indicates the first Sunday of the StandardMonth, "2" indicates the second Sunday, and so on. The value "5" indicates the last StandardDayOfWeek in the month.	View
int32	StandardHour	Hour of the day when the transition from daylight saving time to standard time occurs on an operating system.	View
int32	StandardMinute	Minute of the StandardDay when the transition from daylight saving time to standard time occurs on an operating system.	View

Type	Name	Description	Access
int32	StandardMonth	Month when the transition from daylight saving time to standard time occurs on an operating system. For example, “1” is January, “2” is February, and so on.	View
int32	StandardSecond	Second of the StandardMinute when the transition from daylight saving time to standard time occurs on an operating system.	View
int32	StandardWeek	Week of the StandardMonth when the transition from daylight saving time to standard time occurs on an operating system.	View
string	Windows_TZID	The unique name that Windows uses to identify the timezone in the registry.	View

User-Defined Value Lists

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

- Lnl_BUILDING
- Lnl_DEPT
- Lnl_DIVISION
- Lnl_LOCATION
- Lnl_TITLE
- Lnl_VISITTYPE

Abstract: No

Access: View/Add/Modify/Delete

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

Type	Name	Description	Access
int32	ID	Internal database ID. Key field.	View
string	NAME	Name of the list value.	Edit
int32	SEGMENTID	Segment to which the user-defined value list belongs.	Read

Association Classes

When using a filter to get instances of an association class, configure the filter as shown in this example:

```
type_name=Lnl_AccessLevelGroupAssignment and
filter=AccessGroup="Lnl_AccessGroup.ID=1"
```

This filter provides all access levels that belong to the access group with ID = 1.

Lnl_AccessLevelGroupAssignment

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

Abstract: No

Access: View

Superclass: Lnl_Element

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

Access: View

Superclass: Lnl_Element

Platforms: OnGuard

Properties:

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

Lnl_CardholderAccount

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

Abstract: No

Access: View

Superclass: Lnl_PersonAccount

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Account	ACCOUNT	Reference to the account
ref:Lnl_Cardholder	PERSON	Reference to the cardholder

Lnl_CardholderBadge

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

Abstract: No

Access: View

Superclass: Lnl_BadgeOwner

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Badge	BADGE	Reference to the badge
ref:Lnl_Cardholder	PERSON	Reference to the cardholder

Lnl_CardholderMultimediaObject

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

Abstract: No

Access: View

Superclass: Lnl_MultimediaObjectOwner

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_MultimediaObject	MULTIMEDIAOBJECT	Reference to the multimedia object
ref:Lnl_Cardholder	PERSON	Reference to the cardholder

Lnl_DirectoryAccount

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

Abstract: No

Access: View

Superclass: Lnl_Element

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

Access: View

Superclass: Lnl_Element

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

Access: View

Superclass: Lnl_Element

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

Access: View

Superclass: Lnl_Element

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

Access: View

Superclass: Lnl_Element

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

Access: View

Superclass: Lnl_Element

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

Access: View

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_VisitorBadge

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

Abstract: No

Access: View

Superclass: Lnl_BadgeOwner

Platforms: OnGuard

Properties:

Type	Name	Description
ref:Lnl_Badge	BADGE	Reference to the badge
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

Access: View

Superclass: Lnl_MultimediaObjectOwner

Platforms: OnGuard

Properties:

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

CHAPTER 8

Using OpenAccess to Send Alarms to OnGuard

OpenAccess 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 Logical Source using System Administration before using this capability of OpenAccess. OpenAccess will use this source as the device to display alarms for in Alarm Monitoring. For more information, refer to Add a Logical Source on page 342.

Note: Starting with OnGuard 7.6, the preferred method for using OpenAccess to send alarms to OnGuard is with the **post send_incoming_events** call. For more information, refer to post send_incoming_events on page 132.

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

After configuring the Logical Source, you should also add any Logical Device and Logical 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 Logical Source child and sub-child devices in Alarm Monitoring. For more information, refer to Add a Logical Device on page 344 and Add a Logical Sub-Device on page 346.

Sending alarms to Alarm Monitoring is very simple.

Note: To use the following example, change “localhost” to the Fully Qualified Domain Name (FQDN) of your server.

