SEL Configuration API

Version: `1.1`

Overview

The SEL Configuration API is an application programming interface (API) that provides access to protection, security, and communications device configuration stored in the AcSELerator® Database and managed by Device Manager in AcSELerator QuickSet SEL-5030 Software.

The API is deployed as a representational state transfer (RESTful) web service running as a Windows service. The RESTful web service is a de facto standard that can be accessed by client applications by using most programming languages.

This manual describes the programming model and operations provided by the API. The target audience is the application programmer writing a custom application that accesses device configuration information.

The resource model of the API closely follows the business object model presented in the Device Manager user interface. For example, both Device Manager and the API support creating, viewing, and updating devices. Device properties, displayed in their respective tabs in Device Manager, are accessed as properties of the device API resource. Composite resources of a device such as connection, settings working copy, and settings versions, are modeled as sub-objects of the device resource. Devices and folders are generalized as 'assets' in the API, and the hierarchy displayed and managed in the Connection Explorer is exposed in the asset node resource of the API.

> *CAUTION: This is a powerful API that can insert, modify, and permanently delete device configurations stored in the AcSELerator Database. Perform all development by using a test instance of the database and test the custom application thoroughly before deploying to a production database.*

Getting Started with the API

First, install and configure the API as explained in the instruction manual.

You may then choose to enable development and testing features that are disabled by default to prevent their misuse in a production environment. You can enable these features in your development environment by using the Enable API Metadata configuration setting, as explained in the instruction manual.

Enable API Metadata adds the following API endpoints:

Endpoint[1]	**Description**

Endpoint[1]	Description
/swagger-ui	The Swagger user interface (https://swagger.io/tools/swagger-ui) lets you visualize and interact with the resources of the API through your web browser. You can use Swagger to gain a better understanding of the resource model and specific operations, and to help test your application as it is developed. Notes: • The Try it out! button in Swagger invokes an API operation against the live, connected database so it must be used with care. • Use the 'Authorize' button to enter the user name and password of a Device Manager user whose credentials are used for API operations. See Client Authentication and Authorization for more information. • Swagger waits until the Try it out! button is invoked to pass user credentials to the API, so it may appear that the Authorize operation is successful when incorrect credentials are entered. If you receive an authentication (401) or permissions (403) error, use the Authorize button to log out, and then re-authorize.
/types/csharp /types/typescript /types/typescript.d	The types endpoint provides templates for API request and response packet definitions for the C# and Typescript languages. Note that the C# types file is also included with the Coding Examples.

[1] Relative to the base API endpoint, e.g., http:/localhost/5231/ or https:/apiserver/5231/

Coding Examples

The following C# examples are provided:

Example	Description
CreateFolderAndDevicesExample	• Log into the API. • Call POST /sca/asset_node to create a Substation folder at the Connection Explorer root. • Call POST /sca/asset_node to create an SEL-3620 Ethernet Security Gateway under the folder. • Call POST /sca/asset_node to create an SEL-421-5 Protection, Automation, and Control System under the SEL-3620. • Log out of the API.

Example	Description
GetDevicesExample	• Log into the API. • Call POST /sca/devices/get_list to get all devices. • Write out the device count and some properties of the first few devices. • Log out of the API.
UpdateDeviceExample	• Log into the API. • Call POST /sca/asset_node to create an SEL-421-5 Protection, Automation, and Control System. • Call GET /sca/device to fetch the relay. • Call PUT /sca/device to update the relay. • Log out of the API.

The following supporting files are provided with the examples:

File	Description
Types.cs	C# types for API requests and responses. These are used in the examples and can be used in your applications.

To run an example, construct an instance of the example class, then call its Perform() method.

The constructor takes in the user name and password, and optionally the base API URL. For a default installation that is running HTTP and accepting only local connections, the URL can be omitted, as follows:

```
GetDevicesExample example = new GetDevicesExample("janedoe", "secret");
await example.Perform();
```

When the API is configured to run HTTPS on a remote server, provide the base API URL in the constructor, as follows:

```
GetDevicesExample example = new GetDevicesExample("janedoe", "secret", "https://apiserver:5231/");
await example.Perform();
```

Notes:

- Some examples create folders and devices in the AcSELerator Database. After running the example, you can use Device Manager to view the resulting information.
- Because folders and devices created by the examples are retained in the database, they must first be deleted before rerunning the example. Otherwise, the subsequent creation operation will fail with an HTTP status of 422 and error message “An item with this name already exists at this location.”

Client Authentication and Authorization

To fetch or update configuration information through the API, it is first necessary to log into the API with the credentials of a registered Device Manager user. Both local and LDAP users are supported.

The user must meet these criteria:

- The user must have a non-blank password.
- The user must have group membership with at least the following permissions:
  - Log On
  - Manage Devices
  - If adding or updating configuration information, Update Database

Logging In to the API

POST /authenticate/credentials

Request Body:

Name	Description	Data type
UserName	The name of the user to be authenticated. The user must be configured in Device Manager and have the Log On permission.	*string*
Password	The password of the user to be authenticated. Blank passwords are not supported.	*string*

Response Body:

Name	Description	Data type
bearerToken	Bearer token that hold credentials for accessing the API. This value must be provided in the 'Authorization' HTTP Header on subsequent API operations. See the API Instruction Manual for information on configuring bearer token expiration.	*string*
Other properties returned in the response are subject to change and should not be used.		

Logging Out of the API

POST /authenticate/logout

The properties returned in the response are subject to change and should not be used.

HTTP Header Fields

The following HTTP header fields are observed by the API and can be set by the client:

Field	Description
Accept	This is a standard header that is used to identify the content types that the client accepts. The API supports the application/json content type:
Authorization	This is a standard header that is used to pass a bearer token for client authentication. The required format for the header value is: Bearer *token-value*
User-Agent	This is a standard request header (RFC 1945) used by the client to self-identify the client application. Use of this header is optional. If used, the value should be constant for the lifetime of the client instance.
X-SEL-Client-Instance-Id	This is an SEL custom header used to self-identify an instance of client execution. Use of this header is optional. If used, the value should be constant for the lifetime of the client instance.

Field	Description
X-SEL-Tracking-Id	This is an SEL custom header used to track a client task through the system. Use of this header is optional. If the tracking ID is not provided with a request, the API will generate one for that request. If the client task consists of multiple requests to the API, the client should provide the same tracking ID for each of the task's requests.

API Operations

The Summary section lists the available API operations, followed by detailed descriptions of the operations and their parameters.

API Processing Notes:

- Extra *key/value pairs* contained in the JSON body of a request are ignored. No validation errors or any other indication is given to the caller.
- Update (PUT) requests use the convention that a null value in a PUT request indicates that the requester does not want to change the original value. Only non-null values in a PUT request are persisted to the database. A value that is missing from the request body is treated as null and will also result in no update to the original value.
- Collections read from the API are returned unordered, unless otherwise specified.
- Properties of type *string (date-time)* are UTC timestamps in ISO 8601 format. Example: 2017-10-30T18:29:55.2574840Z

Working with Device Configurations

The device configuration model of the API closely follows that presented in the 'Settings' tab in Device Manager.

A device may have a configuration working copy and one or more versions. These are shown in the Workspace and Version History tables, respectively, in Device Manager. The API provides operations to create and update the working copy, create (baseline) a version from the working copy, and create a version from provided content.

In addition to its content, a configuration has a set of information properties. These are shown as columns in both the Workspace and Version History tables in Device Manager (e.g., Saved, Saved By, Comment, State, Version).

Configuration content consists of settings and supplementary information contained in miscellaneous files. This follows the model of Settings Database (RDB) files.

> *Updating device configuration with improperly formatted settings or miscellaneous files may result in the configuration being unable to be opened in QuickSet or other SEL software.*

The API supports two formats of configuration content.

Settings Format

In the 'settings' format, settings are directly defined in the request and response packets. Miscellaneous files are represented as Base 64 Encoded content (each miscellaneous file has specific content that is generally not usable by customer applications).

Use the settings format when you want direct access to the settings and groups in your application.

Example:

```
{
  "device_settings": {
    "settings_groups": [
      {
        "name": "1",
        "settings": [
          {
            "name": "RID",
            "value": "FEEDER 1",
            "comment": ""
          },
          {
            "name": "TID",
            "value": "STATION A",
            "comment": ""
          }
        ]
      }
    ],
    "misc_files": [
      {
        "name": "Cfg.txt",
        "content": "W0lORk9dDQpSRUxBWVRZUEU9MDM1MQ..."
      }
    ]
  }
}
```

Files Format

In the 'files' format, both settings and miscellaneous files are represented as Base 64 Encoded content.

Use the file format when your application is written to process Settings Database (RDB) files, or when you are exchanging configuration between instances of the API.

Example:

```
{
  "configuration_files": {
    "settings_files": [
      {
        "name": "SET_1.TXT",
        "content": "W0lORk9dDQpSRUxBWVRZUEU9U0VMLT...."
      }
    ],
    "misc_files": [
      {
        "name": "Cfg.txt",
        "content": "W0lORk9dDQpSRUxBWVRZUEU9MDM1MQ..."
      }
    ]
  }
}
```

Understanding Settings Values From the API

Settings retrieved from the API are the raw settings content saved by QuickSet or other applications. These settings are not intended for direct deployment to an SEL device. There may be slight differences between the settings content retrieved from the API and either the actual settings physical device or how the settings are displayed within SEL software.

Examples:

- Compound device settings may be split up into their individual components for ease of user configuration. For example, this is the case for the terminal-to-bus-zone connections for the SEL-487B Bus Differential and Breaker Failure. For the SEL-487B, the device setting I*nn*BZpC is equivalent to the combination of the three QuickSet settings: TBZT*b*, TBZB*b*, TBZP*b*. Refer to the SEL-487B instruction manual for more information about the relationship between these settings.
- A complicated setting may be stored as non-human readable data. Examples of these settings include the Time-of-Use setting for the SEL-735Power Quality and Revenue Meter, the touchscreen display for the SEL-751 Feeder Protection Relay, and the bay screen for the SEL-400 series relays.
- Settings retrieved from the API contain settings that would be otherwise disabled by the part number of the device. These setting values are persisted by QuickSet so if a user changes the part number, they do not lose previously configured settings values.

