Roger Access Control System

Integration of RACS 5 Through Web Services API

Software version: 2.0.6

Document version: Rev. A

INTEGRATION CONCEPT

The RACS 5 system can be integrated with other systems via RACS Integration Server (RIS). The RACS Integration Server provides access to RACS 5 databases (Event Log, Access Users Management, Access Credentials and Authorizations Management), allow to execute remote command and allow synchronize settings between software and hardware.
Integration is based on Windows Communication Foundation (WCF) framework. WCF implements modern industry standards for Web services interoperability, so it could be used at any platforms that support Web services technology.

CONFIGURATION OF INTEGRATION SERVER

1. Install and configure VISO application.
2. Install RogerSVC software package. During installation select RACS Communication Server, RACS Licensing Server and RACS Integration Server components.
3. Configure RACS Communication Server (use RACS Services Manager for that). Service address, port and VISO database must be specified. Restart service after that.
4. Configure RACS Licensing Server (use RACS Services Manager for that). Service address and port must be specified. Licensing file, that allow work with Integration server must be loaded. Restart service after that.
5. Configure RACS Integration Server endpoints addresses (use RACS Services Manager for that). By default, the services are listening for requests on the following local addresses:
    - http://127.0.0.1:8892/SessionManagement
    - http://127.0.0.1:8892/ConfigurationQuery
    - http://127.0.0.1:8892/EventLogManagement
    - http://127.0.0.1:8892/SystemSynchronization
    - http://127.0.0.1:8892/Integration
    - http://127.0.0.1:8892/Communication
    - http://127.0.0.1:8892/SystemReporting

6. For testing server functionality you can use Roger.Racs.IntegrationServer.TestClient.exe sample application (.NET 4). To log in use VISO Operator login and password. Source code of this sample application is available in RogerSVC main catalog in sub catalog Roger.Racs.IntegrationServer.TestClient.

WEB SERVICES

General rules

To execute any server functionality you must be connected to RACS Integration Server. For that purpose you should use Connect method from SessionManagement Service. It returns current session token, that must be apply as the last parameter of any other functions.

Common Types

This section contains common types description.

Object Types

- 3 - Operator ID (Remote Command)
- 1025 – Access Credential
- 1026 – Access User Group
- 1027 – Access User Person
- 1028 – Access User Asset
- 1029 – Access User Visitor
- 1030 – Access Zone
- 1031 – Access Point
- 1032 – Authentication Policy
- 1034 – Calendar
- 1036 – Schedule
- 1039 – Access Rights
- 1042 – Access Door
- 1043 – Credential Data Input
- 1045 – Time Attendance Mode
- 1049 – Input
- 1050 – Output
- 1069 – Alarm Zone
- 1076 – Function Key
- 1077 – Automation Node
- 1078 – Access Door Group
- 1079 – Automation Node Group
- 1082 – Control Command
- 1083 – Power Supply
- 1084 – Main Board
- 16384 – Call Type

Session Management

This service allow you to connect and disconnect to/from RACS Integration Server.

Connect

Allow to log in to RACS Integration Server. It must be executed before use of any other function. It should be executed on external system start up.

Method signature:

```
Guid Connect(string login, string password)
```

Where,

- login - VISO Operator login
- password - VISO Operator password

Result:

If valid authorization data are specified, result is current session token – it's required to apply this token to any other function as last parameter.

If invalid authorization data are specified, result is empty guid.

Disconnect

Allow to log out from RACS Integration Server. This operation is not required but is recommended. Function should be executed on external system close down.

Method signature:

```
bool Disconnect(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

GetOperatorBySession

Returns the logged in operator.

Method signature:

```
OperatorData GetOperatorBySession(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Logged in operator.

GetOperatorSettingsBySession

Returns the logged in operator settings.

Method signature:

```
OperatorSettingsData GetOperatorSettingsBySession(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Logged in operator settings.

Configuration Query

This service allow to query RACS Integration server for specified data. It also contains methods that allow to manage Access User Person data (credentials, authentication factors and permissions).

GetCredentials

Get all Access Credentials existing in the system.

Method signature:

```
IEnumerable<CredentialData> GetCredentials(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials existing in the system.

GetCredentialsNotAssignedToUsers

Get Access Credentials which are not assigned to users.

Method signature:

```
IEnumerable<CredentialData> GetCredentialsNotAssignedToUsers(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials not assigned to users.

GetCredentialsBySession

Get Access Credentials belongs to User linked with current logged Operator.

Method signature:

```
IEnumerable<CredentialData> GetCredentialsBySession(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials belongs to User linked with current logged Operator. Empty list is returned if there is no link between User and Operator.

GetFactorTypes

Get all Authentication Factor Types existing in the system.

Method signature:

```
IEnumerable<FactorTypeData> GetFactorTypes(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Authentication Factor Types existing in the system.

GetAllAuthenticationFactors

Get all Authentication Factors all cards.

Method signature:

```
IEnumerable<FactorData> GetAllAuthenticationFactors(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Authentication Factors.

GetAuthenticationFactorsFromCardBox

Get all Authentication Factors all cards that are in the Card Box.

Method signature:

```
IEnumerable<FactorData> GetAuthenticationFactorsFromCardBox(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Authentication Factors that are in the Card Box.

GetOperatorNames

Get all Authentication Factor Types existing in the system.

Method signature:

```
IEnumerable<OperatorData> GetOperatorNames(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Operators existing in the system.

GetControllers

Get all Access Controllers existing in the system.

Method signature:

```
IEnumerable<AccessControllerData> GetControllers(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Controllers existing in the system.

GetDoors

Get all Access Doors existing in the system.

Method signature:

```
IEnumerable<AccessDoorData> GetDoors(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Doors existing in the system.

GetPoints

Get all Access Points existing in the system.

Method signature:

```
IEnumerable<AccessPointData> GetPoints(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Points existing in the system.

GetRights

Get all Access Rights existing in the system.

Method signature:

```
IEnumerable<AccessRightsData> GetRights(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights existing in the system.

GetRightsGroups

Get all Access Rights Groups existing in the system.

Method signature:

```
IEnumerable<AccessRightsGroupData> GetRightsGroups(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights Groups existing in the system.

GetAssets

Get all Access User Assets existing in the system.

Method signature:

```
IEnumerable<AssetData> GetAssets(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access User Assets existing in the system.

GetGroups

Get all Access User Groups existing in the system.

Method signature:

```
IEnumerable<GroupData> GetGroups(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access User Groups existing in the system.

GetPersons

Get all Access User Persons existing in the system.

Method signature:

```
IEnumerable<PersonData> GetPersons(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access User Persons existing in the system.

GetPersonCredentialFactorIdsByCardNumber

Get Access User Person ID for giving card number.

Method signature:

```
IEnumerable<PersonCredentialFactorData>
GetPersonCredentialFactorIdsByCardNumber(string cardNumber, Guid sessionToken)
```

Where,

- cardNumber – Factor value
- sessionToken – Current session token, obtained by Connect method execution

Result:

If a factor is found for the given card number then data is returned.

If not found then empty list returned.

GetVisitors

Get all Access User Visitor existing in the system.

Method signature:

```
IEnumerable<VisitorData> GetVisitors(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access User Visitors existing in the system.

GetVisitZones

Get all Visit Zones existing in the system.

Method signature:

```
IEnumerable<VisitZoneData> GetVisitZones(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Visit Zones existing in the system.

GetAccessZones

Get all Access Zones existing in the system.

Method signature:

```
IEnumerable<AccessZoneData> GetAccessZones(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Zones existing in the system.

GetAlarmZones

Get all Alarm Zones existing in the system.

Method signature:

```
IEnumerable<AlarmZoneData> GetAlarmZones(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Alarm Zones existing in the system.

GetAutomationPoints

Get all Automation Points existing in the system.

Method signature:

```
IEnumerable<AutomationPointData> GetAutomationPoints(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Automation Points existing in the system.

GetControlCommands

Get all Control Commands existing in the system.

Method signature:

```
IEnumerable<ControlCommandData> GetControlCommands(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Control Commands existing in the system.

GetInputs

Get all Inputs existing in the system.

Method signature:

```
IEnumerable<InputData> GetInputs(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Inputs existing in the system.

GetFunctionKeys

Get all Function Keys existing in the system.

Method signature:

```
IEnumerable<FunctionKeyData> GetFunctionKeys(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Function Keys existing in the system.

GetOutputs

Get all Outputs existing in the system.

Method signature:

```
IEnumerable<OutputData> GetOutputs(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Outputs existing in the system.

GetCommunicationServers

Get all Communication Servers existing in the system.

Method signature:

```
IEnumerable<CommunicationServerData> GetCommunicationServers(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Communication Servers existing in the system.

GetCommunicationServerById

Get Communication Server by its ID.

Method signature:

```
CommunicationServerData GetCommunicationServerById(int communicationServerId,
Guid sessionToken)
```

Where,

- communicationServerId – ID of the requested Communication Server
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Communication Server exist in the system, then CommunicationServerData is returned.

If Communication Server not exist in the system, then NULL is returned.

GetNetworks

Get all Networks existing in the system.

Method signature:

```
IEnumerable<NetworkData> GetNetworks(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Networks existing in the system.

GetGlobalCommands

Get all Global Commands existing in the system.

Method signature:

```
IEnumerable<GlobalCommandData> GetGlobalCommands(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Global Commands existing in the system.

GetTimeAttendanceModes

Get all Time Attendance Modes existing in the system.

Method signature:

```
IEnumerable<TimeAttendanceModeData> GetTimeAttendanceModes(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Time Attendance Modes existing in the system.

GetCredentialDataInputs

Get all Credential Data Inputs existing in the system.

Method signature:

```
IEnumerable<CredentialDataInputData> GetCredentialDataInputs(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Credential Data Inputs existing in the system.

GetAttendanceZones

Get all AttendanceZones existing in the system.

Method signature:

```
IEnumerable<AttendanceZoneData> GetAttendanceZones(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Attendance Zones existing in the system.

GetDevices

Get all physical devices existing in the system.

Method signature:

```
IEnumerable<DeviceData> GetDevices(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Devices existing in the system.

GetDeviceObjects

Get physical devices objects.

Method signature:

```
IEnumerable<DeviceObjectData> GetDeviceObjects(int deviceId, Guid sessionToken)
```

Where,

- deviceId – ID of Device
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of physical device objects.

GetDeviceObjectById

Get device object by ID.

Method signature:

```
DeviceObjectData GetDeviceObjectById(int deviceObjectId, Guid sessionToken)
```

Where,

- deviceObjectId – ID of Device Object
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Device Object exist in the system, then device object data is returned.

If Device Object not exist in the system, then NULL is returned.

GetCredentialById

Get Access Credential by its ID.

Method signature:

```
CredentialData GetCredentialById(int credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of the requested credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Credential exist in the system, then CredentialData is returned.

If Access Credential not exist in the system, then NULL is returned.

GetCredentialTemplate

Get default values of Access Credential object properties.

Method signature:

```
IEnumerable<CredentialData> GetCredentialTemplate(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Default credential data is returned.

GetCredentialRights

Get Access Rights identifiers assigned to specified Access Credential.

Method signature:

```
IEnumerable<int> GetCredentialRights(int credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights identifiers assigned to specified Access Credential.

GetCredentialRightsView

Get Access Rights identifiers assigned to specified Access Credential.

Method signature:

```
IEnumerable<AssignedAccessRightsData> GetCredentialRightsView(int credentialId,
Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights identifiers assigned to specified Access Credential.

GetCredentialRightsGroups

Get Access Rights Groups identifiers assigned to specified Access Credential.

Method signature:

```
IEnumerable<int> GetCredentialRightsGroups(int credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights Groups identifiers assigned to specified Access Credential.

GetCredentialRightsGroupsView

Get Access Rights Groups identifiers assigned to specified Access Credential.

Method signature:

```
IEnumerable<AssignedAccessRightsGroupData> GetCredentialRightsGroupsView(int
credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights Groups identifiers assigned to specified Access Credential.