Here is an example using an HTTP request:

```
POST localhost/api/access/onguard/openaccess/execute_method
Header:
    Session-Token : 12345-67890-12345-67890
    Application-Id : SUPPLIED_APPLICATION_ID
Body:
{
        "type_name" : "Lnl_IncomingEvent",
        "property_value_map" :
        {
        },
```

```
11          "method_name" : "SendIncomingEvent",
12          "in_parameter_value_map" :
13          {
14                  "Description" : "Test event from OpenAccess",
15                  "Source" : "Logical Source 6"
16          }
17  }
```

The above sample will display and log an alarm with the description “Test Event From OpenAccess” from controller name “Logical Source 6”. This sample assumes System Administration was used to create a Logical Source called “Logical Source 6” and demonstrates how to send an alarm to Alarm Monitoring. The Source refers to the logical source setup in System Administration. The Description property is the actual text of the alarm that will display in Alarm Monitoring and be logged into the OnGuard database.

The Lnl_IncomingEvent object has no properties and currently supports the methods “SendIncomingEvent”, “AcknowledgeAlarm”, and “AcknowledgeAlarmWithAuth”. For more information, refer to Lnl_IncomingEvent on page 265.

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

If ‘IsAccessGrant’ is set to true, the ‘Granted Access’ event will be reported for the Logical Source, Device or Sub-Device 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 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 269.

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 Logical Source name provided matches a Logical Source. If not, then a check is made to see if it matches the name of a LenelS2 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’.

CHAPTER 9

Logical Sources Folder

OpenAccess is an advanced application integration service that allows real time, bidirectional integration between OnGuard and third party IT sources. OpenAccess 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.

Note: To achieve the best performance in OpenAccess when sending events, confirm that the Linkage Server is configured (the machine host name is set in **System Administration > Administration > System Options > Linkage Server host**), and the service is running.

Logical Sources Folder

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

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

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

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

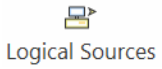
Note: The Logical Sources method has four parameters: the source, description, device (optional), and sub-device (optional). The source of the Logical Sources method must match the source name on the Logical Sources form. If the optional parameters are used, the device of the Logical Sources method must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

- Have at least one panel (non-system Logical Source) configured and marked online so that the Communications Server will work properly with Logical 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 configured, events can be received successfully by Alarm Monitoring from Logical Sources.

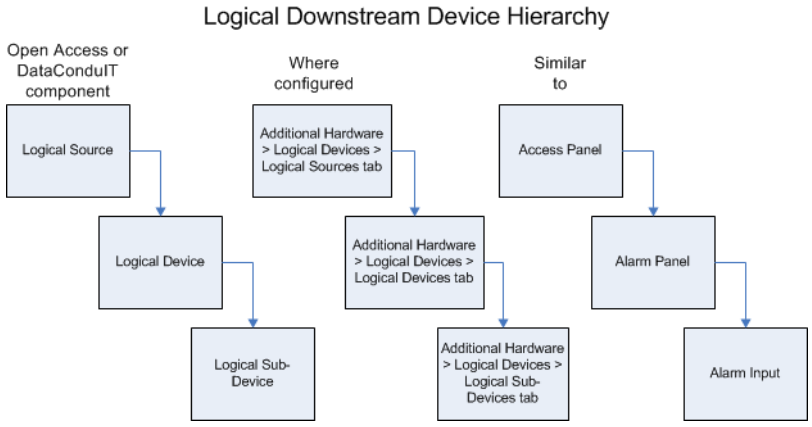
This folder is displayed by selecting ***Logical Sources*** from the ***Additional Hardware*** menu, or by selecting the Logical Sources toolbar buttonin System Administration or ID CredentialCenter.

Toolbar Shortcut



Logical Source Downstream Devices

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



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

User Permissions Required

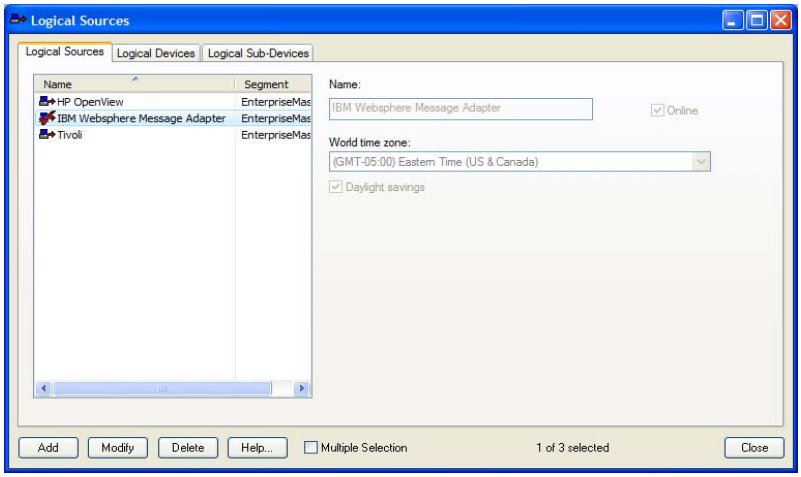
Add, Modify, and Delete Logical Sources, Devices, and Sub-Devices

The add, modify, and/or delete Logical Sources permissions, determine what functions a user can perform on Logical Sources, Logical Devices, and Logical Sub-Devices in the Logical 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 Logical Sources, Devices, and Sub-Devices

In addition, user permissions are required to trace Logical Sources, Logical Devices, and Logical 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.