Handling Imported Metadata

When importing asset modes or replicating a device configuration version, the API imports metadata as follows:

- Timestamps from the import source are retained so that they reflect the original time of occurrence rather than the time of import.
- When a username from the imported metadata exists in the destination database, that user is recorded with the imported metadata. Otherwise, just the username is recorded in the metadata displayed in Device Manager as "not registered".

Error Handling

The API detects error conditions and reports these errors to client programs by using the HTTP response status code.

The General Status Response packet is returned in the response body to provide additional error information. The following table describes the parts of the response body:

<table>
<tr><th>Name</th><th>Description</th><th>Data type</th></tr>
<tr><td>tracking_id</td><td>A unique identifier for the request that resulted in this status report. This value is either the X-SEL-Tracking-Id request header value provided by the client program, or a value generated by the API (see HTTP Header Fields).</td><td><i>string</i></td></tr>
<tr><td>code</td><td>A code that represents the reported status. This field is not localized and is intended for programmatic processing by API client software.
<table>
<tr><th>HTTP Status</th><th>Code</th><th>Description</th></tr>
</table>
</td><td><i>string</i></td></tr>
</table>

Name	Description	Data type

HTTP Status	Code	Description
401	AuthenticationFailure	The API is unable to authenticate the request. When logging in to the API, authentication failure causes include the following: • The provided user name or password do not match those of a user registered in the AcSELerator Database. Note that the API does not support blank passwords. • The user is marked with "User must change password at next logon attempt." Log into Device Manager to update the password. • The user does not have the Log On permission. On subsequent API requests after logging in, authentication failure causes include the following: • The bearer token returned in the log in response was not properly set in the 'Authorization' HTTP Header of the request. • The bearer token is expired.
400	BadRequest	The request packet is invalid and cannot be processed.
404	DataNotFound	The requested resource cannot be found in the database. The resource ID might be incorrect or the resource might have been deleted.
500	InternalError	An internal error has occurred in the API service. Contact SEL for assistance.

<table>
<tr><th>Name</th><th>Description</th><th>Data type</th></tr>
<tr><td></td><td>
<table>
<tr><th>HTTP Status</th><th>Code</th><th>Description</th></tr>
<tr><td>422</td><td>InvalidRequestParameters</td><td>The request has one or more invalid fields. See <i>details</i> for additional information.</td></tr>
<tr><td>501</td><td>NotImplemented</td><td>The API does not support the functionality required to fulfill the request.</td></tr>
<tr><td>409</td><td>OptimisticLock</td><td>The submitted resource is out of date. Refetch the resource and reapply your changes. See Optimistic Locking for more information.</td></tr>
<tr><td>403</td><td>PermissionsViolation</td><td>The requested operation cannot be performed due to insufficient permissions. Use Device Manager to verify the permissions of the logged-in user.</td></tr>
<tr><td>404</td><td>ResourceNotFound</td><td>The API does not support the type of resource requested. This is typically because of an error in the request URL.</td></tr>
<tr><td>503</td><td>ServiceNotAvailable</td><td>The API reports this error when attempting to log in an LDAP user if the following problems are encountered:<ul><li>The API cannot connect to the AcSELerator® Database.</li><li>The API is not compatible with the version of the AcSELerator® Database.</li><li>LDAP is not configured or is improperly configured. Correct the configuration by using Device Manager.</li><li>The API cannot connect to the configured LDAP server.</li></ul></td></tr>
</table>
</td><td></td></tr>
<tr><td>message</td><td>A localized, human-readable explanation of the reported status.</td><td><i>string</i></td></tr>
<tr><td>message_inserts</td><td>Unused.</td><td><i>string[]</i></td></tr>
<tr><td>details</td><td>When <i>code</i> = "InvalidRequestParameters," <i>details</i> provides additional information on each validation error. For all other values of code, this property is not used.</td><td><i>object[]</i></td></tr>
</table>

The *details* list is included in the General Status Response on request validation errors (code = "InvalidRequestParameters"), with one entry for each request property in error. The following table describes validation details reported by the API:

<table>
<tr><th>Name</th><th>Description</th><th>Data type</th><th></th></tr>
<tr><td>code</td><td>A code that represents a validation condition that was found to be in error. This field is not localized and is intended for programmatic processing by API client software.
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td><i>property</i>LengthWithinMax</td><td>The value provided for <i>property</i> exceeds its maximum length.</td></tr>
<tr><td><i>property</i>CannotBeEmpty</td><td>The value provided for <i>property</i> is empty.</td></tr>
<tr><td><i>property</i>InRange</td><td>The value provided for <i>property</i> is not within its valid range.</td></tr>
<tr><td><i>property</i>IsSupported
<i>property</i>IsValid</td><td>The value provided for <i>property</i> is not supported.</td></tr>
<tr><td><i>property</i>FreeOfInvalidCharacters</td><td>The value provided for <i>property</i> contains invalid characters:
\
/
:
*
?
"
<
>
|</td></tr>
<tr><td><i>resource</i>Exist(s)</td><td>The referenced <i>resource</i> does not exist.</td></tr>
<tr><td>AssetNodeIsValidParent</td><td>The parent asset node cannot contain child nodes.</td></tr>
<tr><td>AssetParentIsValidForThisNode</td><td>The parent asset node contains a child node with a conflicting name.</td></tr>
</table></td><td><i>string</i></td><td></td></tr>
</table>

<table>
<tr><th>Name</th><th>Description</th><th>Data type</th></tr>
<tr><td></td><td>
<table>
<tr><th>Code</th><th>Description</th></tr>
<tr><td>CustomAssetIdUnique</td><td>The provided Custom Asset ID is used in another asset.</td></tr>
<tr><td>ConfigurationVersion<i>format</i>ProvidedOrHasWorkingCopy</td><td>A version cannot be created because the device does not have a working copy and no content was provided in the request.</td></tr>
<tr><td><i>format</i>DeviceTypeMatchesDevice</td><td>The device type in the provided configuration does not match the type of the device.</td></tr>
<tr><td>DeviceSupportsIntegratedSettingsById
DeviceSupportsIntegratedSettingsByDeviceId</td><td>Settings are not supported on this device type.</td></tr>
<tr><td>NameUniqueAtAssetLocation
NameUniqueAtAssetLocationByParentNodeId
AssetNodeNameIsValidAtDestinationNodeId</td><td>The provided name is used in another asset at this location.</td></tr>
<tr><td>InventoryDatabaseMismatch</td><td>Some properties of an asset node in the provided inventory do not match the database.</td></tr>
<tr><td>PermissionSupportedForDevice</td><td>The provided permission is not supported on this device type.</td></tr>
<tr><td>DeviceIsDescendantOfSecurityGateway</td><td>The device must be a descendant of a security gateway to set rights.</td></tr>
<tr><td>RequestParseError</td><td>The provided property value cannot be parsed.</td></tr>
<tr><td>SecurityGatewayTerminationStringIsSupported</td><td>The value provided for TerminationString is not supported for Security Gateway devices.</td></tr>
<tr><td>WorkflowStateIsActive</td><td>The workflow state cannot be assigned because it is inactive.</td></tr>
</table>
</td><td></td></tr>
<tr><td>context</td><td>A context that can be used to associate this detail with a resource or property. This field is not localized and is intended for programmatic processing by API client software.</td><td><i>string</i></td></tr>
</table>

Name	Description	Data type
message	A localized, human-readable explanation of the reported status.	*string*
message_inserts	Unused.	*string[]*
severity	The severity level of the condition. For validation errors, this is always "Error."	*string*

Example

The following is an attempt to create a device, providing invalid values for the type of device ("SEL-000", a non-existent device type) and the name of the device ("Relay\0423", which has an invalid backslash character):

```
POST \sca\asset_node

{
  "asset_type": "device",
  "asset_subtype_name": "SEL-000",
  "name": "Relay\0423",
  "custom_asset_id":,
  "parent_node_id":
}
```

The following shows the API response with a General Status Response with two *details*, one for each invalid request property:

```
{
  "request_id": "fc700494-f07b-462c-942f-6ca5c7cff715",
  "code": "InvalidRequestParameters",
  "message": "The request has one or more invalid fields.  See the details for additional information.",
  "details":
  [
    {
      "code": "NameFreeOfInvalidCharacters",
      "context": "name",
      "message": "This value contains invalid characters.",
      "severity": "Error"
    },
    {
    "code": "AssetSubtypeExists",
    "context": "asset_subtype_name",
    "message": "The referenced item does not exist.",
    "severity": "Error"
    }
  ]
}
```

Optimistic Locking

Optimistic locking refers to a scheme that prevents stale data from being written to the database.

Each record in a database table protected by this scheme has an 'optimistic lock' value that is maintained by the database system each time the record it belongs to is updated. Resources that are persisted in the database include this value in the *optimistic_lock_field* property of responses and requests. Each time a resource is read from the API or returned by the API following a create or update operation, the *optimistic_lock_field* property of the response reflects the current value in the database.

Prior to an update, the *optimistic_lock_field* property from the request is compared to the current value in the database table. If they are the same, the data is considered current and the update is allowed to proceed. However, if the values differ, another updater has modified the database record since the user retrieved the data; it is therefore considered stale and the update is not permitted. The user must fetch a fresh copy of the record's data and reapply any changes that were previously made before attempting the update again.

Name	Type	Description
optimistic_lock_field	unsigned long	Used in the optimistic locking system to prevent stale data from being written to the database.