GetAuthenticationFactorsByCredentialId

Get Authentication Factors assigned to specified Access Credential.

Method signature:

```
IEnumerable<FactorData> GetAuthenticationFactorsByCredentialId(int credentialId,
Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Authentication Factors assigned to specified Access Credential.

GetCustomFieldDefinitions

Get all Custom Field Definitions existing in the system.

Method signature:

```
IList<CustomFieldDefinitionData> GetCustomFieldDefinitions
(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Custom Field Definitions existing in the system.

CustomFieldDefinitionData Object Type

Custom Field Definition Data Object Type Properties	
ID	Unique ID of Custom Field Definition.
Name	Unique Name of Custom Field Definition.
Description	Text to describe Custom Field Definition.
FieldType	Type of Custom Field Definition (Field Types).
TargetObjectCode	Type of Object (Object Types) for which Custom Field Definition is defined.

ScreenLabel	Name under which Custom Field is displayed to end-user (for example: on VISO interface)
ScreenToolTip	Text of a Tool Tip that is displayed to end-user (for example: on VISO interface)
Values	List of CustomFieldDefinitionValueData objects. Applicable for FieldType: List of Values.
Deleted	Boolean value indicating whether Custom Field Definition was deleted.
UpdatedDate	Last modification Date Time.

Field Types

- 0: Text
- 1: List of Values
- 2: On/Off

CustomFieldDefinitionValueData Object Type

Custom Field Definition Value Data Object Type Properties	
ID	Unique ID of Custom Field Definition Value. This ID should be used as the Value of AccessUserCustomFieldValueData not the text description.
Value	Text description of the value.

AccessUserCustomFieldValueData Object Type

Access User Custom Field Value Data Object Type Properties	
DefinitionID	ID of one of existing Custom Field Definitions
CustomFieldDefinitionName	Name of Custom Field Definition (this property does not autoupdate when you change DefinitionID, you need to update it manually)
Value	String value of Custom Field. Accepted values: Text: any String (including empty String) List of Values: ID of

	CustomFieldDefinitionValueData object (or empty String which means: no selection) On/Off: string parsable to Boolean value: "True" or "False". NULLs are not accepted.

GetPersonById

Get Access User Person by its ID.

Method signature:

```
PersonData GetPersonById(int personId, Guid sessionToken)
```

Where,

- personId – ID of the requested Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Person exist in the system, then PersonData is returned.

If Access User Person not exist in the system, then NULL is returned.

GetPersonRights

Get Access Rights identifiers assigned to specified Access User Person.

Method signature:

```
IEnumerable<int> GetPersonRights(int personId, Guid sessionToken)
```

Where,

- personId – ID of the Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights identifiers assigned to specified Access User Person.

GetPersonAssetsIds

Get Access User Assets identifiers assigned to specified Access User Person\Visitor.

Method signature:

```
IEnumerable<int> GetPersonAssetsIds(int personOrVisitorId, Guid sessionToken)
```

Where,

- personOrVisitorId– ID of the Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights identifiers assigned to specified Access User Person\Visitor.

GetPersonRightsView

Get Access Rights assigned to specified Access User Person.

Method signature:

```
IEnumerable<AssignedAccessRightsData> GetPersonRightsView(int personId, Guid sessionToken)
```

Where,

- personId – ID of the Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights assigned to specified Access User Person.

GetVisitorById

Get Access User Visitor by its ID.

Method signature:

```
VisitorData GetVisitorById(int visitorId, Guid sessionToken)
```

Where,

- visitorId – ID of the requested Access User Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Visitor exist in the system, then VisitorData is returned.

If Access User Visitor not exist in the system, then NULL is returned.

GetAssetById

Get Access User Asset by its ID.

Method signature:

```
AssetData GetAssetById(int assetId, Guid sessionToken)
```

Where,

- assetId – ID of the requested Access User Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Asset exist in the system, then AssetData is returned.

If Access User Asset not exist in the system, then NULL is returned.

GetGroupById

Get Access User Group by its ID.

Method signature:

```
GroupData GetGroupById(int groupId, Guid sessionToken)
```

Where,

- groupId – ID of the requested Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Group exist in the system, then GroupData is returned.

If Access User Group not exist in the system, then NULL is returned.

GetGroupRights

Get Access Rights identifiers assigned to specified Access User Group.

Method signature:

```
IEnumerable<int> GetGroupRights(int groupId, Guid sessionToken)
```

Where,

- groupId – ID of the Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights identifiers assigned to specified Access User Group.

GetGroupRightsView

Get Access Rights assigned to specified Access User Group.

Method signature:

```
IEnumerable<AssignedAccessRightsData> GetGroupRightsView(int groupId, Guid
sessionToken)
```

Where,

- groupId – ID of the Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Rights identifiers assigned to specified Access User Group.

GetPersonCredentials

Get Access Credentials assigned to specified Access User Person.

Method signature:

```
IEnumerable<CredentialData> GetPersonCredentials(int personId, Guid
sessionToken)
```

Where,

- personId – ID of the Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials assigned to specified Access User Person.

GetVisitorCredentials

Get Access Credentials assigned to specified Access User Visitor.

Method signature:

```
IEnumerable<CredentialData> GetVisitorCredentials(int visitorId, Guid
sessionToken)
```

Where,

- visitorId – ID of the Access User Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials assigned to specified Access User Visitor.

GetAssetCredentials

Get Access Credentials assigned to specified Access User Asset.

Method signature:

```
IEnumerable<CredentialData> GetAssetCredentials(int assetId, Guid sessionToken)
```

Where,

- assetId – ID of the Access User Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials assigned to specified Access User Asset.

GetGroupCredentials

Get Access Credentials assigned to specified Access User Group.

Method signature:

```
IEnumerable<CredentialData> GetGroupCredentials(int groupId, Guid sessionToken)
```

Where,

- groupId – ID of the Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Credentials assigned to specified Access User Group.

GetUserByCredential

Get information about the user assigned to Access Credential

Method signature:

```
UserData GetUserByCredential(int credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

Object containing information about the user.

GetMaxPersonId

Get max Access User Person identifier.

Method signature:

```
int GetMaxPersonId(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Max identifier of Access User Person existing in the system.

GetMaxCredentialId

Get max Access User Credential identifier.

Method signature:

```
int GetMaxCredentialId(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Max identifier of Access Credential existing in the system.

GetMaxFactorId

Get max Authentication Factor identifier.

Method signature:

```
int GetMaxFactorId(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Max identifier of Authentication Factor existing in the system.

GetMaxGroupId

Get max Access User Group identifier.

Method signature:

```
int GetMaxGroupId(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

Max identifier of Access User Group existing in the system.

InsertPerson

Allow to insert new Access User Person data.

Method signature:

```
int InsertPerson(PersonData personData, Guid sessionToken)
```

Where,

- personData – Inserted Access User Person data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Person data is successfully persisted on the database then the last person ID is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

Additional information:

Person custom fields values can be inserted by filling **`PeronsData.CustomFieldValues`** list

UpdatePerson

Allow to update existing Access User Person data.

Method signature:

```
int UpdatePerson(PersonData personData, Guid sessionToken)
```

Where,

- personData – Updated Access User Person data (valid person ID must be specified)
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Person data is successfully persisted on the database then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

Additional information:

Person custom fields values can be inserted by filling **`PeronsData.CustomFieldValues`** list

DeletePerson

Allow to delete existing Access User Person.

Method signature:

```
int DeletePerson(int personId, bool autoUnlinkRelatedObjects, bool
deleteAssignedCredentials, Guid sessionToken)
```

Where,

- personId - ID of the Access User Person to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- deleteAssignedCredentials – defines if assigned Access Credentials should be deleted along with Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Person data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

RestorePerson

Allow to restore existing (not erased) Access User Person.

Method signature:

```
int RestorePerson(int personId, Guid sessionToken)
```

Where

- personId - ID of the Access User Person to be deleted
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Person data is successfully restored then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

ErasePerson

Allow to erase existing Access User Person.

Method signature:

```
int PersonVisitor(int personId, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- personId - ID of the Access User Person to be deleted
- deleteWithLinkedData – determines whether the associated personal data will be deleted
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Person data is successfully erased then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

InsertVisitor

Allow to insert new Access User Visitor data.

Method signature:

```
int InsertVisitor(VisitorData visitorData, Guid sessionToken)
```

Where,

- visitorData – inserted Access User Visitor data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Visitor data is successfully persisted on the database then the last Visitor ID is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateVisitor

Allow to update existing Access User Visitor data.

Method signature:

```
int UpdateVisitor(VisitorData visitorData, Guid sessionToken)
```

Where,

- visitorData – updated Access User Visitor data (valid Visitor ID must be specified)
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Visitor data is successfully persisted on the database then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteVisitor

Allow to delete existing Access User Visitor.

Method signature:

```
int DeleteVisitor(int visitorId, bool autoUnlinkRelatedObjects, bool deleteAssignedCredentials, Guid sessionToken)
```

Where,

- visitorId - ID of the Access User Visitor to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- deleteAssignedCredentials – defines if assigned Access Credentials should be deleted along with Access User Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Visitor data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

RestoreVisitor

Allow to restore existing (not erased) Access User Visitor.

Method signature:

```
int RestoreVisitor(int visitorId, Guid sessionToken)
```

Where,

- visitorId - ID of the Access User Visitor to be deleted
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Visitor data is successfully restored then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

EraseVisitor

Allow to erase existing Access User Visitor.

Method signature:

```
int EraseVisitor(int visitorId, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- visitorId - ID of the Access User Visitor to be deleted
- deleteWithLinkedData – determines whether the associated personal data will be deleted
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access User Visitor data is successfully erased then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignCredentials

Allow assign Access Credentials to Access User Person\Visitor.

Method signature:

```
int AssignCredentials(int personOrVisitorId, int[] credentialIds, Guid
sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor to which the credentials will be assigned
- credentialIds – List of the Access Credential identifiers that will be assigned to specified Access User Person\Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Credentials are successfully assigned then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignCredential

Allow to unassign Access Credential from specified Access User Person\Visitor.

Method signature:

```
int UnassignCredential(int personOrVisitorId, int credentialId, Guid
sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor from which credential will be removed
- credentialId - unassigned Access Credential ID
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Credential is successfully unassigned from person 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignPersonRights

Allow to assign Access Rights to Access User Person\Visitor.

Method signature:

```
int AssignPersonRights(int personOrVisitorId, int[] accessRightsIds, Guid sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor to which the rights will be assigned
- accessRigthsIds – List of the Access Rights identifiers that will be assigned to specified Access User Person\Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Rights are successfully assigned to Access User Person\Visitor then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignPersonRightsWithActivateExpiryTime

Allow to assign Access Rights to Access User Person with right Activation/Expiration time.

Method signature:

```
int AssignPersonRightsWithActivateExpiryTime(int personOrVisitorId, int[] accessRightsIds, DateTime? activationTime, DateTime? expiryTime, Guid sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person or Visitor to which the rights will be assigned

- accessRigthsIds – List of the Access Rights identifiers that will be assigned to specified Access User Person
- activationTime – DateTime object of activation time, can be null
- expiryTime – DateTime object of expiry time, can be null
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Rights are successfully assigned to Access User Person then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignPersonRights

Allow to unassign Access Rights from Access User Person\Visitor.

Method signature:

```
int UnassignPersonRights(int personOrVisitorId, int accessRightsId, Guid sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor from which Access Rights will be removed
- accessRightsId – unassigned Access Rights ID
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Rights is successfully unassigned from Access User Person\Visitor then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

EnableAssignedPersonRights

Allow to enable Access Rights in Access User Person\Visitor.

Method signature:

```
int EnableAssignedPersonRights(int personOrVisitorId, int accessRightsId, Guid sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor to which the rights will be enabled
- accessRigthsId – ID of the Access Rights identifier that will be enabled to specified Access User Person

- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Rights are successfully enabled in Access User Person\Visitor then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DisableAssignedPersonRights

Allow to disable Access Rights in Access User Person\Visitor.

Method signature:

```
int DisableAssignedPersonRights(int personOrVisitorId, int accessRightsId, Guid sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor to which the rights will be disabled
- accessRigthsId – ID of the Access Rights identifier that will be disabled to specified Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Rights are successfully disabled in Access User Person\Visitor then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

InsertCredential

Allow to insert new Access Credential data.