Logical Sources Form



Listing window
Lists Logical Source names.

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

Online
The Logical Source is always online and ready for use. This status does not apply to the Logical Source.

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 Logical Source.

Modify
Click this button to modify a Logical Source.

Delete
Click this button to delete a Logical 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 Logical Sources.

Close
Click this button to close the Logical Sources folder.

Logical Sources Form Procedures

Use the following procedures on this form.

Add a Logical Source

1. From the ***Additional Hardware*** menu, select ***Logical Sources***. The Logical Sources folder opens.
2. On the Logical 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 to which this Logical Source will be assigned.
   b. Click [OK].
4. In the **Name** field, type a name for the Logical Source.
5. Select whether the Logical 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 Logical Source configured, there must be at least one panel (non-system Logical Source) configured and marked online so that the Communications Server will work properly with Logical 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 received successfully by Alarm Monitoring and event subscribers from Logical Sources.

Modify a Logical Source

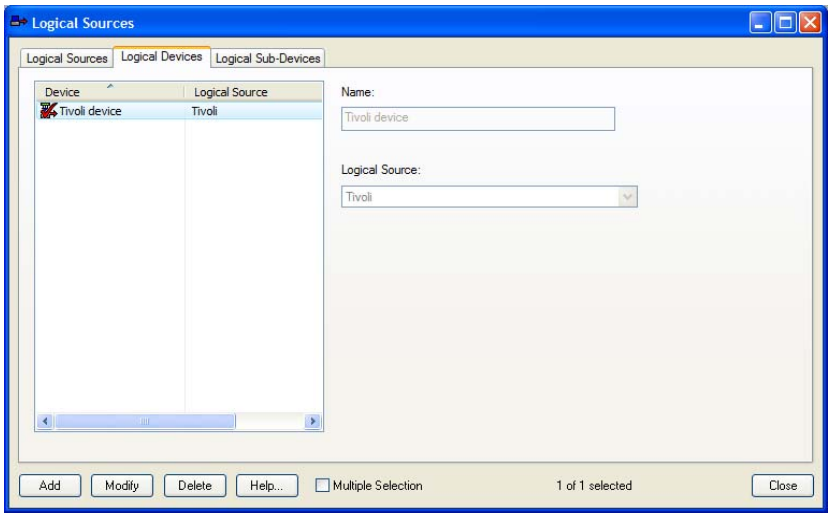
1. From the ***Additional Hardware*** menu, select ***Logical Sources***.
2. On the Logical Sources tab, select the entry you want to modify from the listing window.
3. Click [Modify].
4. Make any changes.
5. Click [OK].
6. A prompt to confirm that you want to make the modification displays. Click [OK].

Delete a Logical Source

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

1. From the ***Additional Hardware*** menu, select ***Logical Sources***.
2. On the Logical 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].

Logical Devices Form



Listing window
Lists Logical Device names.

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

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

Add
Click this button to add a Logical Device.

Modify
Click this button to modify a Logical Device.

Delete
Click this button to delete a Logical 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 Logical Devices.

Close
Click this button to close the Logical Sources folder.

Logical Devices Form Procedures

Use the following procedures on this form.