Summary

Path	Operation	Description
/sca/asset_node	POST	Creates a new asset and assigns it to an asset node.
/sca/asset_node/{id}	GET	Gets an asset node.
/sca/asset_node/{id}/clone	POST	Clones an asset node and its associated assets, including its child nodes.
/sca/asset_node/{id}/move	POST	Moves an asset node.
/sca/asset_nodes/delete_list	POST	Deletes a list of asset nodes and associated assets, including their child nodes.
/sca/asset_nodes/export	POST	Exports a list of asset nodes and associated assets.
/sca/asset_nodes/get_list	POST	Gets a list of asset nodes.
/sca/asset_nodes/import	POST	Adds assets to the database.
/sca/asset_nodes/merge	POST	Merges asset information into the database, adding assets that do not exist.
/sca/database_information	GET	Gets information about the AcSELerator Database served by the API.
/sca/device/configuration/version/baseline	POST	Baseline the configuration working copy as a new configuration version.
/sca/device/configuration/version/files	POST	Creates a configuration version for a device from the provided configuration files.
/sca/device/configuration/version/replicate	POST	Replicates a device configuration version from another database.
/sca/device/configuration/version/settings	POST	Creates a configuration version for a device from the provided device settings.
/sca/device/configuration/version/{id}/files	GET	Gets a configuration version as files.
/sca/device/configuration/version/{id}/settings	GET	Gets a configuration version as settings.
/sca/device/{id}	GET	Gets a device.

Path	Operation	Description
	PUT	Updates a device.
/sca/device/{id}/configuration/working_copy	PUT	Updates information about a configuration working copy.
/sca/device/{id}/configuration/working_copy/files	GET	Gets a configuration working copy as files.
	PUT	Replaces the working copy of a configuration with the provided configuration files.
/sca/device/{id}/configuration/working_copy/settings	GET	Gets a configuration working copy as settings.
	PUT	Replaces the working copy of a configuration with the provided device configuration settings.
/sca/device/{id}/connection	GET	Gets a device connection.
	PUT	Updates a device connection.
/sca/device/{id}/password/{level}	PUT	Updates a device password.
/sca/device/{id}/right	POST	Creates a right.
/sca/device_types/get_list	POST	Gets a list of device types.
/sca/devices/get_list	POST	Gets a list of devices.
/sca/folder/{id}	GET	Gets a folder.
	PUT	Updates a folder.
/sca/folder_types/get_list	POST	Gets a list of folder types.
/sca/folders/get_list	POST	Gets a list of folders.
/sca/rights/delete_list	POST	Deletes a list of rights.
/sca/service_information	GET	Gets information about the API.
/sca/support_data	POST	Gets API service support data.
/sca/template/device	GET	Gets a device template.

Paths

POST /sca/asset_node

Tags: sca

Creates a new asset and assigns it to an asset node.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

AssetNodePostDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
asset_type	The type of the asset.	formData	*string* , x ∈ { "Device" , "Folder" }	**required**
parent_node_id	The unique identifier of the parent of this asset node. A null or empty string indicates that the asset node is a root node.	formData	*string* (uuid)	
asset_subtype	The subtype of the asset. For devices, this is the name of the device type. For folders, this is the name of the folder type.	formData	*string*	**required**

Name	Description	Type	Data type	
custom_asset_id	A user-defined identifier of the asset. If a value is not provided, the API supplies a default for the asset. This value must be unique among all assets.	formData	*string*	
name	The name of the asset. The name must be unique among the siblings of the asset node.	formData	*string*	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
A node in an asset hierarchy.

AssetNodeDto

GET /sca/asset_node/{id}

Tags: sca

Gets an asset node.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type			
id	The unique identifier of the asset node to fetch.	path	*string* (uuid)	**required**		
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK

A node in an asset hierarchy.

AssetNodeDto

POST /sca/asset_node/{id}/clone

Tags: sca

Clones an asset node and its associated assets, including its child nodes.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database

- Manage Devices

A cloned device acquires these parts from the source device: nameplate properties, connection properties, passwords, and (if available) configuration working copy. If the source device does not have an active working copy, then the content of the latest version is included as the cloned working copy, with all information properties null. Otherwise, if the source device has no versions, then no working copy is included in the cloned device.

A cloned folder acquires the properties of the source folder.

Cloned device and folders do not include custom attributes, attached documents, permissions, applications, configuration versions, or TEAM configuration.

REQUEST BODY

application/x-www-form-urlencoded

AssetNodeCloneDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the asset node to clone.	path	*string* (uuid)	**required**
destination_node_id	The unique identifier of the destination for the cloned asset node. A null or empty string indicates that the cloned node is a root node.	formData	*string* (uuid)	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Information about the asset nodes that were cloned.

The audit record contains these ResourceChangeRecordDtos for asset nodes and assets that were added to the database:

```
change_type: 'Create'
resource_type: 'AssetNodeDto'
id : <new asset node id>
attribute_name: 'id'
prior_value: <source asset node id>
current_value: <new asset node id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'id'
prior_value: <source asset id>
current_value: <new asset id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'path'
prior_value: <source asset path>
current_value: <new asset path>
```

CloneResponseDto

POST /sca/asset_node/{id}/move

Moves an asset node.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

AssetNodeMoveDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the asset node to move.	path	*string* (uuid)	**required**
destination_node_id	The unique identifier of the destination node to which the requested node is to be moved. A null or empty string indicates that the moved node is to be a root node.	formData	*string* (uuid)	
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A node in an asset hierarchy.

AssetNodeDto

POST /sca/asset_nodes/delete_list

Tags: sca

Deletes a list of asset nodes and associated assets, including their child nodes.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

AssetNodeDeleteListDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
ids	The unique identifiers of the asset nodes to delete.	formData	*string[]* , multiple parameters (`ids=aaa&ids=bbb`)	**required**

Name	Description	Type	Data type	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

204 No Content
No Content

POST /sca/asset_nodes/export

Tags: sca

Exports a list of asset nodes and associated assets.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices
- Manage Device Permissions (if device_parts_selector includes rights)

The following resources are always included in the export: asset nodes, devices, folders.

REQUEST BODY

application/x-www-form-urlencoded

AssetNodeExportDto

REQUEST PARAMETERS

Name	Description	Type	Data type
export_parts_selector	Selects which additional resources are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+). If the selection is unspecified, no additional resources are exported.	query	*string* , x ∈ { "DeviceConfigurations" , "All" }
device_parts_selector	Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+). If the selection is unspecified, the Default selection is used. Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation. NamePlate properties are always included regardless of selection.	query	*string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }

Name	Description	Type	Data type	
device_configuration_parts_selector	Selects which device configuration resources are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+). If the selection is unspecified, the Default selection is used. Default includes these parts: WorkingCopy.	query	*string* , x ∈ { "WorkingCopy" , "Default" , "All" }	
ids	An optional list of the asset nodes to export. If not provided, all asset nodes are exported.	formData	*string[]* , multiple parameters (`ids=aaa&ids=bbb`)	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
A list of asset nodes and associated assets.

AssetNodeInventoryDto

POST /sca/asset_nodes/get_list

Tags: sca

Gets a list of asset nodes.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

AssetNodeGetListDto

REQUEST PARAMETERS

Name	Description	Type	Data type			
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK
Success

ITEMS

AssetNodeDto

POST /sca/asset_nodes/import

Tags: sca

Adds assets to the database.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices
- Manage Device Permissions (if the inventory includes devices with rights)

If an inventory asset node has a parent node Id that is null or empty, or does not reference an asset node in the inventory,
the new asset node is added as a child of the destination asset node (as specified by the request's destination node Id). Otherwise, new asset nodes are added as child nodes of their respective parents.

Device properties working copy information, device configuration information, and custom attributes are not imported.

For any non-required properties missing from the request, the API supplies default values appropriate for the asset.

If an asset node and asset provide different values for a property, the asset's value is used.

If an asset's custom unique identifier is provided in the inventory, the new asset retains this value.

Asset nodes and assets added to the database are both issued new unique Ids.

REQUEST BODY

application/x-www-form-urlencoded

AssetNodeImportDto

REQUEST PARAMETERS

Name	Description	Type	Data type

Name	Description	Type	Data type	
destination_node_id	The unique identifier of the destination node under which new asset nodes are created. A null or empty string indicates that new nodes are to be root nodes.	formData	*string* (uuid)	
inventory	The asset nodes and assets to be imported.	formData	*object*	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Information about the asset nodes that were imported.

The audit record contains these ResourceChangeRecordDtos for asset nodes and assets that were added to the database:

```
change_type: 'Create'
resource_type: 'AssetNodeDto'
id : <new asset node id>
attribute_name: 'id'
prior_value: <source asset node id>
current_value: <new asset node id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'id'
prior_value: <source asset id>
current_value: <new asset id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'path'
prior_value: <source asset path>
current_value: <new asset path>
```

ImportResponseDto

POST /sca/asset_nodes/merge

Merges asset information into the database, adding assets that do not exist.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database

- Manage Devices
- Manage Device Permissions (if the inventory includes devices with rights)

Merge uses an asset's unique identifier to find matching assets.

If an inventory asset matches an existing asset in the database, its properties are merged into the matching asset. Asset nodes are never merged.

A device's working copy is only updated when it is present in the inventory and its ConfigurationFiles is not null. It is not possible to clear a working copy when merging.

Device Rights cannot be removed by a merge operation, only added.

If an inventory asset does not match an existing asset in the database, a new asset and asset node are added to the database. The asset node is added as a child of the specified parent Id if it exists; otherwise, the asset node is added as a child of the destination asset node (as specified by the request's destination node Id).

The new asset node retains its unique Id value from the inventory. The new asset retains its unique Id and custom unique Id values from the inventory. If no custom unique Id is provided, a new value is issued.

If an asset node and asset provide different values for a property, the asset's value is used.

Unique ids of device sub-objects will be preserved.

This operation does not use optimistic locking.

Device properties working copy information, device configuration information, and custom attributes are not merged or used in new assets.

REQUEST BODY

application/x-www-form-urlencoded

AssetNodeMergeDto

REQUEST PARAMETERS

Name	Description	Type	Data type

Name	Description	Type	Data type	
destination_node_id	The unique identifier of the destination node under which asset nodes in the inventory that do not have a match are created. A null or empty string indicates that new nodes are to be root nodes.	formData	*string* (uuid)	
inventory	The asset nodes and assets to be merged.	formData	*object*	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
Information about the asset nodes that were merged.