Method signature:

```
int InsertCredential(CredentialData credentialData, Guid sessionToken)
```

Where,

- credentialData – Inserted Access Credential data
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Credential data is successfully persisted on the database then the last credential ID is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateCredential

Allow to update existing Access Credential data.

Method signature:

```
int UpdateCredential(CredentialData credentialData, Guid sessionToken)
```

Where,

- credentialData - Updated the Access Credential data (valid credential ID must be specified)
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Credential data is successfully persisted on the database then the last credential ID is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteCredential

Allow to delete existing Access Credential.

Method signature:

```
int DeleteCredential(int credentialId, bool autoUnlinkRelatedObjects, bool
returnFactorsToCardbox, Guid sessionToken)
```

Where,

- credentialId - ID of the Access Credential to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- returnFactorsToCardbox – defines if assigned Authentication Factors (of card type) should be returned to Cardbox or deleted along with Access Credential
- sessionToken – Current session token, obtained by Connect method execution
- synchronizeAfterDeletion – after deleting, sending to the controller partial configuration

Result:

If Access Credential data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

InsertAuthenticationFactor

Allow to insert new Authentication Factor (e.g. proximity card number) and assign it to specified Access Credential.

Method signature:

```
int InsertAuthenticationFactor(int credentialId, FactorData factorData, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the factor will be assigned
- factorData – Inserted Authentication Factor data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Authentication Factor is successfully persisted on the database then the last factor ID is returned. If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateAuthenticationFactor

Allow to updating existing Authentication Factor (e.g. proximity card number) data. Only Name and Disable properties can be updated using this function.

Method signature:

```
int UpdateAuthenticationFactor(FactorData factorData, Guid sessionToken)
```

Where,

- factorData – Updated Authentication Factor data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Authentication Factor is successfully updated on the database then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

RemoveAuthenticationFactor

Allow to remove Authentication Factor.

Method signature:

```
int RemoveAuthenticationFactor(int factorId, Guid sessionToken)
```

Where,

- factorId – ID of the Authentication Factor
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Authentication Factor is successfully removed then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignRights

Allow to assign Access Rights to Access Credential.

Method signature:

```
int AssignRights(int credentialId, int[] accessRightsIds, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the rights will be assigned
- accessRigthsIds – List of the Access Rights identifiers that will be assigned to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights are successfully assigned to Access Credential then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignRights

Allow to unassign Access Rights from Access Credential.

Method signature:

```
int UnassignRights(int credentialId, int accessRightsId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential from which Access Rights will be removed
- accessRightsId – unassigned Access Rights ID
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights is successfully unassigned from Access Credential then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignRightsGroups

Allow to assign Access Rights Groups to Access Credential.

Method signature:

```
int AssignRightsGroups(int credentialId, int[] accessRightsGroupsIds, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the rights will be assigned
- accessRigthsGroupsIds – List of the Access Rights Groups identifiers that will be assigned to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights Groups are successfully assigned to Access Credential then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetAssetRightIds

Get Access Rights identifiers assigned to specified Access User Asset.

Method signature:

```
IEnumerable<int> GetAssetRightIds(int accessUserAssetId, Guid sessionToken)
```

Where,

- accessUserAssetId – ID of the AccessUserAsset
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of AccessRights identifiers assigned to specified AccessUserAsset.

AssignPersonAssets

Allow to assign Access User Asset to Access User Person\Visitor.

Method signature:

```
int AssignPersonAssets(int personOrVisitorId, int[] assetsIds, Guid
sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor to which the assets will be assigned
- assetsIds – List of the Assets identifiers that will be assigned to specified Access User Person\Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Assets are successfully assigned to AccessUserPersons\Visitors then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignPersonAsset

Allow to unassign Access User Asset from Access User Person\Visitor.

Method signature:

```
int UnassignPersonAsset(int personOrVisitorId, int assetId, Guid sessionToken)
```

Where,

- personOrVisitorId – ID of the Access User Person\Visitor from which asset will be removed
- assetsIds – List of the Assets identifiers that will be assigned to specified Access User Person\Visitor
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Assets are successfully assigned to AccessUserPersons\Visitors then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignAssetCredentials

Allow to assign Access User Asset to Access Credential.

Method signature:

```
int AssignAssetCredentials(int accessUserAssetId, int[] credentialIds, Guid
sessionToken);
```

Where,

- accessUserAssetId – ID of the Access User Asset to which the credentials will be assigned

- credentialIds – List of the credentials identifiers that will be assigned to specified Access User Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Credentials were successfully assigned to AccessUserAsset then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignAssetCredential

Allow to unassign Access Credential from Access User Asset.

Method signature:
```
int UnassignAssetCredential(int accessUserAssetId, int credentialId, Guid sessionToken)
```
Where,

- accessUserAssetId – ID of the Access User Asset to which the credentials will be assigned
- credentialId – ID of the Access Credential wich will be unassigned from specified Access User Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:
If AccessCredential was successfully unassigned from AccessUserAsset then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignAssetRights

Allow to assign Access User Asset to Access Rights.

Method signature:
```
int AssignAssetRights(int accessUserAssetId, int[] accessRightsIds, Guid sessionToken)
```
Where,

- accessUserAssetId – ID of the Access User Asset to which the rights will be assigned
- accessRightsIds – List of the rights identifiers that will be assigned to specified Access User Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Rights were successfully assigned to AccessUserAsset then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignAssetRight

Allow to unassign Access Right from Access User Asset.

Method signature:

```
int UnassignAssetRight(int accessUserAssetId, int accessRightId, Guid sessionToken)
```

Where,

- accessUserAssetId – ID of the Access User Asset to which the rights will be assigned
- accessRightId – List of the rights identifiers that will be assigned to specified Access User Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:
If AccesRight was successfully unassigned from AccessUserAsset then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignRightsGroup

Allow to unassign Access Rights Group from Access Credential.

Method signature:

```
int UnassignRights(int credentialId, int accessRightsGroupId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential from which Access Rights will be removed
- accessRightsGroupId – unassigned Access Rights Group ID
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access Rights Group is successfully unassigned from Access Credential then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

EnableAssignedCredentialRights

Allow to enable Access Rights in Access Credential.

Method signature:

```
int EnableAssignedCredentialRights(int credentialId, int accessRightsId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the rights will be enabled
- accessRigthsId – ID of the Access Rights identifier that will be enabled to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights are successfully enabled in Access Credential then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DisableAssignedCredentialRights

Allow to enable Access Rights in Access Credential.

Method signature:

```
int DisableAssignedCredentialRights(int credentialId, int accessRightsId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the rights will be disabled
- accessRigthsId – ID of the Access Rights identifier that will be disabled to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights are successfully disabled in Access Credential then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

EnableAssignedCredentialRightsGroup

Allow to enable Access Rights Group in Access Credential.

Method signature:

```
int EnableAssignedCredentialRightsGroup(int credentialId, int
accessRightsGroupId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the rights will be enabled
- accessRigthsId – ID of the Access Rights Group identifier that will be enabled to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights Group are successfully enabled in Access Credential then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DisableAssignedCredentialRightsGroup

Allow to disable Access Rights Group in Access Credential.

Method signature:

```
int DisableAssignedCredentialRightsGroup(int credentialId, int
accessRightsGroupId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the rights will be disabled
- accessRigthsId – ID of the Access Rights Group identifier that will be disabled to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights Group are successfully disabled in Access Credential then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

InsertGroup

Allow to insert new Access User Group data.

Method signature:

```
int InsertGroup(GroupData groupData, Guid sessionToken)
```

Where,

- groupData – Inserted Access User Group data

- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access User Group data is successfully persisted on the database then the last group ID is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateGroup

Allow to update existing Access User Group data.

Method signature:

```
int UpdateGroup(GroupData groupData, Guid sessionToken)
```

Where,

- groupData – Updated Access User Group data (valid group ID must be specified)
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access User Group data is successfully persisted on the database then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteGroup

Allow to delete existing Access User Group.

Method signature:

```
int DeleteGroup(int groupId, bool autoUnlinkRelatedObjects, bool
deleteAssignedCredentials, Guid sessionToken)
```

Where,

- groupId - ID of the Access User Group to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- deleteAssignedCredentials – defines if assigned Access Credentials should be deleted along with Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:
If Access User Group data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignGroupRights

Allow to assign Access Rights to Access User Group.

Method signature:

```
int AssignGroupRights(int groupId, int[] accessRightsIds, Guid sessionToken)
```

Where,

- groupId – ID of the Access User Group to which the rights will be assigned
- accessRigthsIds – List of the Access Rights identifiers that will be assigned to specified Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights are successfully assigned to Access User Group then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UnassignGroupRights

Allow to unassign Access Rights from Access User Group.

Method signature:

```
int UnassignGroupRights(int groupId, int accessRightsId, Guid sessionToken)
```

Where,

- groupId – ID of the Access User Group from which Access Rights will be removed
- accessRightsId – unassigned Access Rights ID
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights is successfully unassigned from Access User Group then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

EnableAssignedGroupRights

Allow to enable Access Rights in Access User Group.

Method signature:

```
int EnableAssignedGroupRights(int groupId, int accessRightsId, Guid
sessionToken)
```

Where,

- groupId – ID of the Access User Group to which the rights will be enabled
- accessRigthsId – ID of the Access Rights identifier that will be enabled to specified Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights are successfully enabled in Access User Group then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DisableAssignedGroupRights

Allow to disable Access Rights in Access User Group.

Method signature:

```
int DisableAssignedGroupRights(int groupId, int accessRightsId, Guid
sessionToken)
```

Where,

- groupId – ID of the Access User Group to which the rights will be disabled
- accessRigthsId – ID of the Access Rights identifier that will be disabled to specified Access User Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Access Rights are successfully disabled in Access User Group then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetPowerSupplies

Get all Power Supplies existing in the system.

Method signature:

```
IEnumerable<PowerSupplyData> GetPowerSupplies(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Power Supplies existing in the system.

GetDisplays

Get all Displays existing in the system.

Method signature:

```
IEnumerable<DisplayData> GetDisplays(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Displays existing in the system.

GetFunctions

Get all Functions existing in the system.

Method signature:

```
IEnumerable<FunctionData> GetFunctions(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Functions existing in the system.

GetFunctionGroups

Get all FunctionGroups existing in the system.

Method signature:

```
IEnumerable<FunctionGroupData> GetFunctions(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all FunctionGroups existing in the system.

GetCalendars

Get all Calendars existing in the system.

Method signature:

```
IEnumerable<CalendarData> GetCalendars(Guid sessionToken);
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Calendars existing in the system.

GetCalendarByID

Get Calendar for given calendar ID.

Method signature:

```
CalendarData GetCalendarByID(int calendarId, Guid sessionToken)
```

Where,

- calendarId – ID of Calendar
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Calendar exist in the system, then data is returned.

InsertCalendar

Allow to insert new Calendar data.

Method signature:

```
int InsertCalendar(CalendarData calendarData, Guid sessionToken)
```

Where,

- calendarData – Inserted Calendar data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Calendar data is successfully persisted on the database then the last calendar ID is returned. If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateCalendar

Allow to updating existing Calendar data.

Method signature:

```
int UpdateCalendar(CalendarData calendarData, Guid sessionToken)
```

Where,

- calendarData – Updated Calendar data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Calendar is successfully updated on the database then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteCalendar

Allow to delete existing Calendar.

Method signature:

```
int DeleteCalendar(int calndarId, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- calendarId - ID of the Calendar to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Calendar data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetCalendarEntries

Get all list of CalendarEntry for given calendar ID.

Method signature:

```
IEnumerable<CalendarEntryData> GetCalendarEntries(int calendarId, Guid
sessionToken);
```

Where,

- calendarId - ID of the Calendar

- sessionToken – Current session token, obtained by Connect method execution

Result:
List of all CalendarEntryData.