Add a Logical Device

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

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

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

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

6. Click [OK].

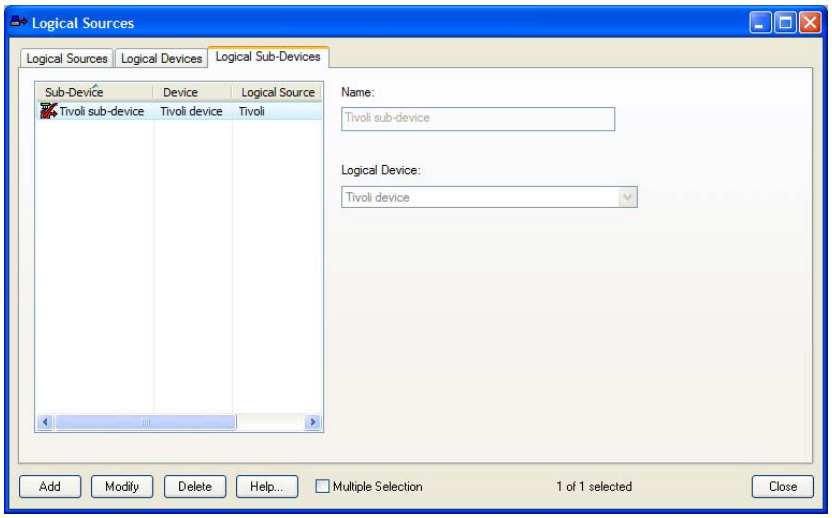
Modify a Logical Device

1. From the ***Additional Hardware*** menu, select ***Logical Sources***.
2. Click the Logical 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 Logical Device

1. From the ***Additional Hardware*** menu, select ***Logical Sources***.
2. Click the Logical 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].

Logical Sub-Devices Form



Listing window

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

Name

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

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

Add
Click this button to add a Logical Sub-Device.

Modify
Click this button to modify a Logical Sub-Device.

Delete
Click this button to delete a Logical 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 Logical Sub-Devices.

Close
Click this button to close the Logical Sources folder.

Logical Sub-Devices Form Procedures

Use the following procedures on this form.

Add a Logical Sub-Device

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

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

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

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

6. Click [OK].

Modify a Logical Sub-Device

1. From the ***Additional Hardware*** menu, select ***Logical Sources***.
2. Click the Logical 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 Logical Sub-Device

1. From the ***Additional Hardware*** menu, select ***Logical Sources***.
2. Click the Logical 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].

CHAPTER 10

Troubleshooting

This section describes the use of techniques to troubleshoot issues with the LS OpenAccess service.

It is also useful to understand the OpenAccess architecture. For more information, refer to OpenAccess Architecture on page 30.

Enabling Verbose Logging

For more information, refer to Enabling Verbose Logging on page 35.

Testing if the LS OpenAccess Service is Online

For a quick test to see if the LS OpenAccess service is configured and online, create a client that supports the **get version** request/response. A **get version** response confirms that the service is online. For more information, refer to get version on page 62.

Error Messages

This section defines how the LS OpenAccess service communicates errors to the client. If an error occurs, the response will include an entry named **error** which is a key/value map. The response may otherwise contain only standard response headers.

The error is a string in a period-delimited hierarchical string that follows the platform namespace. For example:

```
"error":
{
    "code":"openaccess.general.invalidapplicationid",
    "message":"You are not licensed for OpenAccess."
}
```

Name	Type	Required	Description
code	string	yes	The error code, which is a string with a full namespace.
message	string	no	An optional human-readable message to display after the translated error code. The message is sent in the client locale, if possible.
...	...	no	Other optional fields, as defined along with the error code.

For more information about error codes, refer to Errors List on page 350.

Errors List

Notes: This section does not contain every OpenAccess error code that might be logged. Only the most common error codes are listed.

The error code sent to the client generally contains less detail than is logged at the server. Check the server logs for more information.

If the LS OpenAccess service cannot connect to the database, that can cause many of the OpenAccess errors. Confirm that the service has a database connection.

Error Code	Root Cause and Resolution	HTTP Error Code
openaccess.general.missingrequestitem	When a required request item is not present in the request, the name of the missing item is part of the message.	400
openaccess.general.exception	General exception. Refer to server logs for details.	500
openaccess.general.invalidrequestitem	The operation failed because of an invalid request item input. Details provided in the error message.	400
system.invalid_field	The operation failed because of an invalid request item input. Details provided in the error message.	400
openaccess.general.decodingfailed	Failed to generate binary data from base-64 string.	400
openaccess.general.invalidapplicationid	You are not licensed to use OpenAccess with the provided application ID. The application ID is not valid.	401