The audit record contains these ResourceChangeRecordDtos for asset nodes and assets that were added to the database:

```
change_type: 'Create'
resource_type: 'AssetNodeDto'
id : <new asset node id>
attribute_name: 'id'
prior_value: <source asset node id>
current_value: <new asset node id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'id'
prior_value: <source asset id>
current_value: <new asset id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'path'
prior_value: <source asset path>
current_value: <new asset path>
```

MergeResponseDto

GET /sca/database_information

Gets information about the AcSELerator Database served by the API.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type			
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK

Information about the AcSELerator Database served by the API.

DatabaseInformationDto

POST /sca/device/configuration/version/baseline

Baseline the configuration working copy as a new configuration version.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationVersionBaselinePostDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
device_id	The unique identifier of the device.	formData	*string* (uuid)	**required**
version_number	The version number of the configuration version.	formData	*string*	**required**
comment	A user-provided comment about the configuration version.	formData	*string*	
state	The current state of the configuration version.	formData	*string*	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Information about a configuration version.

DeviceConfigurationVersionInformationDto

POST /sca/device/configuration/version/files

Creates a configuration version for a device from the provided configuration files.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationVersionFilesPostDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
device_id	The unique identifier of the device.	formData	*string* (uuid)	**required**
version_number	The version number of the configuration version.	formData	*string*	**required**
comment	A user-provided comment about the configuration version.	formData	*string*	
svn	The settings version number of the configuration version.	formData	*integer* (int32)	**required**
state	The current state of the configuration version.	formData	*string*	**required**

Name	Description	Type	Data type	
configuration_files	The configuration files of the configuration version.	formData	*object*	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Information about a configuration version.

DeviceConfigurationVersionInformationDto

POST /sca/device/configuration/version/replicate

Tags: sca

Replicates a device configuration version from another database.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationVersionReplicatePostDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
device_id	The unique identifier of the device.	formData	*string* (uuid)	**required**
version_number	The version number of the configuration version.	formData	*string*	**required**
comment	A user-provided comment about the configuration version.	formData	*string*	
svn	The settings version number of the configuration version.	formData	*integer* (int32)	**required**
configuration_files	The configuration files.	formData	*object*	**required**
state	The current state of the configuration version.	formData	*string*	**required**
created_by	The user that created the configuration version.	formData	*string*	**required**
created_at	The date and time at which the configuration version was created.	formData	*string* (date-time)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Information about a configuration version.

DeviceConfigurationVersionInformationDto

POST /sca/device/configuration/version/settings

Creates a configuration version for a device from the provided device settings.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationVersionSettingsPostDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
device_id	The unique identifier of the device.	formData	*string* (uuid)	**required**
version_number	The version number of the configuration version.	formData	*string*	**required**

Name	Description	Type	Data type	
comment	A user-provided comment about the configuration version.	formData	*string*	
svn	The settings version number of the configuration version.	formData	*integer* (int32)	**required**
state	The current state of the configuration version.	formData	*string*	**required**
device_settings	The device settings of the configuration version.	formData	*object*	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
Information about a configuration version.

DeviceConfigurationVersionInformationDto

GET /sca/device/configuration/version/{id}/files

Tags: sca

Gets a configuration version as files.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the configuration version to fetch.	path	*string* (uuid)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A configuration version as files.

DeviceConfigurationVersionFilesDto

GET /sca/device/configuration/version/{id}/settings

Gets a configuration version as settings.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the configuration version to fetch.	path	*string* (uuid)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
A configuration version as settings.

DeviceConfigurationVersionSettingsDto

GET /sca/device/{id}

Tags: sca

Gets a device.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices
- Manage Device Permissions (if device_parts_selector includes rights)

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
device_parts_selector	Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+). If the selection is unspecified, the Default selection is used. Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation. NamePlate properties are always included regardless of selection.	query	*string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }	
id	The unique identifier of the device to fetch.	path	*string* (uuid)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A device.

DeviceDto

PUT /sca/device/{id}

Tags: sca

Updates a device.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DevicePutDto

REQUEST PARAMETERS

Name	Description	Type	Data type

Name	Description	Type	Data type	
id	The unique identifier of the device to update.	path	*string* (uuid)	**required**
custom_asset_id	A user-defined identifier of the device. This value must be unique among all assets.	formData	*string*	
name	The name of the device. The name must be unique within the siblings of the device asset node.	formData	*string*	
description	A description of the device.	formData	*string*	
serial_number	The serial number of the device.	formData	*string*	
part_number	The part number of the device.	formData	*string*	
firmware_version	The firmware version of the device.	formData	*string*	
fid	The Firmware Identification value of the device.	formData	*string*	
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A device.

DeviceDto

PUT /sca/device/{id}/configuration/working_copy

Updates information about a configuration working copy.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationWorkingCopyInformationPutDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose configuration working copy information is to be updated.	path	*string* (uuid)	**required**

Name	Description	Type	Data type
comment	A user-provided comment about the working copy.	formData	*string*
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32) **required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" } **required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
Information about a configuration working copy.

DeviceConfigurationWorkingCopyInformationDto

GET /sca/device/{id}/configuration/working_copy/files

Gets a configuration working copy as files.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

If the device does not have an active working copy, the content of the latest version is returned with all information properties as null. Otherwise, if the device has no versions, all response properties are null.

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose configuration working copy is to be fetched.	path	*string* (uuid)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A configuration working copy as files.

DeviceConfigurationWorkingCopyFilesDto

PUT /sca/device/{id}/configuration/working_copy/files Replaces the working copy of a configuration with the provided configuration files.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

A null configuration_files property results in having no active working copy.
All fields are reset to null.

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationWorkingCopyFilesPutDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose configuration working copy is to be updated.	path	*string* (uuid)	**required**
comment	A user-provided comment about the working copy.	formData	*string*	
svn	The settings version number of the working copy.	formData	*integer* (int32)	
configuration_files	The configuration files of the working copy.	formData	*object*	
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**

Name	Description	Type	Data type	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
Information about a configuration working copy.

DeviceConfigurationWorkingCopyInformationDto

GET /sca/device/{id}/configuration/working_copy/settings

Tags: sca

Gets a configuration working copy as settings.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

If the device does not have an active working copy, the content of the latest version is returned with all information properties as null. Otherwise, if the device has no versions, all response properties are null.

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose configuration working copy is to be fetched.	path	*string* (uuid)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
A configuration working copy as settings.

DeviceConfigurationWorkingCopySettingsDto

PUT /sca/device/{id}/configuration/working_copy/settings

Replaces the working copy of a configuration with the provided device configuration settings.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

A null device_settings property results in having no active working copy.
All fields are reset to null.

REQUEST BODY

application/x-www-form-urlencoded

DeviceConfigurationWorkingCopySettingsPutDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose configuration working copy is to be updated.	path	*string* (uuid)	**required**
comment	A user-provided comment about the working copy.	formData	*string*	
svn	The settings version number.	formData	*integer* (int32)	
device_settings	The device settings.	formData	*object*	
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Information about a configuration working copy.

DeviceConfigurationWorkingCopyInformationDto

GET /sca/device/{id}/connection

Gets a device connection.

Tags: sca

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose connection is to be fetched.	path	*string* (uuid)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A device connection.

ConnectionDto

PUT /sca/device/{id}/connection

Tags: sca

Updates a device connection.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

ConnectionPutDto

REQUEST PARAMETERS

Name	Description	Type	Data type

Name	Description	Type	Data type	
id	The unique identifier of the device whose connection is to be updated.	path	*string* (uuid)	**required**
communication_channel	The device communication channel.	formData	*object*	
pass_through_port	The port the access script uses to establish a connection when using a pass-through server.	formData	*integer* (int32)	
termination_string	The string used to terminate a connection on a pass-through server. Used in connections for devices that act as a pass-through server.	formData	*string*	
access_script_name	The script used to access a port on a pass-through server.	formData	*string*	
terminate_script_name	The script used to terminate a connection on a pass-through server.	formData	*string*	
first_delay_time	The idle time required before accepting a termination string on a pass-through server. Used in connections for devices that act as a pass-through server.	formData	*integer* (int32)	
second_delay_time	The idle time required after receiving a termination string before the connection is terminated. Used in connections for devices that act as a pass-through server.	formData	*integer* (int32)	
legacy_mode_enabled	Indicates whether the device is in legacy mode. Used for automation controller devices.	formData	*boolean*	
description	A description of the connection.	formData	*string*	

Name	Description	Type	Data type	
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A device connection.

ConnectionDto

PUT /sca/device/{id}/password/{level}

Tags: sca

Updates a device password.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

DevicePasswordPutDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the device whose password is to be updated.	path	*string* (uuid)	**required**
level	The case-insensitive access level.	path	*string*	**required**
password	The value of the password.	formData	*string*	
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK
A device password.

DevicePasswordDto

POST /sca/device/{id}/right

Tags: sca

Creates a right.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices
- Manage Device Permissions

The device referenced in this operation must be a descendant of a security gateway.