UpdateCalendarEntries

Allow to updating existing CalendarEntries data.

Method signature:

```
int UpdateCalendarEntries(IEnumerable<CalendarEntryData> calendarEntryDatas,
Guid sessionToken)
```

Where,

- calendarEntryDatas – new calendar entry list
- sessionToken – Current session token, obtained by Connect method execution

Result:
If list of CalendarEntryData is successfully updated on the database then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

ClearCalendarEntries

Allow to delete all existing CalendarEntries for given calendar ID.

Method signature:

```
int ClearCalendarEntries(int calendarId, Guid sessionToken)
```

Where,

- calendarId - ID of the Calendar
- sessionToken – Current session token, obtained by Connect method execution

Result:
If CalendarEntries data for given calendarId have been successfully deleted then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetSchedules

Get all Schedules existing in the system.

Method signature:

```
IEnumerable<ScheduleData> GetSchedules(Guid sessionToken);
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Schedules existing in the system.

GetScheduleByID

Get Schedule for given schedule ID.

Method signature:

```
ScheduleData GetScheduleByID(int scheduleId, Guid sessionToken)
```

Where,

- scheduleId – ID of Schedule
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Schedule exist in the system, then data is returned.

GetScheduleByType

Get list of Schedules for given schedule type.

Method signature:

```
IEnumerable<ScheduleData> GetScheduleByType(int scheduleType, Guid sessionToken)
```

Where,

- scheduleType – Code of Schedule Type
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of schedule data for given schedule type is returned.

Schedule Types

0: General Purpose Maintained

1: Door Mode

2: T&A Mode

3: Authentication Policy

4: General Purpose Momentary

8: Global Command

16: Synchronisation

CreateSchedule

Allow to create new Schedule.

Method signature:

```
int CreateSchedule(ScheduleData scheduleData, Guid sessionToken)
```

Where,

- scheduleData – new Schedule data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Schedule data is successfully persisted on the database then the last schedule ID is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

EditSchedule

Allow to updating existing Schedule.

Method signature:

```
int EditSchedule(ScheduleData scheduleData, Guid sessionToken)
```

Where,

- scheduleData – Updated Schedule data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If ScheduleData is successfully updated on the database then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteSchedule

Allow to delete existing Schedule.

Method signature:

```
int DeleteSchedule(int id, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- id - ID of the Schedule to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Schedule data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateWeeklySchedule

Allow to updating Weekly Schedule data for given schedule ID.

Method signature:

```
int UpdateWeeklySchedule(int scheduleId, IList<DailyScheduleData>
modifiedDailySchedules, Guid sessionToken)
```

Where,

- scheduleId – new calendar entry list
- modifiedDailySchedules – new DailySchedule list
- sessionToken – Current session token, obtained by Connect method execution

Result:

If list of DailyScheduleData is successfully updated on the database then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

ClearWeeklySchedule

Allow to delete all existing DailyScheduleData for given schedule ID.

Method signature:

```
int ClearWeeklySchedule(int scheduleId, Guid sessionToken)
```

Where,

- scheduleId - ID of the Schedule
- sessionToken – Current session token, obtained by Connect method execution

Result:

If all DailyScheduleData for given scheduleId have been successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetSpecialEventByID

Get SpecialEvent for given SpecialEvent ID.

Method signature:

```
SpecialEventData GetSpecialEventByID(int specialEventId, Guid sessionToken)
```

Where,

- specialEventId – ID of SpecialEvent
- sessionToken – Current session token, obtained by Connect method execution

Result:

If SpecialEvent exist in the system, then data is returned.

GetSpecialEventsByScheduleID

Get all SpecialEvents for given schedule ID.

Method signature:

```
IEnumerable<SpecialEventData> GetSpecialEventsByScheduleID(int scheduleId, Guid sessionToken)
```

Where,

- scheduleId – ID of Schedule
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of SpecialEvent data for given scheduleId is returned.

InsertSpecialEvent

Allow to insert new SpecialEvent data.

Method signature:

```
int InsertSpecialEvent(SpecialEventData specialEventData, Guid sessionToken)
```

Where,

- specialEventData – Inserted SpecialEvent data
- sessionToken – Current session token, obtained by Connect method execution

Result:
If SpecialEvent data is successfully persisted on the database then the last SpecialEvent ID is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateSpecialEvent

Allow to updating existing SpecialEvent data.

Method signature:

```
int UpdateSpecialEvent(SpecialEventData specialEventData, Guid sessionToken)
```

Where,

- specialEventData – Updated SpecialEvent data
- sessionToken – Current session token, obtained by Connect method execution

Result:
If SpecialEvent is successfully updated on the database then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteSpecialEvent

Allow to delete existing SpecialEvent.

Method signature:

```
int DeleteSpecialEvent(int id, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- id - ID of the SpecialEvent to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- sessionToken – Current session token, obtained by Connect method execution

Result:
If SpecialEvent data is successfully deleted then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetAccessRightsByID

Get AccessRights for given AccessRights ID.

Method signature:

```
AccessRightsData GetAccessRightsByID(int rightId, Guid sessionToken)
```

Where,

- rightId – ID of AccessRights
- sessionToken – Current session token, obtained by Connect method execution

Result:

If AccessRights exist in the system, then data is returned.

InsertAccessRights

Allow to insert new AccessRights data.

Method signature:

```
int InsertAccessRights(AccessRightsData accessRightsData, Guid sessionToken)
```

Where,

- accessRightsData – Inserted AccessRigts data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If AccessRights data is successfully persisted on the database then the last AccessRigts ID is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateAccessRights

Allow to updating existing AccessRights data.

Method signature:

```
int UpdateAccessRights(AccessRightsData accessRightsData, Guid sessionToken)
```

Where,

- accessRightsData – Updated AccessRigts data
- sessionToken – Current session token, obtained by Connect method execution

Result:

If AccessRights is successfully updated on the database then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteAccessRights

Allow to delete existing AccessRights.

Method signature:

```
int DeleteAccessRights(int id, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- id - ID of the AccessRigts to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- sessionToken – Current session token, obtained by Connect method execution

Result:

If AccessRights data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetAccessRuleByID

Get AccessRule for given AccessRule ID.

Method signature:

```
AccessRuleData GetAccessRuleByID(int ruleId, Guid sessionToken)
```

Where,

- ruleId – ID of AccessRule
- sessionToken – Current session token, obtained by Connect method execution

Result:

If AccessRule exist in the system, then data is returned

GetAccessRuleByRightsID

Get all AccessRules for given AccessRights ID.

Method signature:

```
IEnumerable<AccessRuleData> GetAccessRulesByRightsID(int rightsId, Guid
sessionToken)
```

Where,

- rightsId – ID of AccessRights
- sessionToken – Current session token, obtained by Connect method execution

Result:
List of AccessRule data for given rightsId is returned.

InsertAccessRule

Allow to insert new AccessRule data.

Method signature:

```
int InsertAccessRule(AccessRuleData accessRuleData, Guid sessionToken)
```

Where,

- accessRuleData – Inserted SpecialEvent data
- sessionToken – Current session token, obtained by Connect method execution

Result:
If AccessRule data is successfully persisted on the database then the last AccessRule ID is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

UpdateAccessRule

Allow to updating existing AccessRule data.

Method signature:

```
int UpdateAccessRule(AccessRuleData accessRuleData, Guid sessionToken)
```

Where,

- accessRuleData – Updated AccessRule data
- sessionToken – Current session token, obtained by Connect method execution

Result:
If AccessRule is successfully updated on the database then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

DeleteAccessRule

Allow to delete existing AccessRule.

Method signature:

```
int DeleteAccessRule(int id, bool deleteWithLinkedData, Guid sessionToken)
```

Where,

- id - ID of the AccessRule to be deleted
- autoUnlinkRelatedObjects – defines if related objects should be auto unlinked
- sessionToken – Current session token, obtained by Connect method execution

Result:

If AccessRule data is successfully deleted then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetVirtualControllers

Get all Virtual Controllers.

Method signature:

```
IEnumerable<VirtualControllerDto> GetVirtualControllers(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of VirtualControllers data returned.

GetPortProfiles

Get all Port Profiles for given Virtual Controller ID.

Method signature:

```
IEnumerable<PortProfileData> GetPortProfiles(int virtualControllerId, Guid
sessionToken)
```

Where,

- virtualControllerId – ID of Virtual Controller
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Port Profiles data for given Virtual Controller ID is returned.

GetPortTemplates

Get all Port Templates for given virtual controller ID.

Method signature:

```
IEnumerable<PortTemplateData> GetPortTemplates(int virtualControllerId, Guid sessionToken)
```

Where,

- virtualControllerId – ID of Virtual Controller
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Port Templates data for given Virtual Controller ID is returned.

GetPortMasterGroups

Get all Port Masters Groups for given Virtual Controller ID.

Method signature:

```
IEnumerable<PortMasterGroupData> GetPortMasterGroups(int virtualControllerId, Guid sessionToken)
```

Where,

- virtualControllerId – ID of Virtual Controller
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Port Master Groups data for given Virtual Controller ID is returned.

GetCredentialAssignedPortProfiles

Get all assigned PortProfile Ids for given credential ID.

Method signature:

```
IEnumerable<int> GetCredentialAssignedPortProfiles(int credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of AccessCredential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Port Profile Ids assigned to Access Credential for given credentialId is returned.

GetGroupAssignedPortProfiles

Get all assigned PortProfile Ids for given group ID.

Method signature:

```
IEnumerable<int> GetGroupAssignedPortProfiles(int groupId, Guid sessionToken)
```

Where,

- groupId – ID of AccessCredential
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Port Profile Ids assigned to a group for given groupId is returned.

AssignPortProfileToCredential

Allow to assign PortProfile to Access Credential.

Method signature:

```
int AssignPortProfileToCredential(int credentialId, int[] portProfileIds, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to which the PortProfiles will be assigned
- portProfileIds – List of the PortProfile Ids that will be assigned to specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If PortProfiles were successfully assigned to AccessCredential then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

RemovePortProfileFromCredential

Allow to remove PortProfile form Access Credential.

Method signature:

```
int RemovePortProfileFromCredential(int credentialId, int[] portProfileIds, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential from which the PortProfiles will be removed
- portProfileIds – List of the PortProfile Ids that will be remove from specified Access Credential
- sessionToken – Current session token, obtained by Connect method execution

Result:
If PortProfile was successfully removed from AccessCredential then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

AssignPortProfileToGroup

Allow to assign PortProfile to Access User Group.

Method signature:

```
int AssignPortProfileToGroup(int groupId, int[] portProfileIds, Guid sessionToken)
```

Where,

- groupId – ID of the Access User Group to which the PortProfiles will be assigned
- portProfileIds – List of the PortProfile Ids that will be assigned to specified Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:
If PortProfiles were successfully assigned to AccessUserGroup then 0 is returned.
If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

RemovePortProfileFromGroup

Allow to remove PortProfile form Access User Group.

Method signature:

```
int RemovePortProfileFromGroup(int groupId, int[] portProfileIds, Guid sessionToken)
```

Where,

- groupId – ID of the Access User Group from which the PortProfiles will be removed
- portProfileIds – List of the PortProfile Ids that will be remove from specified Access User Group
- sessionToken – Current session token, obtained by Connect method execution

Result:

If PortProfile was successfully removed from AccessUserGroup then 0 is returned.

If any errors occurs then -1 is returned. You can check last error details by executing method GetLastErrorMessage or by checking system log file.

GetKoneAccessRights

Get all KoneAccessRights existing in the system.

Method signature:

```
IEnumerable<KoneAccessRightsData> GetKoneAccessRights(Guid sessionToken);
```

Where

- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetAssignedKoneAccessRightsToAccessCredential

Get all assigned KoneAccessRightses to given AccessCredential ID.