openaccess.general.invaliddb-connection	The database connection is not function-ing. The request cannot be fulfilled. Try again later.	503
openaccess.general.invalid-sessiontoken	The provided session token is not recog-nized as a previously-authenticated token to the service.	401
openaccess.general.invalid-typename	Failed to retrieve type details. Type name specified is not valid. Refer to server logs for details.	400
openaccess.general.invalid-userpassword	The operation failed because the new password you created does not meet the password policies. Details are provided in the error message.	400
openaccess.authentica-tion.failedtoauthenticate	Authentication failed. Could be caused by invalid credentials. Refer to server logs for details.	401
openaccess.authentica-tion.invalidinternallogin	Authentication of an internal user account failed because of invalid creden-tials.	400
openaccess.authentica-tion.invalidthirdpartyauthli-cense	The OpenID Connect feature is not licensed. Acquire a valid license to use this feature.	400
openaccess.authentica-tion.passwordexpired	The user password is expired.	400
openaccess.getin-stances.maxpagesizeex-ceeded	The maximum page size is 100.	400
openaccess.editinstance.error	The add/modify/delete operation failed. Details will be provided in the error mes-sage.	500
openaccess.execute-method.error	Execution of the method failed. Details provided in the error message.	500
system.insufficient_privilege	The user is not the owner of the event subscription.	400
system.missing_field	When a required request item is not present in the request, the name of the missing item is part of the message.	400
system.parse	The filter specified is invalid.	400
system.http_error_code	A timeout occurred because the request took longer than allowed, as configured with the request_timeout property in the openaccess.ini file (for more information, refer to Timeout Property on page 27). Also, the request might be malformed or contain invalid parameters.	40_ (400, 404, 408, and so on)

system.insufficient_privilege	The user logged into OpenAccess does not have the permissions required to perform the requested operation.	403
system.not_implemented	When an unsupported operation is attempted (for example, you try to delete an instance of a type that does not support delete).	501

Warning List

Note: This section does not contain every OpenAccess warning. Only the most common warnings are listed.

Warning Code	Root Cause and Resolution
openaccess.warning. passwordexpiration	Users receive this warning during authentication if their passwords are almost expired. The following policy settings are used when the authentication response contains this warning: • is_expiration_reminders_enabled • expiration_first_reminder_days • expiration_reminder_days For more information, refer to get password policy settings on page 161.

Symptoms and Solutions

Errors Connecting to the Message Broker

There are errors connecting to the Message Broker when it is running on a server not connected to any domain (only local workgroup).

For information about certificates and how to correct these errors, refer to the "OnGuard and the Use of Certificates" appendix in the *OnGuard Installation Guide*.

SSL/TLS Secure Channel Errors

All applications using the LS OpenAccess service must reference the OpenAccess API in a way that exactly matches the certificate name. If the certificate name uses the server's Fully Qualified Domain Name (FQDN), then applications must access OpenAccess using the server's FQDN. Likewise, if the certificate name does not use the server's FQDN, then applications must access OpenAccess by not using the server's FQDN.

For information about certificates and how to correct these errors, refer to the "OnGuard and the Use of Certificates" appendix in the *OnGuard Installation Guide*.

CORS Errors When Accessing the OpenAccess API from a Web Application

There are Cross-Origin Resource Sharing (CORS) errors when accessing the OpenAccess API from a web application.

For more information, refer to Cross-Origin Resource Sharing on page 53.

CORS Errors When Running the Cardholder Sample Web Application

There are CORS errors when running the Cardholder Sample web application.

The Getting Started chapter provides details on how to load the cardholder sample web application properly. See Sample Applications on page 36.

The Using OpenAccess chapter provides details about CORS. See Cross-Origin Resource Sharing on page 53.

Errors After Updating the nginx.conf File

*There are errors accessing the OpenAccess API after updating the **nginx.conf** file.*

Perform the following steps to troubleshoot the NGINX configuration:

1. Verify NGINX is running by checking for two running **nginx.exe** processes. Also point a web browser to https://<Fully Qualified Domain Name of server>:8080. If the default NGINX page loads, the Web Server is running. If the default NGINX page loads on the server but fails to load on the client, there is a problem with the connection between the client and server.
2. Review the NGINX error log (**C:\ProgramData\Lnl\nginx\logs\error.log**). For more verbose logging, add the following line near the top of the **C:\ProgramData\Lnl\nginx\conf\nginx.conf** file. Refer to http://nginx.org/en/docs/ngx_core_module.html#error_log for details about the NGINX error log directive:

   ```
   error_log logs/error.log info;
   ```

Event Subscribers Do Not Receive Any Events

Event subscribers are not receiving any events.

Confirm the following:

- The LS Event Context Provider is running.
- There is an online panel in your default monitoring zone. For more information, refer to Add a Logical Source on page 342.
- Verify the filter you used to subscribe to events. Also verify that the property names are valid. For more information, refer to Using Event Filters with Subscriptions on page 48.

Note: The Event Generator is a useful troubleshooting tool. Use Event Generator to create “fake” events that can be received by event subscribers. For more information, refer to Appendix A: Event Generator on page 357.