REQUEST BODY

application/x-www-form-urlencoded

RightPostDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The identifier of the device on which the right is to be granted.	path	*string* (uuid)	**required**
permission_name	The case-insensitive name of the permission to be granted. Supported values include "Connect" and the Access Levels of the device.	formData	*string*	**required**
group_name	The name of the group to be granted the right.	formData	*string*	**required**

Name	Description	Type	Data type
Accept	Accept Header	header	*string* , x ∈ { "application/json" } **required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Represents a right on a device.

```
RightDto
```

POST /sca/device_types/get_list

Tags: sca

Gets a list of device types.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

```
DeviceTypeGetListDto
```

REQUEST PARAMETERS

Name	Description	Type	Data type			
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK

Success

ITEMS

DeviceTypeDto

POST /sca/devices/get_list

Tags: sca

Gets a list of devices.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices
- Manage Device Permissions (if device_parts_selector includes rights)

REQUEST BODY

application/x-www-form-urlencoded

```
DeviceGetListDto
```

REQUEST PARAMETERS

Name	Description	Type	Data type	
device_parts_selector	Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+). If the selection is unspecified, the Default selection is used. Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation. NamePlate properties are always included regardless of selection.	query	*string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }	
ids	An optional list of devices to get. If not provided, all devices are returned.	formData	*string[]* , multiple parameters (`ids=aaa&ids=bbb`)	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Success

ITEMS

DeviceDto

GET /sca/folder/{id}

Tags: sca

Gets a folder.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type			
id	The unique identifier of the folder to fetch.	path	*string* (uuid)	**required**		
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK

A folder.

FolderDto

PUT /sca/folder/{id}

Tags: sca

Updates a folder.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

FolderPutDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
id	The unique identifier of the folder to update.	path	*string* (uuid)	**required**
custom_asset_id	A user-defined identifier of the folder. This value must be unique among all assets.	formData	*string*	

Name	Description	Type	Data type		
name	The folder's name. The name must be unique within the siblings of the folder asset node.	formData	*string*		
description	A description of the folder.	formData	*string*		
folder_type_name	The folder's type.	formData	*string*		
optimistic_lock_field	Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.	formData	*integer* (int32)	**required**	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global** #/parameters/Accept

RESPONSES

application/json

200 OK
A folder.

FolderDto

POST /sca/folder_types/get_list

Tags: sca

Gets a list of folder types.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

FolderTypeGetListDto

REQUEST PARAMETERS

Name	Description	Type	Data type			
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK
Success

ITEMS

FolderTypeDto

POST /sca/folders/get_list

Tags: sca

Gets a list of folders.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/x-www-form-urlencoded

FolderGetListDto

REQUEST PARAMETERS

Name	Description	Type	Data type	
ids	An optional list of folders to get. If not provided, all folders are returned.	formData	*string[]* , multiple parameters (`ids=aaa&ids=bbb`)	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

Success

ITEMS

FolderDto

POST /sca/rights/delete_list

Tags: sca

Deletes a list of rights.

DESCRIPTION

Required Permissions(s):

- Log On
- Update Database
- Manage Devices
- Manage Device Permissions

REQUEST BODY

application/x-www-form-urlencoded

RightDeleteListDto

REQUEST PARAMETERS

Name	Description	Type	Data type			
ids	The ids of the rights to delete.	formData	*string[]* , multiple parameters (`ids=aaa&ids=bbb`)	**required**		
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

204 No Content

No Content

GET /sca/service_information

Tags: sca

Gets information about the API.

DESCRIPTION

Required Permissions(s):

- Log On

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type			
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK

Information about the API.

ServiceInformationDto

POST /sca/support_data

Tags: sca

Gets API service support data.

DESCRIPTION

Required Permissions(s):

- Log On

REQUEST BODY

application/json

SupportDataGetDto

REQUEST PARAMETERS

Name	Description	Type	Data type			
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**	**global**	#/parameters/Accept

RESPONSES

application/json

200 OK

Success

Stream

GET /sca/template/device

Tags: sca

Gets a device template.

DESCRIPTION

Required Permissions(s):

- Log On
- Manage Devices

REQUEST BODY

application/json

REQUEST PARAMETERS

Name	Description	Type	Data type
path	The path to the device template to fetch. The path has the form: /category/template Note: Only the 'SEL Templates' category is supported at this time.	query	*string*

Name	Description	Type	Data type	
device_parts_selector	Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+). If the selection is unspecified, the Default selection is used. Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation. NamePlate properties are always included regardless of selection.	query	*string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }	
Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required** **global** #/parameters/Accept

RESPONSES

application/json

200 OK

A device.

DeviceDto

Parameter definitions

Key	Name	Description	Type	Data type	
Accept	Accept	Accept Header	header	*string* , x ∈ { "application/json" }	**required**

Schema definitions

AssetNodeCloneDto: *object*

DESCRIPTION

Clones an asset node and its associated assets, including its child nodes.

PROPERTIES

destination_node_id: *string* **(uuid)**
The unique identifier of the destination for the cloned asset node. A null or empty string indicates that the cloned node is a root node.

id: *string* **(uuid)** **required**
The unique identifier of the asset node to clone.

AssetNodeDeleteListDto: *object*

DESCRIPTION

Deletes a list of asset nodes and associated assets, including their child nodes.

PROPERTIES

ids: *string[]* **required**
The unique identifiers of the asset nodes to delete.

> **ITEMS**
>
> *string*

AssetNodeDto: *object*

DESCRIPTION

A node in an asset hierarchy.

PROPERTIES

asset_id: *string* **(uuid)**
The unique identifier of the asset referenced by this asset node.

asset_subtype: *string*
The subtype of the asset. For devices, this is the name of the device type. For folders, this is the name of the folder type.

asset_type: *string* , x ∈ { "Device" , "Folder" }
The type of the asset.

custom_asset_id: *string*
A user-defined identifier of the asset.

id: *string* **(uuid)**
The unique identifier of the asset node.

name: *string*
The name of the asset.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

parent_node_id: *string* **(uuid)**
The unique identifier of the parent of this asset node. A null or empty string indicates that the asset node is a root node.

AssetNodeExportDto: *object*

DESCRIPTION

Exports a list of asset nodes and associated assets.

PROPERTIES

device_configuration_parts_selector: *string* , x ∈ { "WorkingCopy" , "Default" , "All" }
Selects which device configuration resources are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+).

If the selection is unspecified, the Default selection is used.

Default includes these parts: WorkingCopy.

device_parts_selector: *string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }
Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+).

If the selection is unspecified, the Default selection is used.

Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation.

NamePlate properties are always included regardless of selection.

export_parts_selector: *string* , x ∈ { "DeviceConfigurations" , "All" }
Selects which additional resources are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+).

If the selection is unspecified, no additional resources are exported.

ids: *string[]*
An optional list of the asset nodes to export. If not provided, all asset nodes are exported.

> **ITEMS**
>
> *string*

AssetNodeGetDto: *object*

DESCRIPTION

Gets an asset node.

PROPERTIES

id: *string* **(uuid)** **required**
The unique identifier of the asset node to fetch.

AssetNodeGetListDto: *object*

DESCRIPTION

Gets a list of asset nodes.

PROPERTIES

AssetNodeImportDto: *object*

DESCRIPTION

Adds assets to the database.

PROPERTIES

destination_node_id: *string* **(uuid)**
The unique identifier of the destination node under which new asset nodes are created. A null or empty string indicates that new nodes are to be root nodes.

inventory: *AssetNodeInventoryDto* **required**
The asset nodes and assets to be imported.

AssetNodeInventoryDto: *object*

DESCRIPTION

A list of asset nodes and associated assets.

PROPERTIES

asset_nodes: *object[]*
The asset nodes.

> **ITEMS**
>
> *AssetNodeDto*

devices: *object[]*
The devices.

> **ITEMS**
>
> *DeviceDto*

folders: *object[]*
The folders.

> **ITEMS**
>
> *FolderDto*

source_metadata: *SourceMetadata*
Metadata about the source of this inventory. This is supplied by the SCA Asset Node Export operation. This information should be preserved and supplied to operations that consume an inventory.

working_copies: *object[]*
The device configuration working copies as files.

> **ITEMS**
>
> *DeviceConfigurationWorkingCopyFilesDto*

AssetNodeMergeDto: *object*

DESCRIPTION

Merges asset information into the database, adding assets that do not exist.

PROPERTIES

destination_node_id: *string* **(uuid)**
The unique identifier of the destination node under which asset nodes in the inventory that do not have a match are created. A null or empty string indicates that new nodes are to be root nodes.

inventory: *AssetNodeInventoryDto* **required**
The asset nodes and assets to be merged.

AssetNodeMoveDto: *object*

DESCRIPTION

Moves an asset node.

PROPERTIES

destination_node_id: *string* **(uuid)**
The unique identifier of the destination node to which the requested node is to be moved. A null or empty string indicates that the moved node is to be a root node.

id: *string* **(uuid)** **required**
The unique identifier of the asset node to move.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

AssetNodePostDto: *object*

DESCRIPTION

Creates a new asset and assigns it to an asset node.

PROPERTIES

asset_subtype: *string* **required**
The subtype of the asset. For devices, this is the name of the device type. For folders, this is the name of the folder type.

asset_type: *string* , x ∈ { "Device" , "Folder" } **required**
The type of the asset.

custom_asset_id: *string*
A user-defined identifier of the asset. If a value is not provided, the API supplies a default for the asset. This value must be unique among all assets.

name: *string* **required**
The name of the asset.

The name must be unique among the siblings of the asset node.

parent_node_id: *string* **(uuid)**
The unique identifier of the parent of this asset node. A null or empty string indicates that the asset node is a root node.

AuditRecordDto: *object*

DESCRIPTION

Holds auditing information for a API operation.

PROPERTIES

generated_at: *string* **(date-time)**
The date and time at which the operation was performed.

resource_change_records: *object[]*
The resource change records.

> **ITEMS**
>
> *ResourceChangeRecordDto*

CloneResponseDto: *object*

DESCRIPTION

Information about the asset nodes that were cloned.

The audit record contains these ResourceChangeRecordDtos for asset nodes and assets that were added to the database:

```
change_type: 'Create'
resource_type: 'AssetNodeDto'
id : <new asset node id>
attribute_name: 'id'
prior_value: <source asset node id>
current_value: <new asset node id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'id'
prior_value: <source asset id>
current_value: <new asset id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'path'
prior_value: <source asset path>
current_value: <new asset path>
```

PROPERTIES

asset_nodes: *object[]*
The asset nodes.

> **ITEMS**
>
> *AssetNodeDto*

audit_record: *AuditRecordDto*
The audit record.