Method signature:

```
IEnumerable<KoneAssignedAccessRightsData>
GetAssignedKoneAccessRightsToAccessCredential(int credentialId, Guid sessionToken);
```

Where

- `credentialId` – ID of AccessCredential
- sessionToken – Current session token, obtained by Connect method execution

Result:

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetKoneHomeFloors

Get all KoneHomeFloors existing in the system.

Method signature:

```
IEnumerable<KoneHomeFloorData> GetKoneHomeFloors(Guid sessionToken);
```

Where

- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetKoneCallTypes

Get all KoneCallTypes existing in the system.

Method signature:

```
IEnumerable<KoneCallTypeData> GetKoneCallTypes(Guid sessionToken);
```

Where

- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetKonePersonCategories

Get all KonePersonCategories existing in the system.

Method signature:

```
IEnumerable<KonePersonCategoryData> GetKonePersonCategories(Guid sessionToken);
```

Where

- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetKoneTimeZones

Get all KoneTimeZones existing in the system.

Method signature:

```
IEnumerable<KoneTimeZoneData> GetKoneTimeZones(Guid sessionToken);
```

Where

- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

AddKoneAccessRights

Allow to add KoneAccessRightses to AccessCredential.

Method signature:

```
int AddKoneAccessRights(int credentialId, KoneAssignedAccessRightsData[]
koneRightses, Guid sessionToken);
```

Where,

- credentialId - ID of Access Credential
- koneRightses – Kone Access Rightses to assign
- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then error code (less than 0) is returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

RemoveKoneAccessRights

Allow to remove KoneAccessRightses from AccessCredential.

Method signature:

```
int RemoveKoneAccessRights(int credentialId, int[] rightsId, Guid sessionToken);
```

Where,

- credentialId - ID of Access Credential
- koneRightsIds – IDs of Kone Access Rights

- sessionToken – Current session token, obtained by Connect method execution

Result

If any errors occurs then error code (less than 0) is returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetLastErrorMessage

Allow to get the last error details.

Method signature:

```
string GetLastErrorMessage()
```

Result:

Last error details.

GetLastActivityLogId

Get last registered Operator Activity Log entry ID.

Method signature:

```
int GetLastActivityLogId(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

If any Operator Activity Log entry exist in the system, then last registered entry ID is returned.

Else 0 is returned.

TakeActivityLogsStartingFrom

Take Operator Activity Log entries starting from specified entry ID.

Method signature:

```
IList<OperatorActivityLogData> TakeActivityLogsStartingFrom(int startingLogId, Guid sessionToken)
```

Where,

- startingLogId – Starting Operator Activity Log Entry ID
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Operator Activity Log entries starting from specified entry ID.

OperatorActivityLogData Object Type

Operator Activity Log Data Object Type Properties	
ID	Unique ID of operator log entry.
OperatorID	ID of operator who executted logged command
TimeStamp	TimeStamp of time when command execution occured
CommandCode	Code of executed command. Available code are decribed below (Command Codes)
ObjectCode	Operator Activity Log entry source type. Available types are described below (Object Types).
ObjectID	Operator Activity log source object ID. (if available)
ObjectDecription	Operator Activity log source name (if available)
ObjectSnapshotID	Operator Activity log source object snapshot (if available)
HostName	Name of host from logged command was executed

Command Codes

- 0 – Undefined
- 1 – Test
- 2 – Show
- 3 – Add
- 4 – Edit
- 5 – Delete
- 6 – Refresh
- 7 – Report
- 8 – Copy
- 9 – Paste
- 10 – Cut
- 11 – Undo
- 12 – Open
- 13 – Close
- 14 – Print

- 15 – OK
- 16 – Cancel
- 18 – Assign
- 19 – Remove Assign
- 20 – Enable
- 21 – Disable
- 22 – Find
- 23 – Grant
- 24 – Revoke
- 25 – Merge
- 26 – Overwrite
- 27 – change
- 28 – Continue
- 29 – Abort
- 30 – Monitore
- 31 – Export
- 32 – Select
- 33 – Read
- 40 – Erase
- 50 – Grant Permission
- 51 – Revoke Permission
- 52 – Assign Role
- 53 – Remove Assigned Role
- 54 – Show Organization
- 55 – Show Operator
- 56 – Show Role
- 57 – Show Operator Activity Log
- 70 – Add new Connection
- 71 – Change Connection
- 72 – Remove Connection
- 100 – Login
- 101 – Logout
- 102 – Next
- 103 – Previous
- 104 – Import
- 105 – Display
- 106 – Confirm
- 107 – Restore

Event Log Management

This service allow you to search data in the system Event Log.

EventLogEntryData Object Type

Event Log Entry Data Object Type Properties	
ID	Unique ID of Event Log Entry.
EventID	ID of Event Log Entry from Access Controller. It's not unique.
EventCode	Code of event. Detailed event codes list is described below (Event Codes).
LoggedOn	Event Log Entry logging date and time.
SourceType	Event Log Entry source type. Available types are described below (Object Types).
SourceID	Event Log Entry source object ID.
LocationType	Event Log Entry location type. Available types are described below (Object Types).
LocationID	Event Log Entry location object ID.
OptionType	Event Log Entry option type. Available types are described below (Object Types).
OptionID	Event Log Entry option object ID.
Function	Event Log Entry Function.
ActionStatus	Event Log Entry Action Status. Available statutes are described below (Action Statuses).
AccessCredentialID	Event Log Entry Access Credential ID. It can be null.
CategoryID	ID of Event Log Entry Category. It can be null.
NetworkID	Event Log Entry Network ID.
ControllerID	Event Log Entry Controller ID.
PersonID	Event Log Entry Person or Visitor ID. It's only available for entries which Access Credential ID is not empty.
GroupID	Event Log Entry Group ID. It's only available for entries which Access Credential ID is not empty.
AssetID	Event Log Entry Asset ID. It's only available for entries which Access

	Credential ID is not empty.
Details	Event Log Entry details (List of Details).
EventTag	Event Log Entry tag.
Comment	Event Log Entry comment. It can be defined by the system operator.

Event Codes

- 0 - Test Event
- 1 - Failed Login Attempts Limit Exceeded
- 2 - Access Confirmed by Door Open Sensor
- 3 - Access Not Confirmed by Door Open Sensor
- 4 - Card Inserted
- 5 - Card Removed
- 6 - Access Point Lockout
- 8 - Access Point Lockout Returned
- 13 - Authentication Factor Read
- 261 - Door Lock Activated
- 262 - Door Lock Deactivated
- 302 - Unlocked Door Mode
- 321 - Forced Door Alarm
- 322 - Door Open Too Long Alarm
- 323 – Access Door Open Too Long Prealarm
- 601 - Door Access Granted
- 602 - Door Access Denied
- 603 – Event Access Point Entry Random Personal Check
- 604 - Pass-back Violation
- 605 - Elevator Access Granted
- 606 - Elevator Access Denied
- 607 - Door Access Disabled
- 608 - Disable Door Access Denied
- 609 - Door Access Enabled
- 610 - Enable Door Access Denied
- 611 - Maintained T&A Mode
- 612 - Set Maintained T&A Mode Denied
- 613 - Momentary T&A Mode
- 614 - Set Momentary T&A Mode Denied
- 615 - T&A Event
- 616 - Register T&A Event Denied

- 617 - Authentication Policy Set
- 618 - Set Authentication Policy Denied
- 619 - Door Bell
- 620 - Door Bell Denied
- 621 - Duress Event
- 622 - Register Duress Event Rejected
- 623 - Trace Event
- 624 - Register Trace Event Denied
- 625 - Guard Tour Event
- 626 - Register Guard Tour Event Denied
- 627 - Muster Event
- 628 - Register Muster Event Denied
- 629 - Access Granted
- 630 - Access Denied
- 631 - Emergency Door Locked Mode
- 632 - Emergency Door Locked Mode Denied
- 633 - Emergency Door Unlocked Mode
- 634 - Emergency Door Unlocked Mode Denied
- 635 - Emergency Door Mode Cleared
- 636 - Clear Emergency Door Mode Denied
- 637 - Normal Door Mode
- 638 - Normal Door Mode Denied
- 639 - Locked Door Mode
- 640 - Locked Door Mode Denied
- 641 - Unlocked Door Mode
- 642 - Unlocked Door Mode Denied
- 643 - Conditional-unlocked Door Mode
- 644 - Set Conditional-unlocked Door Mode Denied
- 645 - Door Open
- 646 - Door Open Denied
- 647 - Door Closed
- 648 - Door Closed Denied
- 649 - Door Open Too Long Alarm Cleared
- 650 - Cancel Door Open Too Long Alarm Rejected
- 651 - Automation Node Set ON Instant
- 652 - Set Automation Node ON Instant Denied
- 653 - Automation Node Set ON Timed
- 654 - Set Automation Node ON Timed Denied

- 655 - Automation Node OFF Instant
- 656 - Set Automation Node OFF Instant Denied
- 657 - Automation Node Switched ON
- 658 - Automation Node Switched OFF
- 663 - Access Door Bell
- 664 - Access Door Bell Denied
- 671 - Armed ON Mode
- 672 - Armed ON Mode Denied
- 673 - Armed OFF Mode
- 674 - Armed OFF Mode Denied
- 675 – Automation Nodes Clear Timed Granted
- 676 - Automation Nodes Clear Timed Denied
- 677 – Main Board Controller Restart Granted
- 678 - Main Board Controller Restart Denied
- 681 - Postpone Auto-arming
- 682 - Postpone Auto-arming Denied
- 683 - Armed ON Mode Disabled
- 684 - Armed ON Mode Disable Denied
- 685 - Armed ON Mode Disable Cleared
- 686 - Clear Armed ON Mode Disable Denied
- 687 - Armed ON Mode Request
- 688 - Armed ON Mode Request Denied
- 689 - Armed OFF Mode Request
- 690 - Armed OFF Mode Request Denied
- 691 - Block Door Toggle ON
- 692 - Block Door Toggle ON Denied
- 693 - Unblock Door Toggle OFF
- 694 - Unblock Door Toggle OFF Denied
- 695 - Block Door Toggle OFF
- 696 - Block Door Toggle OFF Denied
- 697 - Forced Door Alarm Cleared
- 698 - Clear Door Forced Alarm Denied
- 701 - Occupancy Register Cleared
- 702 - Clear Occupancy Register Denied
- 703 - Pass-back Register Cleared
- 704 - Clear Pass-back Register Denied
- 705 - AC Lost
- 706 - AC Lost Denied

- 707 - AC Lost Returned
- 708 - AC Lost Return Denied
- 709 - AC Lost Delayed
- 710 - AC Lost Delayed Denied
- 711 - Delayed AC Lost Returned
- 712 - Delayed AC Lost Return Denied
- 713 - Low Battery
- 714 - Low Battery Denied
- 715 - Low Battery Returned
- 716 - Low Battery Return Denied
- 717 - Battery Failure
- 718 - Battery Failure Denied
- 719 - Battery Failure Returned
- 720 - Battery Failure Return Denied
- 721 - TML Power Output Overload
- 722 - TML Power Output Overload Denied
- 723 - TML Power Output Overload Returned
- 724 - TML Power Output Overload Return Denied
- 725 - AUX Power Output Overload
- 726 - AUX Power Output Overload Rejected
- 727 - AUX Power Output Overload Returned
- 728 - AUX Power Output Overload Return Denied
- 729 - Power Output Failure
- 730 - Power Output Failure Denied
- 731 - Power Output Failure Returned
- 732 - Power Output Failure Return Denied
- 733 - Authorisation Mode: Deny All
- 734 - Denied to set Authorisation Mode: Deny All
- 735 - Authorisation Mode: Grant All
- 736 - Denied to set Authorisation Mode: Grant All
- 737 - Authorisation Mode: Authorise
- 738 - Denied to set Authorisation Mode: Authorise
- 739 - Authorisation Mode: External Verification
- 740 - Denied to set Authorisation Mode: External Verification
- 741 - Authorisation Mode: Authorisation Delayed
- 742 - Denied to set Authorisation Mode: Authorisation Delayed
- 743 - Authorisation Mode: None
- 744 - Denied to set Authorisation Mode: None

- 745 - External authorisation - access confirmed
- 746 - External authorisation denied
- 747 - Delayed authorisation - access disabled
- 748 - Delayed authorisation denied
- 801 - Preparing to Start Service Mode
- 802 - Preparing to Start Normal Mode
- 803 - Controller in Service Mode
- 804 - Controller in Normal Mode without Access Control Logic
- 805 - Controller in Normal Mode with Access Control Logic
- 807 - Output Current Overload
- 808 - Output Current Overload Returned
- 809 – External Signal Level Normal
- 810 - External Signal Level Low
- 811 - Controller Restart Request
- 812 - Controller Restart Success
- 813 - Communication Watchdog Restart
- 814 - Communication Key Set
- 815 - Elevator Access Trouble
- 816 - Elevator Access Trouble Returned
- 817 - Communication with External Module Lost
- 818 - Communication with External Module Returned
- 819 - External Module Operation Discontinued
- 820 - External Module Operation Resumed
- 821 - Time and Date Change Request
- 822 - Time and Date Changed
- 823 - Event Subdevice Normal
- 824 - Event Subdevice Fail
- 831 - Switch to New Configuration Request
- 832 - Switch to New Configuration Success
- 833 - Switch to New Configuration Error
- 841 - Device Discovery Request
- 842 - Device Discovery Error
- 843 - Device Discovery Start
- 851 - Firmware Upgrade Request
- 852 - Firmware Upgrade Error
- 853 - Firmware Upgrade Start
- 861 - Load Main Board Configuration Request
- 862 - Load Main Board Configuration Error

- 863 - Load Main Board Configuration Success

Action Statuses

- 0 - OK
- 1 - No Authorisation to login on Access Point
- 2 - Authentication Factor: Parse error
- 3 - Authenticaiton Factor: Unknown
- 4 - Authenticaiton Factor: Incorrect,
- 5 - Authentication Factor: Disabled,
- 6 - Another authentication process in progress,
- 7 - Authentication Policy Check: Canceled,
- 8 - Authentication Policy Check: Timeout,
- 9 - Authentication Policy Check: Failed,
- 10 - Access Credential: Diabled,
- 11 - Access Credential: Not yet active,
- 12 - Access Credential: Expired,
- 13 - Access Credential: Before valid time,
- 14 - Access Credential: After valid time,
- 15 - Access Point: Temporary blocked due to penalty timer,
- 16 – Object controlled by other Input,
- 31 - Source object beyond activity schedule,
- 32 - Destination object beyond activity schedule,
- 33 - Authorisation Check: No decision,
- 34 - Authorisation Check: Function Denied,
- 35 - Authorisation Check: Access Point Denied,
- 36 - Authorisation Check: Place of Action Denied,
- 37 - Authorisation Check: Function Parameter Denied,
- 41 - No Authorisation for Authentication,
- 51 - No Authorisation for Action,
- 52 - Access Credential: Over Max Usage Rule,
- 53 - Access Credential: Not active,
- 54 - Access Credential: Over Max Days Rule,
- 55 - Passback rule violation,
- 56 - Wrong Exit Zone,
- 57 - Wrong Last Zone,
- 58 - Upper Occupancy Limit,
- 59 - Lower Occupancy Limit violation,
- 60 - Thread too low,

- 61 – Access disabled because zone armed,
- 62 - No auhorisation for extended lock pulse time,
- 63 - Arming disabled,
- 64 - Alarm zone already armed (function arming disabled),
- 65 - Alarm zone already armed (function auto-arming delay),
- 66 - Too long time to planned arming,
- 67 - Elevator call rejected because zone is armed,
- 68 - Access disabled by external signal,
- 69 - Floor doesn't exist,
- 70 – Access Point not ready,
- 71 – Access Door taken by another Access Point,
- 72 – Access Credential Awaited Elsewhere,
- 91 - No Authorisation for Place of Action,
- 92 - Lock disabled,
- 101 - Object Set,
- 102 - Object Already Set,
- 103 - Object Cleared,
- 104 - Object Already Cleared,
- 111 - Lock Pulse Enable: Normal Pulse,
- 112 - Lock Pulse Enable: Extended Pulse,
- 113 - Lock Pulse Enable: Unlimited Pulse,
- 114 - Lock Pulse Enable: In Pulse,
- 115 - Lock Pulse Enable: Delay,
- 116 - Lock Pulse Enable: Lock On,
- 117 - Lock Pulse Enable: Lock Off,
- 118 - Lock Pulse Enable: Conditional Unblocked,
- 121 - Automation Node Delay: State changed,
- 122 - Automation Node Delay: State not changed,
- 123 - Automation Node Set Timed: State changed,
- 124 - Automation Node Set Timed: State not changed,
- 125 - Automation Node Set Permanent: State changed,
- 126 - Automation Node Set Permanent: State not changed,
- 127 - Automation Node Clear: State changed,
- 128 - Automation Node Clear: State not changed,
- 131 - Auto-arming Delay: Zone Armed,
- 132 - Auto-arming Delay:Time Long Enough,
- 133 - No Auto-arming Delay In Progress,
- 134 - Auto-arming Delay: Signaling From Parent,

- 135 - Auto-arming Local: Timer Changed,
- 136 - Auto-arming Delay: Done

List of Details

- Event Code: 13 - Authentication Factor Read
  Known Authentication factor:
  Authentication factor ID,

  Example:
  0000000003,

  Unknown Authentication factor:
  ,Authentication factor value

  Example:
  ,0000000B000B8F12

- Event Code: 601 - Door Access Granted
  Details:
  NO- Normal,
  LO- Lock On,
  LF- Lock Off.

- Event Code: 602 - Door Access Denied
  Details:
  NO- Normal,
  LO- Lock On,
  LF- Lock Off.

- Event Code: 603 – Event Access Point Entry Random Personal Check
  Details: Output ID.

EventFilterData Object Type

Event Log Entry Data Object Type Properties	
ID	Unique ID of Event Filter.
Name	Unique Name of Event Filter.

TimeRangeType	Type of time range (Time Range Types). Filtering by time is based on a value of LoggedOn property of Event Log Entry.
TimeRangeStart	If TimeRangeType=Custom, Event Log Entries are get starting from this DateTime.
TimeRangeEnd	If TimeRangeType=Custom, Event Log Entries are get up to this DateTime.
LimitResult	Number indicating how many Event Log Entries can be retrieved for one query.
Severities	Comma separated string with Severities Codes. (Severities Codes). Empty means: Any.
Events	Comma separated string with Events Codes. (Events Codes). Empty means. Any.
Networks	Comma separated string with Networks Ids. Empty means: Any.
Controllers	Comma separated string with Access Controllers Ids. Empty means: Any.
VirtualControllers	Comma separated string with Virtual Controllers Ids. Empty means: Any.
Categories	Comma separated string with Event Categories Ids. Empty means: Any.
Locations	Comma separated string with some of Object Types (Object Types) that could be a Location of Event Log Entry. Empty means: Any.
Points	Comma separated string with Access Points Ids. Empty means: Any. (connected with **Locations**)
Doors	Comma separated string with Access Doors Ids. Empty means: Any. (connected with **Locations**)
AccessZones	Comma separated string with Access Zones Ids. Empty means: Any. (connected with **Locations**)
AlarmZones	Comma separated string with Alarm Zones Ids. Empty means: Any. (connected with **Locations**)

AutomationPoints	Comma separated string with Automation Points (Automation Nodes) Ids. Empty means: Any. (connected with **Locations**)
Sources	Comma separated string with some of Object Types (Object Types) that could be a Source of Event Log Entry. Empty means: Any.
PointSources	Comma separated string with Access Points Ids. Empty means: Empty means: Any. (connected with **Sources**)
Inputs	Comma separated string with Inputs Ids. Empty means: Empty means: Any. (connected with **Sources**)
FunctionKeys	Comma separated string with Function Keys Ids. Empty means: Empty means: Any. (connected with **Sources**)
Users	Comma separated string with some of Object Types (Object Types) that could be a User for Event Log Entry. Empty means: Any.
Persons	Comma separated string with Access User Persons Ids. Empty means: Any. (connected with **Users**)
Assets	Comma separated string with Access User Assets Ids. Empty means: Any. (connected with **Users**)
Groups	Comma separated string with Access User Groups Ids. Empty means: Any. (connected with **Users**)
Details	This property may contain some additional information about Event Log Entry.
IsSystemFilter	Indicates whether Event Filter is one of predefined filters.

Filtering logic

[Time] AND Events AND Severities AND Categories AND Details AND (Networks OR Controllers) AND VirtualControllers AND Locations (Points OR Doors OR AccessZones OR AlarmZones OR AutomationPoints) AND Sources (PointSources OR Inputs OR FunctionKeys) AND Users (Persons OR Assets OR Groups)

Time Range Types

- 0: Anytime
- 1: Last hour
- 2: Last 12 hours
- 3: Last 24 hours
- 4: Last 7 days
- 5: Last 30 days
- 6: Custom
- 7: Current Week
- 8: Last Week
- 9: Current Month
- 10: Last Month

Severities

- 0: Low
- 1: Medium
- 2: High
- 3: Critical

GetAllEntries

Function was deleted, please use GetLasEntryId and TakeEntriesStartingFrom instead.

GetEntriesBetweenDates

Get Event Log entries from specified data and time range.

Method signature:

```
IEnumerable<EventLogEntryData> GetEntriesBetweenDates(DateTime startDateTime,
DateTime endDateTime, Guid sessionToken)
```

Where,

- startDateTime – Start date and time
- endDateTime – End date and time

- sessionToken – Current session token, obtained by Connect method execution

Result:
List of Event Log entries from specified data and time range. Limit of records that function returns equal 100000.

TakeEntriesStartingFrom

Take Event Log entries starting from specified entry ID.

Method signature:
```
IEnumerable<EventLogEntryData> TakeEntriesStartingFrom(int entryId, Guid sessionToken)
```
Where,

- entryId – Starting Event Log Entry ID
- sessionToken – Current session token, obtained by Connect method execution

Result:
List of all Event Log entries starting from specified entry ID. Limit of records that function returns equal 100000.

GetEntriesByFilterId

Get Event Log entries by ID of the existing Event Filter.

Method signature:
```
IEnumerable<EventLogEntryData> GetEntriesByFilterId(int filterId, Guid sessionToken)
```
Where,

- filterId – ID of the existing Event Filter
- sessionToken – Current session token, obtained by Connect method execution

Result:
List of all Event Log entries obtained after filtering.

GetEntriesByFilter

Get Event Log entries by Event Filter Data.

Method signature:

```
IEnumerable<EventLogEntryData> GetEntriesByFilter(EventFilterData filter, Guid
sessionToken)
```

Where,

- filter – object Event Filter Data
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Event Log entries obtained after filtering.

GetEntriesViewDataByFilter

Get Event Log entries by Event Filter Data.

Method signature:

```
IEnumerable<EventLogEntryViewData> GetEntriesViewDataByFilter(
EventFilterData filter, Guid sessionToken)
```

Where,

- filter – object Event Filter Data
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Event Log entries obtained after filtering.

GetEntryById

Get Event Log entry by its ID.

Method signature:

```
EventLogEntryData GetEntryById(int entryId, Guid sessionToken)
```

Where,

- entryId – ID of the requested person
- sessionToken – Current session token, obtained by Connect method execution

Result:

If Event Log entry exist in the system, then entry data is returned.

If Event Log entry not exist in the system, then NULL is returned.

GetLastEntryId

Get last registered Event Log entry ID.

Method signature:

```
int GetLastEntryId(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

If any Event Log entry exist in the system, then last registered entry ID is returned.

Else 0 is returned.

GetAllEvents

Get all Event Types existing in the system.

Method signature:

```
IEnumerable<EventData> GetAllEvents(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Event Types existing in the system.

GetAllEventCategories

Get all Event Categories existing in the system.

Method signature:

```
IEnumerable<EventLogEntryCategoryData> GetAllEventCategories(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Event Categories existing in the system.