Event Subscribers Do Not Receive Software Events

Event subscribers are not receiving software events.

Confirm that on the **System Administration > Administration > System Options** form, the **Generate software events** checkbox is checked.

Cannot Log Into OpenAccess Using Manual Single Sign-On

Manual single sign-on does not work with OpenAccess, after specifying the directory, user name, and password.

Confirm the following:

- The user name and password are correct.
- The specified directory is configured correctly in System Administration on the **Administration > Directories** form.
- Also on the Directories form, confirm that the **Enable single sign-on** and **Allow manual single sign-on** checkboxes are selected.

Note: OpenAccess does not work with directories of type **Windows Local Accounts** because local accounts do not support manual single sign-on. To work around this, create a directory of type **Microsoft Windows NT 4 Domain** and enter the machine name in the **Domain** field.

Cannot Get Cardholders From Active Directory with Administrator Account

Use **Domain.exe** located in the ***TroubleShooting*** directory in the ***DataConduIT*** documentation file structure to determine if this may be the problem. If the NT4Domain is different from the W2KDomain, update the LNL_DIRECTORY.DIR_HOSTNAME in the Access Control database to match the NT4Domain. In case this is Oracle, use all upper case.

A sample SQL query to do this follows; 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
```

Alternatively, add both the fully qualified Active directory and the NT 4 Domain directory.

Unsuccessful OpenAccess Operations From Behind a Network Proxy

An error occurs when performing OpenAccess operations from behind a network proxy.

If you are using OpenAccess to perform OpenAccess operations from behind a network proxy (for example, issue mobile credentials, or using a third-party provider for OnGuard authentication, or performing any other operation that requires OpenAccess to reach an external network location) and are behind a network proxy, an error might occur. To resolve this error, on the server where the LS OpenAccess service is running, change the logon account for the LS OpenAccess service from Local System to a user whose account has the correct proxy settings configured.