CommunicationChannelDto: *object*

DESCRIPTION

The communication parameters of a connection.

PROPERTIES

access_level: *string*
The Access Level associated with authenticating to the device. Used in connections that use Access Level for authentication.

at_string: *string*
The AT string to initialize the modem. Used in Modem connections.

bluetooth: *boolean*
Indicates whether the Bluetooth option is enabled for the connection. Used in Serial connections.

connection_type: *string* , x ∈ { "Serial" , "Modem" , "Network" }
The type of the connection.

credential_source: *string* , x ∈ { "Prompt" , "ActiveUser" , "TitledPasswords" , "AccessLevel" , "AccessLevelAndUserName" , "PromptOrTitledPassword" , "ActiveUserOrTitledPassword" }
The source of the credentials for a secured connection. Used in connections that require authentication.

credentials: *string*

The credentials used when connecting to the device. Used in connections that use Titled Passwords for authentication.

data_bits: *integer* **(int32)**
The number of data bits used in a transmission. Used in Serial connections.

data_speed: *integer* **(int32)** , x ∈ { "300" , "600" , "1200" , "2400" , "4800" , "9600" , "19200" , "38400" , "57600" , "115200" }
The rate of data used by the port. The data speed is the number of bits transferred per second. Used in Serial and Modem connections.

dtr: *boolean*
Indicates whether the Data Terminal Ready signal is enabled. Used in Serial connections.

file_transfer_option: *string* , x ∈ { "FTP" , "Telnet" , "RawTCP" , "SSH" , "Https" }
The method of file transfer. Used in Network connections.

ftp_port_number: *integer* **(int32)**
The port number for the FTP connection. Used in FTP Network connections.

ftp_proxy_port_number: *integer* **(int32)**
The port number a security gateway used to establish an FTP connection with this child device. Used in FTP Network connections when the device is the child of a security gateway.

host_ip_address: *string*
The IP address of the host device. Used in Network connections.

host_protocol: *string*
The engineering access protocol for the device. Used in Network connections when the device is the child of a security gateway.

https_port_number: *integer* **(int32)**
The port number for the HTTPS connection. Used in Network connections.

master_device_id: *integer* **(int32)**
The master identifier of the device. Used when using the Courier protocol.

optimistic_lock_field: *integer* **(int32)**

Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

parity: *string* , x ∈ { "NONE" , "EVEN" , "ODD" }
The parity-checking mode for the port. Used in Serial connections.

phone_number: *string*
The phone number to dial to access the device. Used in Modem connections.

port_number: *integer* **(int32)**
The network port number. Used in Network connections.

protocol: *string* , x ∈ { "Unspecified" , "ModbusRtu" , "ModbusTcp" , "Courier" }
The type of the protocol used to communicate with the device. Used for devices that support protocol selection.

protocol_interface: *string* , x ∈ { "Unspecified" , "IEC870" }
The protocol interface being used to communicate with the device. Used when using the Courier protocol.

rts: *boolean*
Indicates whether the Ready to Send signal is enabled. Used in Serial connections.

rtscts: *boolean*
Indicates whether the Ready To Send / Clear To Send signal is enabled. Used in Serial connections.

serial_interface: *string* , x ∈ { "Eia232" , "Eia485TwoWire" , "Eia422Or485FourWire" }
The type of serial connection. Used in Serial connections when the device is the child of a security gateway.

slave_device_id: *integer* **(int32)**
The slave identifier of the device. Used when using the Courier protocol.

stop_bits: *integer* **(int32)**
The number of stop bits for the port. Stop bits signal the end of a packet of information. Used in Serial connections.

unit_id: *integer* **(int32)**

The identifier of the unit. Used when using the Modbus protocol.

user_name: *string*
The user name used when authenticating to the device. Used in connections that use Access Level and User Name for authentication.

xon_xoff: *boolean*
Indicates whether software flow control is enabled for the connection. Used in Serial and Modem connections.

CommunicationChannelPutDto: *object*

DESCRIPTION

Updates the communication parameters of a connection.

PROPERTIES

access_level: *string*
The Access Level associated with authenticating to the device. Used in connections that use Access Level for authentication.

at_string: *string*
The AT string to initialize the modem. Used in Modem connections.

bluetooth: *boolean*
Indicates whether the Bluetooth option is enabled for the connection. Used in Serial connections.

connection_type: *string* , x ∈ { "Serial" , "Modem" , "Network" }
The type of the connection.

credential_source: *string* , x ∈ { "Prompt" , "ActiveUser" , "TitledPasswords" , "AccessLevel" , "AccessLevelAndUserName" , "PromptOrTitledPassword" , "ActiveUserOrTitledPassword" }
The source of the credentials for a secured connection. Used in connections that require authentication.

credentials: *string*
The credentials used when connecting to the device. Used in connections that use Titled Passwords for authentication.

data_bits: *integer* **(int32)**
The number of data bits used in a transmission. Used in Serial connections.

data_speed: *integer* **(int32)** , x ∈ { "300" , "600" , "1200" , "2400" , "4800" , "9600" , "19200" , "38400" , "57600" , "115200" }
The rate of data used by the port. The data speed is the number of bits transferred per second. Used in Serial and Modem connections.

dtr: *boolean*
Indicates whether the Data Terminal Ready signal is enabled. Used in Serial connections.

file_transfer_option: *string* , x ∈ { "FTP" , "Telnet" , "RawTCP" , "SSH" , "Https" }
The method of file transfer. Used in Network connections.

ftp_port_number: *integer* **(int32)**
The port number for the FTP connection. Used in FTP Network connections.

ftp_proxy_port_number: *integer* **(int32)**
The port number a security gateway used to establish an FTP connection with this child device. Used in FTP Network connections when the device is the child of a security gateway.

host_ip_address: *string*
The IP address of the host device. Used in Network connections.

host_protocol: *string*
The engineering access protocol for the device. Used in Network connections when the device is the child of a security gateway.

https_port_number: *integer* **(int32)**
The port number for the HTTPS connection. Used in Network connections.

master_device_id: *integer* **(int32)**
The master identifier of the device. Used when using the Courier protocol.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

parity: *string* , x ∈ { "NONE" , "EVEN" , "ODD" }
The parity-checking mode for the port. Used in Serial connections.

phone_number: *string*
The phone number to dial to access the device. Used in Modem connections.

port_number: *integer* **(int32)**
The network port number. Used in Network connections.

protocol: *string* , x ∈ { "Unspecified" , "ModbusRtu" , "ModbusTcp" , "Courier" }
The type of the protocol used to communicate with the device. Used for devices that support protocol selection.

protocol_interface: *string* , x ∈ { "Unspecified" , "IEC870" }
The protocol interface being used to communicate with the device. Used when using the Courier protocol.

rts: *boolean*
Indicates whether the Ready to Send signal is enabled. Used in Serial connections.

rtscts: *boolean*
Indicates whether the Ready To Send / Clear To Send signal is enabled. Used in Serial connections.

serial_interface: *string* , x ∈ { "Eia232" , "Eia485TwoWire" , "Eia422Or485FourWire" }
The type of serial connection. Used in Serial connections when the device is the child of a security gateway.

slave_device_id: *integer* **(int32)**
The slave identifier of the device. Used when using the Courier protocol.

stop_bits: *integer* **(int32)**
The number of stop bits for the port. Stop bits signal the end of a packet of information. Used in Serial connections.

unit_id: *integer* **(int32)**
The identifier of the unit. Used when using the Modbus protocol.

user_name: *string*

The user name used when authenticating to the device. Used in connections that use Access Level and User Name for authentication.

xon_xoff: *boolean*
Indicates whether software flow control is enabled for the connection. Used in Serial and Modem connections.

ConfigurationFilesDto: *object*

DESCRIPTION

The configuration files for a device.

PROPERTIES

misc_files: *object[]*
The misc files.

> **ITEMS**
>
> *FileDto*

settings_files: *object[]*
The settings files.

> **ITEMS**
>
> *FileDto*

ConnectionDto: *object*

DESCRIPTION

A device connection.

PROPERTIES

access_script_name: *string*

The script used to access a port on a pass-through server.

communication_channel: *CommunicationChannelDto*
The device communication channel.

description: *string*
A description of the connection.

first_delay_time: *integer* **(int32)**
The idle time required before accepting a termination string on a pass-through server. Used in connections for devices that act as a pass-through server.

id: *string* **(uuid)**
The unique identifier of the connection.

legacy_mode_enabled: *boolean*
Indicates whether the device is in legacy mode. Used for automation controller devices.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

pass_through_port: *integer* **(int32)**
The port the access script uses to establish a connection when using a pass-through server.

second_delay_time: *integer* **(int32)**
The idle time required after receiving a termination string before the connection is terminated. Used in connections for devices that act as a pass - through server.

terminate_script_name: *string*
The script used to terminate a connection on a pass-through server.

termination_string: *string*
The string used to terminate a connection on a pass-through server. Used in connections for devices that act as a pass-through server.

ConnectionGetDto: *object*

DESCRIPTION

Gets a device connection.

PROPERTIES

id: *string* **(uuid)** **required**
The unique identifier of the device whose connection is to be fetched.

ConnectionPutDto: *object*

DESCRIPTION

Updates a device connection.

PROPERTIES

access_script_name: *string*
The script used to access a port on a pass-through server.

communication_channel: *CommunicationChannelPutDto*
The device communication channel.

description: *string*
A description of the connection.

first_delay_time: *integer* **(int32)**
The idle time required before accepting a termination string on a pass-through server. Used in connections for devices that act as a pass-through server.

id: *string* **(uuid)** **required**

The unique identifier of the device whose connection is to be updated.

legacy_mode_enabled: *boolean*
Indicates whether the device is in legacy mode. Used for automation controller devices.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

pass_through_port: *integer* **(int32)**
The port the access script uses to establish a connection when using a pass-through server.

second_delay_time: *integer* **(int32)**
The idle time required after receiving a termination string before the connection is terminated. Used in connections for devices that act as a pass-through server.

terminate_script_name: *string*
The script used to terminate a connection on a pass-through server.

termination_string: *string*
The string used to terminate a connection on a pass-through server. Used in connections for devices that act as a pass-through server.