GetAllEventFilters

Get all Event Filters existing in the system.

Method signature:

```
IEnumerable<EventFilterData> GetAllEventFilters(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Event Filters existing in the system.

System Synchronization

Common command results

All of partial Credential synchronization command can return one of below code as a result.

000: Synchronization completed successfully.
255: Unspecified error occurred.
257: Configuration sending error. It could be caused by communication problems. If occurred, please try again.
258: Configuration processing error. If occurred, please try again. If that not help then full system synchronization is required (use VISO application/or RunConfigurationSynchronization command for that).
259: Communication server unavailable.
261: Partial synchronization not available at this moment. Full synchronization already in progress on other station/process.
262: Full system synchronization is required (use VISO application/or RunConfigurationSynchronization command for that).

PartialCredentialSynchronization

Send the specified Access Credential to physical access controllers. This command should be executed after creation or modification Access Credential objects or its dependencies.

Method signature:

```
int PartialCredentialSynchronization(int credentialId, Guid sessionToken)
```

Where,

- credentialId – ID of the Access Credential to send
- sessionToken – Current session token, obtained by Connect method execution

Result:

One of available synchronization command results. Configuration of all required controllers need to ended with success for global success result. Otherwise one of synchronization error result is returned.

PartialCredentialsSynchronization

Send the Access Credentials list to physical access controllers. This command should be executed after creation or modification Access Credential objects or its dependencies.

Method signature:

```
int PartialCredentialsSynchronization(int[] credentialIds, Guid sessionToken)
```

Where,

- credentialIds – Access Credential identifiers list
- sessionToken – Current session token, obtained by Connect method execution

Result:

One of available synchronization command results. Configuration of all required controller need to ended with success for global success result. Otherwise one of synchronization error result is returned.

PartialCredentialsSynchronizationWithDetailResult

Send the Access Credentials list to physical access controllers and get more detailed result (separately for each controller). This command should be executed after creation or modification Access Credential objects or its dependencies.

Method signature:

```
PartialSynchronizationResult[]
PartialCredentialsSynchronizationWithDetailResult(int[] credentialIds, Guid
sessionToken)
```

Where,

- credentialIds – Access Credential identifiers list
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of controllers that has been synchronized, with it's separate operation result. If none data is returned, that means that specified credentials not required synchronization of any controller.

PartialSynchronizationResult Object Type

Partial Synchronization Result Object Type Properties	
AccessControllerID	ID of Access Controller.
ResultCode	One of available synchronization command results (Common command results)
Details	Additional text information

General Integration

This service contains two general methods that allow you to quickly register or disable RACS 5 user credential.

GetAttendanceZoneOccupancy

Get current occupancy of the specified Attendance Zone.

Method signature:

```
int GetAttendanceZoneOccupancy(int attendanceZoneId, Guid sessionToken)
```

Where,

- attendanceZoneId – ID of Attendance Zone
- sessionToken – Current session token, obtained by Connect method execution

Result:

Returns current occupancy of the specified Attendance Zone.

If any errors occurred, then -1 is returned.

GetAttendanceZoneOccupancies

Get list of person present in the specified Attendance Zone.

Method signature:

```
IList<PersonData> GetAttendanceZoneOccupancies(int attendanceZoneId, Guid sessionToken)
```

Where,

- attendanceZoneId – ID of Attendance Zone
- sessionToken – Current session token, obtained by Connect method execution

Result:

Returns list of person present in the specified Attendance Zone.

GetAccessZoneOccupancy

Get current occupancy of the specified Access Zone.

Method signature:

```
int GetAccessZoneOccupancy(int accessZoneId, int monitoringTime, Guid sessionToken)
```

Where,

- accessZoneId – ID of Access Zone
- monitoringTime – back monitoring time in hours. E.g. if 24 will be specified, then only events from last 24 hours will be used during calculations. If all events should be take into account then use 0 as a value of this parameter
- sessionToken – Current session token, obtained by Connect method execution

Result:

Returns current occupancy of the specified Access Zone.

If any errors occurred, then -1 is returned.

GetPersonLastAccessPoint

Get the information about last granted physical access on Access Point for specified Person.

Method signature:

```
AccessData GetPersonLastAccessPoint(int personId, Guid sessionToken)
```

Where,

- personId – ID of Person
- sessionToken – Current session token, obtained by Connect method execution

Result:

If any physical access granted event found, AccessData for last Access Point is returned.

If none physical access granted found, then NULL is returned.

GetAssetLastAccessPoint

Get the information about last granted physical access on Access Point for specified Asset.

Method signature:

```
AccessData GetAssetLastAccessPoint(int assetId, Guid sessionToken)
```

Where,

- assetId – ID of Asset
- sessionToken – Current session token, obtained by Connect method execution

Result:

If any physical access granted event found, AccessData for last Access Point is returned.

If none physical access granted found, then NULL is returned.

Communication

This service allow to communicate with physical access controller.

RemoteCommand

Allow to execute remote command on the access controller.

Method signature:

```
int RemoteCommand(short objectType, int objectId, int functionCode, string
functionParameterValue, bool raiseEvents, bool checkRights, bool
checkObjectRights, bool checkParameterRights, int credentialId,
Guid sessionToken)
```

Where,

- objectType – type of object on which function will be executed (accessible object types are described below)
- objectId – ID of object on which function will be executed
- functionCode – code of function that will be executed (accessible function codes for object types are described below)
- functionParameterValue – value of function additional parameter (optional)
- raiseEvents – defines if events should be raised
- checkRights – defines if user authentication is required; this parameter will be automatically set to 'true' if checkObjectRights or checkParameterRights is 'true'
- checkObjectRights – defines if authorization for place of action is required
- checkParameterRights – defines if authorization for function parameter is required
- credentialId – ID of Access Credential that belongs to Access User Person assigned to logged Operator (optional, if 0 will be passed, then first Access Credential will be used)
- sessionToken - Session token returned by Connect function

Object types and functions:

- 1030 – Access Zone
  - 111 – Clear occupancy register
  - 112 – Reset Pass-back register
- 1031 – Access Point
  - 151 – Grant Physical Access with Normal Lock Pulse
  - 152 – Grant Physical Access with Extended Lock Pulse
  - 153 – Set T&A Mode Momentary (required an parameter: ID of Time Attendance Mode)

  - 154 – Set T&A Mode Maintained (required an parameter: ID of Time Attendance Mode)
  - 155 – Register T&A Event (required an parameter: ID of Time Attendance Mode)
  - 156 – Set Authentication Policy (required an parameter: ID of Authentication Policy)
  - 159 – Signal Door Bell on Access Point
  - 171 – Register Guard Tour Event
  - 172 – Register Trace Event
  - 173 – Register Muster Event
  - 174 – Register Duress Event
- 1042 – Access Door
  - 121 – Unblock Door in Emergency Mode
  - 122 – Block Door in Emergency Mode
  - 123 – Clear Emergency Door Control Mode
  - 124 – Set Blocked Door Mode
  - 125 – Set Unblocked Door Mode
  - 126 – Set Normal Door Mode
  - 127 – Set Conditional-unblocked Door Mode
  - 128 – Generate Normal Door Lock Pulse
  - 129 – Generate Extended Door Lock Pulse
  - 131 – Signal Door Bell on Access Door
  - 134 – Clear Forced Entry Signaling
  - 135 – Clear Door Open Too Long Signaling
- 1069 – Alarm Zone
  - 102 – Switch Armed Mode ON/OFF
  - 103 – Set Armed Mode ON
  - 104 – Set Armed Mode OFF
  - 106- Postpone Auto-arming
- 1077 – Automation Node
  - 161 – Set Automation Node ON Instant
  - 162 – Set Automation Node ON Timed
  - 163 – Set Automation Node OFF Instant
  - 164 – Switch Automation Node ON/OFF Instant
  - 165 – Switch Automation Node ON/OFF Timed

Result:

000: Remote command was successfully sent to access controller

040: Access Controller determined by object type and object ID not exist

041: Unknown function code

042: Function is not supported at specified context

043: Required function parameter has not been specified
200: Invalid current session token
210: Current logged Operator is not assigned to Access User Person
211: Access User Person to which current logged Operator is assigned does not have Access Credentials
255 : Unspecified error occurred (use GetLastErrorMessage function to see details)
257 : Communication error
258 : Controller processing error (use GetLastErrorMessage function to see details)
259: Communication service is unavailable (run communication service and try again)

SetControllerDateTime

Allow to set clock on selected Access Controller

Method signature:

```
int SetControllerDateTime(int controllerId, DateTime dateTime, Guid sessionToken)
```

Where,

- objectType – type of object on which function will be executed (accessible object types are described below)
- objectId – ID of object on which function will be executed
- sessionToken - Session token returned by Connect function

Result:
000: Clock has been successfully set on selected controller
200: Invalid current session token
210: Current logged Operator is not assigned to Access User Person
211: Access User Person to which current logged Operator is assigned does not have Access Credentials
255 : Unspecified error occurred (use GetLastErrorMessage function to see details)
257 : Communication error or selected controller is unreachable
258 :Controller processing error (use GetLastErrorMessage function to see details)
259: Controller configuration processing error
260: Communication service is unavailable (run communication service and try again)

Communication Process Types

0: Unknown

1: Event download process

4: Controllers time synchronization process

8: Perimeter Zones monitoring process

32: Multi-level task execution process

256: Automation Nodes monitoring process

512: Alarm Zones monitoring process

1028*: External and Machine Authorisation monitoring process

2056*: Hotel Rooms monitoring process

16384: Notifications monitoring process

32768: Visitors monitoring process

65536: Global Access Zone monitoring process

**1028,2056 are correct values, although they do not match the power of 2 schema.*

The other processes should not be controlled from Web Api.

RunProcess

Allow to run communication process.

Method signature:

```
string RunProcess(int processId, Guid sessionToken)
```

Where,

- processId – ID of process type to run
- sessionToken - Session token returned by Connect function

Result:

000: Process was successfully run

255: Unspecified error occurred (use GetLastErrorMessage function to see details)

259: Communication service is unavailable (run communication service and try again)

StopProcess

Allow to stop communication process.

Method signature:

```
string StopProcess(processId, Guid sessionToken)
```

Where,

- processId – ID of process type to stop

- sessionToken - Session token returned by Connect function

Result:
000: Process was successfully run
255: Unspecified error occurred (use GetLastErrorMessage function to see details)
259: Communication service is unavailable (run communication service and try again)

IsProcessActive

Allow to check if process is active.

Method signature:

```
string IsProcessActive(processId, Guid sessionToken)
```

Where,

- processId – ID of process type
- sessionToken - Session token returned by Connect function

Result:
000: Process is inactive
001: Process is active
255: Unspecified error occurred (use GetLastErrorMessage function to see details)
259: Communication service is unavailable (run communication service and try again)

RunGlobalCommand

Allow to run Global command.

Method signature:

```
int RunGlobalCommand(int commandId, int credentialId, Guid sessionToken)
```

Where,

- commandId – ID of Global Command to execute
- credentialId – ID of credential that permissions will be used
- sessionToken - Session token returned by Connect function

Result:
If Task are successfully registered then it's ID is returned (greater than 0). You can trace the task by this ID.
If any errors occurs or the Task Execution Process is inactive then error code (less than 0) is returned.

Possible error codes:
-200: Invalid session token
-210: Session operator is not link with person
-211: Session operator linked person has not credentials
-212: Specified credential ID not belongs to current session operator
-213: Global command not exist or have not any commands to execute
-255: Unknown error
-259: Communication service is unavailable
You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder.

RegisterGlobalCommand

Allow to register Global command to execute in the future.

Method signature:

```
int RegisterGlobalCommand(int commandId, int credentialId, DateTime executionTime, Guid sessionToken)
```

Where,

- commandId – ID of Global Command to execute
- credentialId – ID of credential that permissions will be used
- executionTime – date and time when Global command will be executed
- sessionToken - Session token returned by Connect function

Result:
If Task are successfully registered then it's ID is returned (greater than 0). You can trace the task by this ID.
If any errors occurs or the Task Execution Process is inactive then error code (less than 0) is returned.

Possible error codes:
-200: Invalid session token
-210: Session operator is not link with person
-211: Session operator linked person has not credentials
-212: Specified credential ID not belongs to current session operator
-213: Global command not exist or have not any commands to execute
-255: Unknown error
-259: Communication service is unavailable

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder.

RunConfigurationSynchronization

Allow to run system configuration synchronization.

Method signature:

```
int RunConfigurationSynchronization(Guid sessionToken)
```

Where,

- sessionToken - Session token returned by Connect function

Result:

If Task are successfully registered then it's ID is returned (greater than 0). You can trace the task by this ID.

If any errors occurs or the Task Execution Process is inactive then error code (less than 0) is returned.

Possible error codes:

-255: Unknown error

-259: Communication service is unavailable

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder.

RegisterConfigurationSynchronization

Allow to register system configuration synchronization task to execute in the future.

Method signature:

```
int RegisterConfigurationSynchronization(DateTime executionTime, Guid
sessionToken)
```

Where,

- executionTime – date and time when configuration synchronization will be executed
- sessionToken - Session token returned by Connect function

Result

If Task are successfully registered then it's ID is returned (greater than 0). You can trace the task by this ID.

If any errors occurs or the Task Execution Process is inactive then error code (less than 0) is returned.

Possible error codes:
-255: Unknown error
You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder.

GetTaskState

Allow to get running or registered task state.

Method signature:

```
int GetTaskState(int taskId, Guid sessionToken)
```

Where,

- taskId – ID of task
- sessionToken - Session token returned by Connect function

Result:
-1: Task with specified ID not exist or other error occurs
000: Registered (wait for execution)
001: In progress
002: Completed with success
004: Completed with error
008: Rejected (process was busy)
016: Expired (task has not been executed and expired)

GetTaskProgress

Allow to get running task progress.

Method signature:

```
int GetTaskProgress(int taskId, Guid sessionToken)
```

Where,

- taskId – ID of task
- sessionToken - Session token returned by Connect function

Result:
-1: Task with specified ID not exist or other error occurs

0 - 100: Task progress (%)

GetTaskLogs

Allow to get detailed task execution log.

Method signature:

```
IEnumerable<LongRunningTaskLogData> GetTaskLogs(int taskId, Guid sessionToken)
```

Where,

- taskId – ID of task
- sessionToken - Session token returned by Connect function

Result:

List of **LongRunningTaskLogData** objects.

LongRunningTaskLogData Properties	
ID	ID of task log entry
Timestamp	Date and time
Type	Level { 0 – Info, 1 – Warning, 2 – Error, 3 – Critical }
Message	Detailed message

If task with specified ID not exist or error occurs then empty list is returned.

GrantRemoteAuthorization

Grant Remote Authorization request.

Method signature:

```
bool GrantRemoteAuthorization(int requestId, string processName, Guid
sessionToken)
```

Where,

- requestId – Remote Authorization request ID
- processName – name of process that call the service it should be name of client application
- sessionToken – Current session token, obtained by Connect method execution

Result:

Return True if operation ends with success. Else return false.

DenyRemoteAuthorization

Deny Remote Authorization request.

Method signature:

```
bool DenyRemoteAuthorization(int requestId, string processName, Guid
sessionToken)
```

Where,

- requestId – Remote Authorization request ID
- processName – name of process that call the service it should be name of client application
- sessionToken – Current session token, obtained by Connect method execution

Result:

Return True if operation ends with success. Else return false.

TakeRemoteAuthorizationStartingFrom

Take Remote Authorization requests starting from specified ID.

Method signature:

```
IEnumerable<RemoteAuthorizationRequestData>
TakeRemoteAuthorizationRequestStartingFrom(int requestId, Guid sessionToken)
```

Where,

- requestId – Starting Remote Authorization request ID
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Remote Authorization request starting from specified entry ID.

GetRemoteAuthorizationRequestBetweenDates

Get Remote Authorizationrequests between specified dates.

Method signature:

```
IEnumerable<RemoteAuthorizationRequestData>
GetRemoteAuthorizationRequestBetweenDates(DateTime startDateTime, DateTime
endDateTime, Guid sessionToken)
```

Where,

- startDateTime – start date and time
- endDateTime – end date and time

- sessionToken – Current session token, obtained by Connect method execution

Result:
List of all Remote Authorization request from specified data and time range.

GetRemoteAuthorizationRequestByState

Get Remote Authorization requests by state.

Method signature:

```
int GetLastRemoteAuthorizationRequestId (byte state, Guid sessionToken)
```

Where,

- state – state of Remote Authorization request
  - 0 – Pending
  - 1 – Processed Granted
  - 2 – Processed Denied
  - 4 – Not Processed
  - 8 – Error
- sessionToken – Current session token, obtained by Connect method execution

Result:
List of all Remote Authorization request with specified state.

GetLastRemoteAuthorizationRequestId

Get last Remote Authorization request ID.

Method signature:

```
int GetLastRemoteAuthorizationRequestId (Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:
If any Remote Authorization request exist in the system, then last registered request ID is returned.
Else 0 is returned.

SetCredentialUsesRemaining

Allow to set new value of UsesRemaining.

Method signature:

```
int SetCredentialUsesRemaining(int zoneId, int credentialId, int? usesRemaining, Guid sessionToken)
```

Where,

- zoneId – ID of Access Zone
- credentialId - ID of Access Credential
- usesRemaining – new value 0-255 (255 means No limit)

Result

If any errors occurs then error code (less than 0) is returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

SetCredentialDaysRemaining

Allow to set new value of DaysRemaining.

Method signature:

```
int SetCredentialDaysRemaining(int zoneId, int credentialId, int? daysRemaining, Guid sessionToken)
```

Where,

- zoneId – ID of Access Zone
- credentialId - ID of Access Credential
- daysRemaining – new value 0-255 (255 means No limit)

Result

If any errors occurs then error code (less than 0) is returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

SetCredentialAbsenteeTime

Allow to set new value of AbsenteeTime.

Method signature:

```
int SetCredential AbsenteeTime (int zoneId, int credentialId, int? absenteeTime, Guid sessionToken)
```

Where,

- zoneId – ID of Access Zone
- credentialId - ID of Access Credential

- AbsenteeTime – new value 0-127 (127 means No limit)

Result

If any errors occurs then error code (less than 0) is returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetCredentialUsesRemaining

Get value of UsesRemaining for given ids of AccessZone and AccessCredential.

Method signature:

```
int? GetCredentialUsesRemaining(int zoneId, int credentialId, Guid sessionToken)
```

Where,

- zoneId – ID of Access Zone
- credentialId - ID of Access Credential

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetCredentialDaysRemaining

Get value of DaysRemaining for given ids of AccessZone and AccessCredential.

Method signature:

```
int? GetCredentialDaysRemaining(int zoneId, int credentialId, Guid sessionToken)
```

Where,

- zoneId – ID of Access Zone
- credentialId - ID of Access Credential

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetCredentialAbsenteeTime

Get value of AbsenteeTime for given ids of AccessZone and AccessCredential.

Method signature:

```
int? GetCredentialAbsenteeTime (int zoneId, int credentialId, Guid sessionToken)
```

Where,

- zoneId – ID of Access Zone
- credentialId - ID of Access Credential

Result

If any errors occurs then result is null returned.

You can check the last error details by executing method GetLastErrorMessage or by checking log file on Logs\Integration folder

GetLastErrorMessage

Allow to get the last error details.

Method signature:

```
string GetLastErrorMessage()
```

Result:

Last error details.

System Reporting Service

The service allow to monitor system objects state.

GetAccessDoorStatesWhichLatestUpdateIsSince

Get state of Access Doors that changed its state since the specified date and time.

Method signature:

```
IEnumerable<AccessDoorStateData> GetAccessDoorStatesWhichLatestUpdateIsSince
   (DateTime sinceTime, Guid sessionToken)
```

Where,

- sinceTime – date and time
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Door states.

GetAccessDoorStatesWhichLatestUpdateIsBetween

Get state of Access Doors that changed its state between the specified date and time range.

Method signature:

```
IEnumerable<AccessDoorStateData> GetAccessDoorStatesWhichLatestUpdateIsBeetween
(DateTime fromTime, DateTime toTime, Guid sessionToken)
```

Where,

- fromTime – from date and time
- toTime – to date and time
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Access Door states.

DoorLockMode values:

- 0 - Unknown
- 1 - Normal
- 2 - Locked
- 3 - Emergency Locked
- 4 - Unlocked
- 5 - Emergency Unlocked

- 6 - Conditional Unlocked
- 7 - Door Blocked

DoorOpenState values:

- 0 - Unknown
- 1 - Closed
- 2 - Open
- 3 - Forced Open

GetAutomationPointStatesWhichLatestUpdateIsSince

Get state of Automation Points that changed its state since the specified date and time.

Method signature:

```
IEnumerable<AutomationPointStateData>
GetAutomationPointStatesWhichLatestUpdateIsSince
(DateTime sinceTime, Guid sessionToken)
```

Where,

- sinceTime – Since date and time
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Automation Point states.

GetAutomationPointStatesWhichLatestUpdateIsBetween

Get state of Automation Points that changed its state between the specified date and time range.

Method signature:

```
IEnumerable<AutomationPointStateData>
GetAutomationPointStatesWhichLatestUpdateIsBeetween
(DateTime fromTime, DateTime toTime, Guid sessionToken)
```

Where,

- fromTime – from date and time
- toTime – to date and time
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Automation Point states.

SwitchingState values:

- 0 - Unknown
- 1 - Off
- 2 - On

GetAlarmZoneStatesWhichLatestUpdateIsSince

Get state of Alarm Zones that changed its state since the specified date and time.

Method signature:

```
IEnumerable<AlarmZoneStateData> GetAlarmZoneStatesWhichLatestUpdateIsSince
(DateTime sinceTime, Guid sessionToken)
```

Where,

- sinceTime – date and time
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Alarm Zone states.

GetAlarmZoneStatesWhichLatestUpdateIsBetween

Get state of Alarm Zones that changed its state between the specified date and time.

Method signature:

```
IEnumerable<AlarmZoneStateData> GetAlarmZoneStatesWhichLatestUpdateIsBeetween
(DateTime fromTime, DateTime toTime, Guid sessionToken)
```

Where,

- fromTime – From date and time
- toTime – To date and time
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Alarm Zone states.

ArmingState values:

- 0 - Unknown
- 1 - Disarmed
- 2 - Armed

GetAccessGrantedLogViewDataForUsersInsideAttendanceZone

Get list of users that last access granted was inside the specified Attendance Zone.

Method signature:

```
IEnumerable<AccessGrantedLogViewData>
GetAccessGrantedLogViewDataForUsersInsideAttendanceZone
(int attendanceZoneId, Guid sessionToken)
```

Where,

- attendanceZoneId – ID of Attendance Zone
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Users inside Attendance Zone.

GetAccessGrantedLogViewDataForUsersOutsideAttendanceZone

Get list of users that last access granted was outside the specified Attendance Zone.

Method signature:

```
IEnumerable<AccessGrantedLogViewData>
GetAccessGrantedLogViewDataForUsersOutsideAttendanceZone
(int attendanceZoneId, Guid sessionToken)
```

Where,

- attendanceZoneId – ID of Attendance Zone
- sessionToken – Current session token, obtained by Connect method execution

Result:

List of Users outside Attendance Zone.

GetAccessGrantedLogs

Get information about the last access granted location for all users.

Method signature:

```
IEnumerable<AccessGrantedLogViewData> GetAccessGrantedLogs
(Guid sessionToken)
```

Where,

- sessionToken – Current session token, obtained by Connect method execution

Result:

List of all Users with last access granted location information.

Contact:

Roger Sp. z o.o sp. k.

82-400 Sztum

Gościszewo 59

Tel.: +48 55 272 0132

Fax: +48 55 272 0133

Tech. support: +48 55 267 0126

E-mail: biuro@roger.pl

Web: www.roger.pl