LS OpenAccess Service Does Not Start in a Cluster Environment

The LS OpenAccess service does not start when installed in a cluster environment.

For information on how to troubleshoot this issue, refer to the *Using Microsoft Cluster Services with OnGuard* guide.

Appendices

APPENDIX A

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.

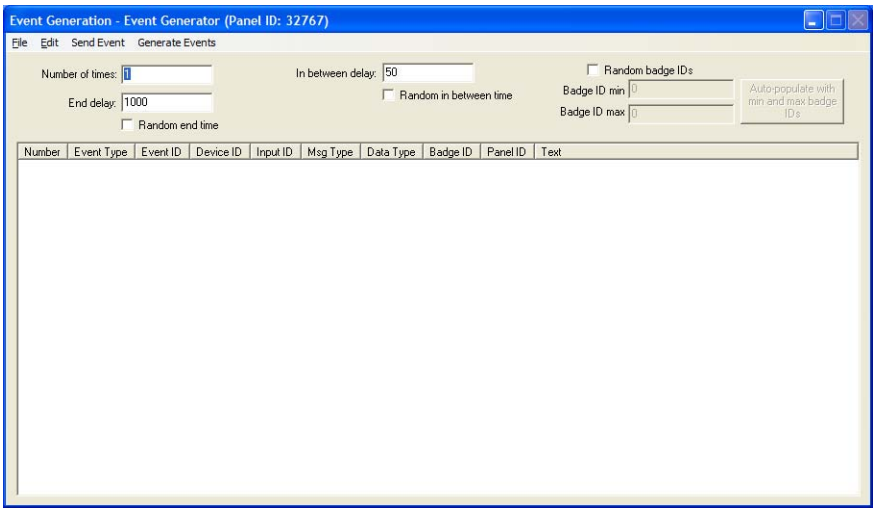
The Event Generator is available on the LenelS2 Connect™ portal at: https://connect.lenels2.com/s/downloads.

Note: When accessing the **Downloads** section at https://connect.lenels2.com/s/downloads, make sure to select the version of OnGuard that is currently installed.

It is also available on the OnGuard Software Development Kit (SDK) media image.

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 365.



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.

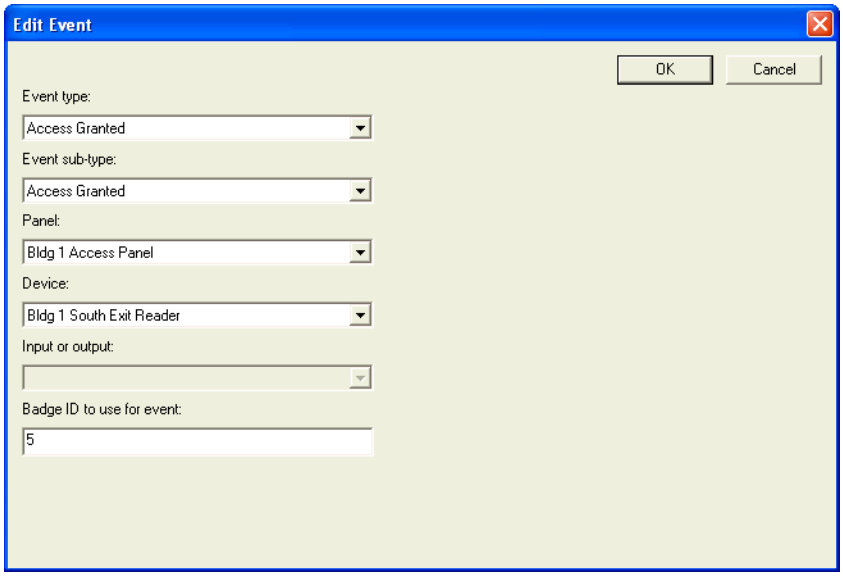
Edit Event (Simple) Window

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



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:

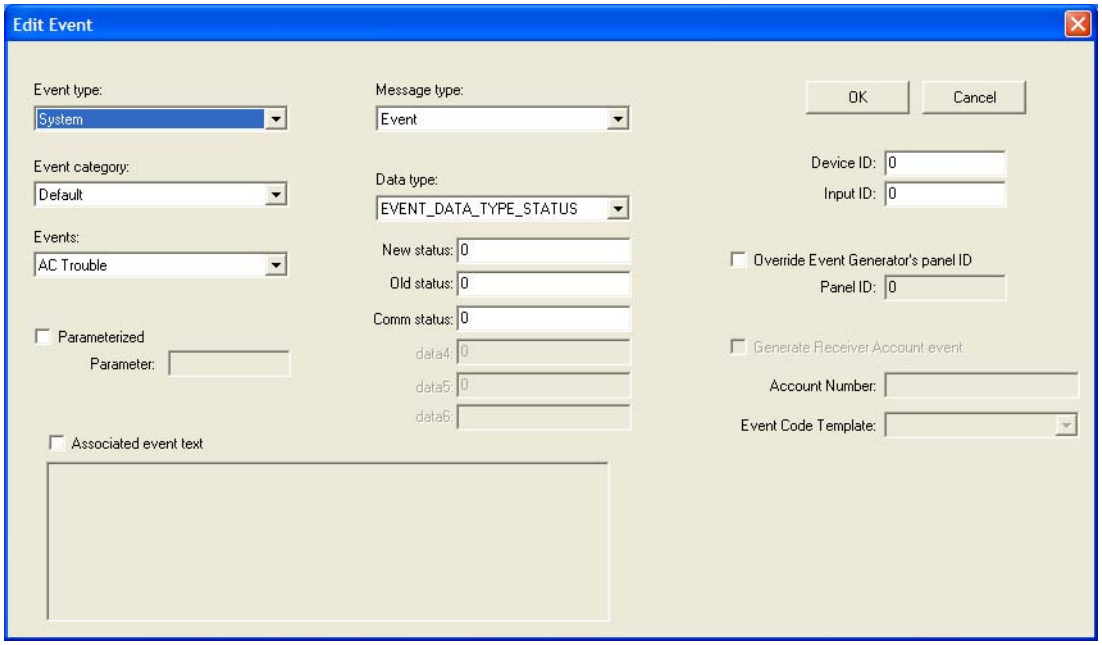
- ***Edit > Create Event > Create Event (Advanced)***, or
- ***Edit > Modify Event > Modify Event (Advanced)*** when an event is selected

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, dat5, 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



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 (x86)\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, dat5, 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 362.

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 the OnGuard software.

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 364.

Generate Events

This option in the Edit menu performs the same function as **Generate Events**. For more information, refer to Generate Events on page 365.

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 (x86)\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 (x86)\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 media image in the **program files (x86)\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 366.