CustomAttributeDto: *object*

DESCRIPTION

A custom attribute.

PROPERTIES

name: *string*
The name of the custom attribute.

value: *string*

The value of the custom attribute.

DatabaseInformationDto: *object*

DESCRIPTION

Information about the AcSELerator Database served by the API.

PROPERTIES

database_application_version: *string*
The version of the SEL Configuration API application supported by the database.

database_name: *string*
The name of the database.

database_schema_version: *string*
The version of the AcSELerator Database schema.

database_size_bytes: *integer* **(int64)**
The database size in bytes.

database_size_pretty: *string*
The formatted database size (e.g., 20MB, 3741KB, 1GB).

DatabaseInformationGetDto: *object*

DESCRIPTION

Gets information about the AcSELerator Database served by the API.

PROPERTIES

DeviceConfigurationVersionBaselinePostDto: *object*

DESCRIPTION

Baseline the configuration working copy as a new configuration version.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

device_id: *string* **(uuid)** **required**
The unique identifier of the device.

state: *string* **required**
The current state of the configuration version.

version_number: *string* **required**
The version number of the configuration version.

DeviceConfigurationVersionFilesDto: *object*

DESCRIPTION

A configuration version as files.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

created_at: *string* **(date-time)**
The date and time at which the configuration version was created.

created_by: *string*
The user that created the configuration version.

device_id: *string* **(uuid)**
The unique identifier of the device with this configuration version.

id: *string* **(uuid)**
The unique identifier of the configuration version.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

state: *string*
The current state of the configuration version.

svn: *integer* **(int32)**
The settings version number of the configuration version.

version_number: *string*
The version number of the configuration version.

configuration_files: *ConfigurationFilesDto*
The configuration files.

DeviceConfigurationVersionFilesGetDto: *object*

DESCRIPTION

Gets a configuration version as files.

PROPERTIES

id: *string* **(uuid)** **required**

The unique identifier of the configuration version to fetch.

DeviceConfigurationVersionFilesPostDto: *object*

DESCRIPTION

Creates a configuration version for a device from the provided configuration files.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

configuration_files: *ConfigurationFilesDto* **required**
The configuration files of the configuration version.

device_id: *string* **(uuid)** **required**
The unique identifier of the device.

state: *string* **required**
The current state of the configuration version.

svn: *integer* **(int32)** **required**
The settings version number of the configuration version.

version_number: *string* **required**
The version number of the configuration version.

DeviceConfigurationVersionInformationDto: *object*

DESCRIPTION

Information about a configuration version.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

created_at: *string* **(date-time)**
The date and time at which the configuration version was created.

created_by: *string*
The user that created the configuration version.

device_id: *string* **(uuid)**
The unique identifier of the device with this configuration version.

id: *string* **(uuid)**
The unique identifier of the configuration version.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

state: *string*
The current state of the configuration version.

svn: *integer* **(int32)**
The settings version number of the configuration version.

version_number: *string*
The version number of the configuration version.

DeviceConfigurationVersionReplicatePostDto: *object*

DESCRIPTION

Replicates a device configuration version from another database.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

configuration_files: *ConfigurationFilesDto* **required**
The configuration files.

created_at: *string* **(date-time)** **required**
The date and time at which the configuration version was created.

created_by: *string* **required**
The user that created the configuration version.

device_id: *string* **(uuid)** **required**
The unique identifier of the device.

state: *string* **required**
The current state of the configuration version.

svn: *integer* **(int32)** **required**
The settings version number of the configuration version.

version_number: *string* **required**
The version number of the configuration version.

DeviceConfigurationVersionSettingsDto: *object*

DESCRIPTION

A configuration version as settings.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

created_at: *string* **(date-time)**
The date and time at which the configuration version was created.

created_by: *string*
The user that created the configuration version.

device_id: *string* **(uuid)**
The unique identifier of the device with this configuration version.

id: *string* **(uuid)**
The unique identifier of the configuration version.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

state: *string*
The current state of the configuration version.

svn: *integer* **(int32)**
The settings version number of the configuration version.

version_number: *string*
The version number of the configuration version.

device_settings: *DeviceSettingsDto*
The device settings.

DeviceConfigurationVersionSettingsGetDto: *object*

DESCRIPTION

Gets a configuration version as settings.

PROPERTIES

id: *string* **(uuid)** **required**
The unique identifier of the configuration version to fetch.

DeviceConfigurationVersionSettingsPostDto: *object*

DESCRIPTION

Creates a configuration version for a device from the provided device settings.

PROPERTIES

comment: *string*
A user-provided comment about the configuration version.

device_id: *string* **(uuid)** **required**
The unique identifier of the device.

device_settings: *DeviceSettingsDto* **required**
The device settings of the configuration version.

state: *string* **required**
The current state of the configuration version.

svn: *integer* **(int32)** **required**
The settings version number of the configuration version.

version_number: *string* **required**
The version number of the configuration version.

DeviceConfigurationWorkingCopyFilesDto: *object*

DESCRIPTION

A configuration working copy as files.

PROPERTIES

comment: *string*
A user-provided comment about the working copy.

created_at: *string* **(date-time)**
The date and time at which the configuration working copy was created.

created_by: *string*
The user that created the configuration working copy.

id: *string* **(uuid)**
The unique identifier of the device with this configuration working copy.

last_modified_at: *string* **(date-time)**
The date and time at which the configuration working copy was last modified.

last_modified_by: *string*
The user that last modified the configuration working copy.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

svn: *integer* **(int32)**

The settings version number.

configuration_files: *ConfigurationFilesDto*
The working copy configuration files.

DeviceConfigurationWorkingCopyFilesGetDto: *object*

DESCRIPTION

Gets a configuration working copy as files.

PROPERTIES

id: *string* **(uuid)** **required**
The unique identifier of the device whose configuration working copy is to be fetched.

DeviceConfigurationWorkingCopyFilesPutDto: *object*

DESCRIPTION

Replaces the working copy of a configuration with the provided configuration files.

PROPERTIES

comment: *string*
A user-provided comment about the working copy.

configuration_files: *ConfigurationFilesDto*
The configuration files of the working copy.

id: *string* **(uuid)** **required**
The unique identifier of the device whose configuration working copy is to be updated.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

svn: *integer* **(int32)**
The settings version number of the working copy.

DeviceConfigurationWorkingCopyInformationDto: *object*

DESCRIPTION

Information about a configuration working copy.

PROPERTIES

comment: *string*
A user-provided comment about the working copy.

created_at: *string* **(date-time)**
The date and time at which the configuration working copy was created.

created_by: *string*
The user that created the configuration working copy.

id: *string* **(uuid)**
The unique identifier of the device with this configuration working copy.

last_modified_at: *string* **(date-time)**
The date and time at which the configuration working copy was last modified.

last_modified_by: *string*
The user that last modified the configuration working copy.

optimistic_lock_field: *integer* **(int32)**

Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

svn: *integer* **(int32)**
The settings version number.

DeviceConfigurationWorkingCopyInformationPutDto: *object*

DESCRIPTION

Updates information about a configuration working copy.

PROPERTIES

comment: *string*
A user-provided comment about the working copy.

id: *string* **(uuid)** **required**
The unique identifier of the device whose configuration working copy information is to be updated.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

DeviceConfigurationWorkingCopySettingsDto: *object*

DESCRIPTION

A configuration working copy as settings.

PROPERTIES

comment: *string*

A user-provided comment about the working copy.

created_at: *string* **(date-time)**
The date and time at which the configuration working copy was created.

created_by: *string*
The user that created the configuration working copy.

id: *string* **(uuid)**
The unique identifier of the device with this configuration working copy.

last_modified_at: *string* **(date-time)**
The date and time at which the configuration working copy was last modified.

last_modified_by: *string*
The user that last modified the configuration working copy.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

svn: *integer* **(int32)**
The settings version number.

device_settings: *DeviceSettingsDto*
The device configuration settings.

DeviceConfigurationWorkingCopySettingsGetDto: *object*

DESCRIPTION

Gets a configuration working copy as settings.

PROPERTIES

id: *string* **(uuid)** **required**
The unique identifier of the device whose configuration working copy is to be fetched.

DeviceConfigurationWorkingCopySettingsPutDto: *object*

DESCRIPTION

Replaces the working copy of a configuration with the provided device configuration settings.

PROPERTIES

comment: *string*
A user-provided comment about the working copy.

device_settings: *DeviceSettingsDto*
The device settings.

id: *string* **(uuid)** **required**
The unique identifier of the device whose configuration working copy is to be updated.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

svn: *integer* **(int32)**
The settings version number.

DeviceDto: *object*

DESCRIPTION

A device.

PROPERTIES

configuration_versions: *object[]*
Information about the configuration versions of the device.

> **ITEMS**
>
> *DeviceConfigurationVersionInformationDto*

connection: *ConnectionDto*
The device connection.

custom_asset_id: *string*
A user-defined identifier of the device.

custom_attributes: *object[]*
The custom attributes of the device.

> **ITEMS**
>
> *CustomAttributeDto*

description: *string*
A description of the device.

device_type_name: *string*
The type of the device.

fid: *string*
The Firmware Identification value of the device.

firmware_version: *string*
The firmware version of the device.

id: *string* **(uuid)**
The unique identifier of the device.

name: *string*
The name of the device.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

part_number: *string*
The part number of the device.

passwords: *object[]*
The passwords of the device. Passwords are ordered as displayed in SEL software.

> **ITEMS**
>
> *DevicePasswordDto*

path: *string*
The asset node path of the device.

rights: *object[]*
The rights of the device.

> **ITEMS**
>
> *RightDto*

serial_number: *string*
The serial number of the device.

working_copy: *DeviceConfigurationWorkingCopyInformationDto*
Information about the configuration working copy of the device.

DeviceGetDto: *object*

DESCRIPTION

Gets a device.

PROPERTIES

device_parts_selector: *string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }
Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+).

If the selection is unspecified, the Default selection is used.

Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation.

NamePlate properties are always included regardless of selection.

id: *string* **(uuid)** **required**
The unique identifier of the device to fetch.

DeviceGetListDto: *object*

DESCRIPTION

Gets a list of devices.

PROPERTIES

device_parts_selector: *string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }
Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+).

If the selection is unspecified, the Default selection is used.

Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation.

NamePlate properties are always included regardless of selection.

ids: *string[]*
An optional list of devices to get. If not provided, all devices are returned.

> **ITEMS**
>
> *string*

DevicePasswordDto: *object*

DESCRIPTION

A device password.

PROPERTIES

id: *string* **(uuid)**
The unique identifier of the device with this password.

level: *string*
The access level of the password.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

password: *string*
The value of the password.

DevicePasswordPutDto: *object*

DESCRIPTION

Updates a device password.

PROPERTIES

id: *string* **(uuid)** **required**

The unique identifier of the device whose password is to be updated.

level: *string* **required**
The case-insensitive access level.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

password: *string*
The value of the password.

DevicePutDto: *object*

DESCRIPTION

Updates a device.

PROPERTIES

custom_asset_id: *string*
A user-defined identifier of the device. This value must be unique among all assets.

description: *string*
A description of the device.

fid: *string*
The Firmware Identification value of the device.

firmware_version: *string*
The firmware version of the device.

id: *string* **(uuid)** **required**
The unique identifier of the device to update.

name: *string*
The name of the device.

The name must be unique within the siblings of the device asset node.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

part_number: *string*
The part number of the device.

serial_number: *string*
The serial number of the device.

DeviceSettingsDto: *object*

DESCRIPTION

The settings for a device.

PROPERTIES

misc_files: *object[]*
The misc files.

> **ITEMS**
>
> *FileDto*

settings_groups: *object[]*
The settings groups.

> **ITEMS**
>
> *SettingsGroupDto*

DeviceTypeDto: *object*

DESCRIPTION

A type of device.

PROPERTIES

device_type_class_name: *string* , x ∈ { "Default" , "PassThroughServer" , "CommProc" , "Rtac" , "SecurityGateway" , "SecureDevice" }
The class of a device type.

is_sel_provided: *boolean*
Indicates whether this device type is pre-configured in the database by SEL. SEL-provided device types cannot be modified or deleted.

name: *string*
The name of the device type.

DeviceTypeGetListDto: *object*

DESCRIPTION

Gets a list of device types.

PROPERTIES

FileDto: *object*

DESCRIPTION

A file.

PROPERTIES

content: *string* **(byte)**
The file's content.

name: *string*
The file's name.

FolderDto: *object*

DESCRIPTION

A folder.

PROPERTIES

custom_asset_id: *string*
A user-defined identifier of the folder.

custom_attributes: *object[]*
The custom attributes of the folder.

> **ITEMS**
>
> *CustomAttributeDto*

description: *string*
A description of the folder.

folder_type_name: *string*
The folder's type.

id: *string* **(uuid)**
The unique identifier of the folder.

name: *string*
The folder's name.

optimistic_lock_field: *integer* **(int32)**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

path: *string*
The folder's asset node path.

FolderGetDto: *object*

DESCRIPTION

Gets a folder.

PROPERTIES

id: *string* **(uuid)** **required**
The unique identifier of the folder to fetch.

FolderGetListDto: *object*

DESCRIPTION

Gets a list of folders.

PROPERTIES

ids: *string[]*
An optional list of folders to get. If not provided, all folders are returned.

> **ITEMS**
>
> *string*

FolderPutDto: *object*

DESCRIPTION

Updates a folder.

PROPERTIES

custom_asset_id: *string*
A user-defined identifier of the folder. This value must be unique among all assets.

description: *string*
A description of the folder.

folder_type_name: *string*
The folder's type.

id: *string* **(uuid)** **required**
The unique identifier of the folder to update.

name: *string*
The folder's name.

The name must be unique within the siblings of the folder asset node.

optimistic_lock_field: *integer* **(int32)** **required**
Used in the optimistic locking system to prevent stale data from being written to the database. See the Optimistic Locking section in the API Programmer's Reference for more information.

FolderTypeDto: *object*

DESCRIPTION

A type of folder.

PROPERTIES

name: *string*
The name of the folder type.

FolderTypeGetListDto: *object*

DESCRIPTION

Gets a list of folder types.

PROPERTIES

ImportResponseDto: *object*

DESCRIPTION

Information about the asset nodes that were imported.

The audit record contains these ResourceChangeRecordDtos for asset nodes and assets that were added to the database:

```
change_type: 'Create'
resource_type: 'AssetNodeDto'
id : <new asset node id>
attribute_name: 'id'
prior_value: <source asset node id>
current_value: <new asset node id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'id'
prior_value: <source asset id>
current_value: <new asset id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'path'
prior_value: <source asset path>
current_value: <new asset path>
```

PROPERTIES

asset_nodes: *object[]*
The asset nodes.

> **ITEMS**
>
> *AssetNodeDto*

audit_record: *AuditRecordDto*
The audit record.

List_AssetNodeDto_: *object[]*

ITEMS

AssetNodeDto

List_DeviceDto_: *object[]*

ITEMS

DeviceDto

List_DeviceTypeDto_: *object[]*

ITEMS

DeviceTypeDto

List_FolderDto_: *object[]*

ITEMS

FolderDto

List_FolderTypeDto_: *object[]*

ITEMS

FolderTypeDto

MergeResponseDto: *object*

DESCRIPTION

Information about the asset nodes that were merged.

The audit record contains these ResourceChangeRecordDtos for asset nodes and assets that were added to the database:

```
change_type: 'Create'
resource_type: 'AssetNodeDto'
id : <new asset node id>
attribute_name: 'id'
prior_value: <source asset node id>
current_value: <new asset node id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'id'
prior_value: <source asset id>
current_value: <new asset id>

change_type: 'Create'
resource_type: 'DeviceDto' or 'FolderDto'
id : <new asset id>
attribute_name: 'path'
prior_value: <source asset path>
current_value: <new asset path>
```

PROPERTIES

added_asset_nodes: *object[]*
The asset nodes that were added.

> **ITEMS**
>
> *AssetNodeDto*

audit_record: *AuditRecordDto*
The audit record.

updated_asset_nodes: *object[]*
The asset nodes that were updated.

> **ITEMS**
>
> *AssetNodeDto*

Object: *object*

DESCRIPTION

Object

PROPERTIES

ResourceChangeRecordDto: *object*

DESCRIPTION

Identifies a change to a resource.

PROPERTIES

attribute_name: *string*
The name of the changed attribute.

change_type: *string* , x ∈ { "Unknown" , "Create" , "Update" , "Delete" }
The type of the change.

current_value: *string*
The current value of the changed attribute.

id: *string* **(uuid)**
The unique identifier of the resource that was changed.

prior_value: *string*
The prior value of the changed attribute.

resource_type: *string*

The type of the resource that was changed. This is the type of the DTO returned by the resource's GET operation as defined in the API documentation. For example, 'AssetNodeDto' for a modified Asset Node.

RightDeleteListDto: *object*

DESCRIPTION

Deletes a list of rights.

PROPERTIES

ids: *string[]* **required**
The ids of the rights to delete.

> **ITEMS**
>
> *string*

RightDto: *object*

DESCRIPTION

Represents a right on a device.

PROPERTIES

group_name: *string*
The name of the group granted the right.

id: *string* **(uuid)**
The identifier of the right.

permission_name: *string*
The name of the permission granted by the right.

RightPostDto: *object*

DESCRIPTION

Creates a right.

PROPERTIES

group_name: *string* **required**
The name of the group to be granted the right.

id: *string* **(uuid)** **required**
The identifier of the device on which the right is to be granted.

permission_name: *string* **required**
The case-insensitive name of the permission to be granted. Supported values include "Connect" and the Access Levels of the device.

ServiceInformationDto: *object*

DESCRIPTION

Information about the API.

PROPERTIES

interface_version: *string*
The interface version of the SEL Configuration API.

product_version: *string*
The product version of the SEL Configuration API.

ServiceInformationGetDto: *object*

DESCRIPTION

Gets information about the API.

PROPERTIES

SettingsGroupDto: *object*

DESCRIPTION

A group of settings.

PROPERTIES

name: *string*
The settings group name.

settings: *object[]*
The group's settings.

> **ITEMS**
>
> *SettingValueDto*

SettingValueDto: *object*

DESCRIPTION

A single setting.

PROPERTIES

comment: *string*
The setting comment.

name: *string*
The setting name.

value: *string*
The setting value.

SourceMetadata: *object*

DESCRIPTION

Metadata about the source of an operation.

PROPERTIES

database_application_version: *string*
The version of the SEL Configuration API application supported by the database.

database_schema_version: *string*
The version of the AcSELerator Database schema.

interface_version: *string*
The interface version of the SEL Configuration API.

Stream: *object*

DESCRIPTION

Stream

PROPERTIES

SupportDataGetDto: *object*

DESCRIPTION

Gets API service support data.

PROPERTIES

TemplateDeviceGetDto: *object*

DESCRIPTION

Gets a device template.

PROPERTIES

device_parts_selector: *string* , x ∈ { "NamePlate" , "Connection" , "ConfigurationWorkingCopyInformation" , "ConfigurationVersionsInformation" , "Default" , "Passwords" , "Rights" , "All" }
Selects which parts of the device are included in the response. Multiple selections can be separated by a comma(,), vertical bar(|), or plus sign(+).

If the selection is unspecified, the Default selection is used.

Default includes these parts: NamePlate, Connection, ConfigurationWorkingCopyInformation, and ConfigurationVersionsInformation.

NamePlate properties are always included regardless of selection.

path: *string*
The path to the device template to fetch. The path has the form: /category/template Note: Only the 'SEL Templates' category is supported at this time.