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.

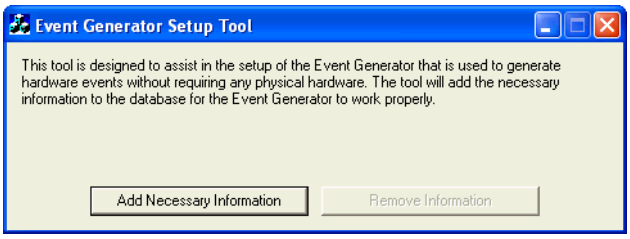
   Copy from **C:\Program Files (x86)\OnGuard Software Development Kit\EventGenerator** directory to **C:\Program Files (x86)\OnGuard** directory

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

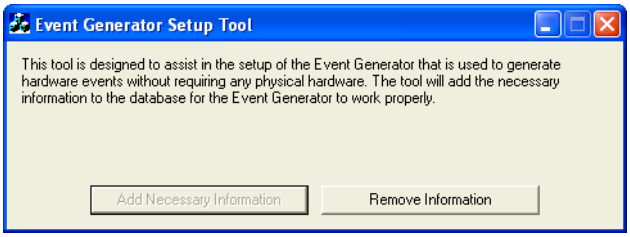
3. Register the **LnlEventGeneratoru.dll**. For more information, refer to Registering the LnlEventGeneratoru.dll on page 366.
4. In the OnGuard software, add hardware such as access panels, readers, and so on. Keep in mind this hardware does not have to be “online”; it might even be hardware that doesn’t 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 (x86)\OnGuard**) 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 366.

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.

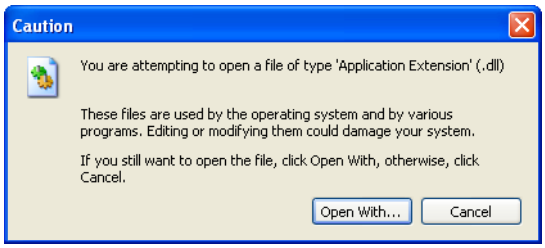


8. Run the Communication Server as an application. To do this:
   a. Open the Communication Server.
   For more information, refer to "Using OnGuard in the Supported Operating Systems" in the Installation Guide.
   b. Right-click on the  icon in the system tray, and then select ***Open Communication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

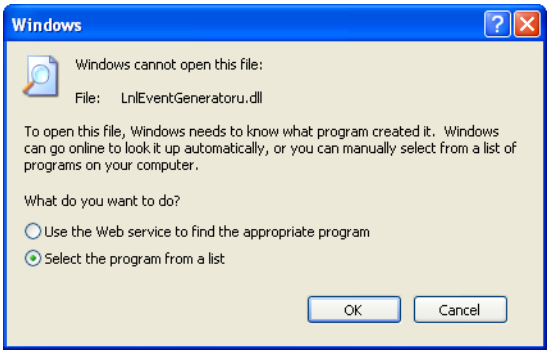
Registering the LnlEventGeneratoru.dll

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].



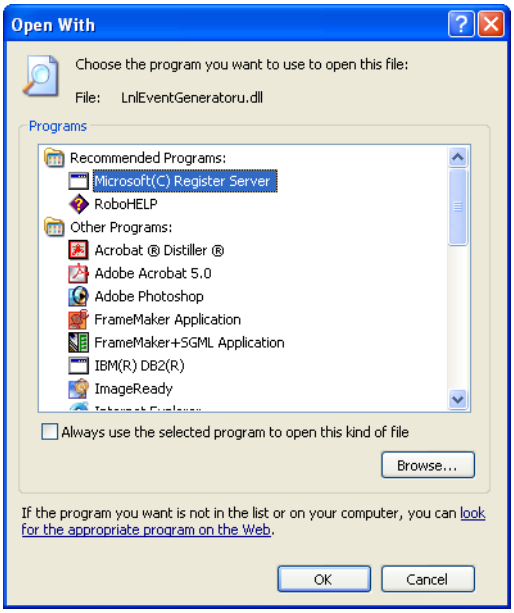
4. Click [Open With...].
5. Select the **Select the program from list** radio button, then click [OK].



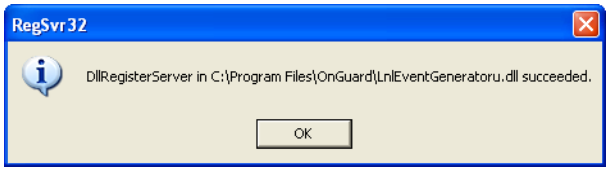
6. The Open With window opens. Click [Browse...], navigate to **C:\Windows\system32**, and then double-click on the **regsvr32.exe** file.

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 **LnlEventGeneratoru.dll** file is now registered. If you were setting up Event Generator, return to Setting Up the Event Generator on page 365.

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:

Index

D

E

M

1212 Pittsford-Victor Road
Pittsford, New York 14534 USA
Tel +1.866.788.5095
www.LenelS2.com