Roger Access Control System

Integration of RACS 5 Through Web Services API

Software version: 1.3

Document version: Rev. C

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 executer 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

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.

GetSessionCultureName

Get logged in operator culture name.

Method signature:

```
OperatorData GetSessionCultureName(Guid sessionToken)
```

Where,

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

Result:

Logged in operator culture name.

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> GetCredentials(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.

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.

GetAssetSnapshots

Get all Access User Asset Snapshots existing in the system. It can be used to analyze Event Log data where snapshots are used.

Method signature:

```
IEnumerable<AssetSnapshotData> GetAssetSnapshots(Guid sessionToken)
```

Where,

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

Result:
List of Access User Asset Snapshots 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.

GetGroupSnapshots

Get all Access User Group Snapshots existing in the system. It can be used to analyze Event Log data where snapshots are used.

Method signature:

```
IEnumerable<GroupSnapshotData> GetGroupSnapshots(Guid sessionToken)
```

Where,

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

Result:
List of Access User Group Snapshots 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.

GetPersonSnapshots

Get all Access User Person Snapshots existing in the system. It can be used to analyze Event Log data where snapshots are used.

Method signature:

```
IEnumerable<PersonSnapshotData> GetPersonSnapshots(Guid sessionToken)
```

Where,

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

Result:
List of Access User Person Snapshots existing in the system.

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.

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.

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.

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.

GetPersonRightsView

Get Access Rights identifiers 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 identifiers 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 identifiers 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.

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.

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.

AssignCredentials

Allow assign Access Credentials to Access User Person.

Method signature:

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

Where,

- personId – ID of the Access User Person to which the credentials will be assigned
- credentialIds – List of the Access Credential identifiers that will be assigned to specified person
- 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.

Method signature:

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

Where,

- personId – ID of the Access User Person 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.

Method signature:

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

Where,

- personId – ID of the Access User Person to which the rights will be assigned

- accessRigthsIds – List of the Access Rights identifiers that will be assigned to specified Access User Person
- 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.

Method signature:

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

Where,

- personId – ID of the Access User Person 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 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.

Method signature:

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

Where,

- personId – ID of the Access User Person 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 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.

Method signature:

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

Where,

- personId – ID of the Access User Person 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 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

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.

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.

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:

```
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:

`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:
`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:
`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.

GetLastErrorMessage

Allow to get the last error details.

Method signature:

```
string GetLastErrorMessage()
```

Result:

Last error details.

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.
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 - Entry Access Confirmed
- 3 – Entry Access Not Confirmed
- 6 - Access Point Lockout
- 8 - Access Point Lockout End
- 13 - Authentication Factor Read
- 261- Lock Switched On
- 262 - Lock Switched Off
- 302 - Unlocked Door Mode
- 321 - Forced Door Open Alarm
- 322 - Door Open Too Long Alarm
- 601 - Door Access Granted
- 602 - Door Access Denied
- 604 - Pass-back Violation
- 605 - Elevator Access Granted
- 606 - Elevator Access Denied
- 607 - Disable Door Access Granted
- 608 - Disable Door Access Denied
- 609 - Disable Door Access Clear Granted
- 610 - Disable Door Access Clear Denied
- 611 - Maintained T&A Mode Granted
- 612 - Maintained T&A Mode Denied
- 613 - Momentary T&A Mode Granted
- 614 - Momentary T&A Mode Denied
- 615 - T&A Event
- 616 - T&A Event Denied
- 617 - Authentication Policy Change Granted
- 618 - Authentication Policy Change Denied
- 619 - Door Bell on Access Point Granted
- 620 - Door Bell on Access Point Denied

- 621 - Duress Event
- 622 - Duress Event Denied
- 623 - Trace Event
- 624 - Trace Event Denied
- 625 - Guard Tour Event
- 626 - Guard Tour Event Denied
- 627 - Muster Event
- 628 - Muster Event Denied
- 631 - Emergency Lock On Granted
- 632 - Emergency Lock On Denied
- 633 - Emergency Lock Off Granted
- 634 - Emergency Lock Off Denied
- 635 - Emergency Lock Control Cancel Granted
- 636 - Emergency Lock Control Cancel Denied
- 637 - Normal Door Mode Granted
- 638 - Normal Door Mode Denied
- 639 - Locked Door Mode Granted
- 640 - Locked Door Mode Denied
- 641 - Unlocked Door Mode Granted
- 642 - Unlocked Door Mode Denied
- 643 - Conditional Unlocked Door Mode Granted
- 644 - 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 Signaling Cancel Granted
- 650 - Door Open Too Long Alarm Signaling Cancel Denied
- 651 - Automation Node Set Instant Granted
- 652 - Automation Node Set Instant Denied
- 653 - Automation Node Set Timed Granted
- 654 - Automation Node Set Timed Denied
- 655 - Automation Node Clear Instant Granted
- 656 - Automation Node Clear Instant Denied
- 671 - Armed Mode
- 672 - Armed Mode Denied
- 673 - Armed Mode Off Granted
- 674 - Armed Mode Off Denied

- 681 - Postpone Auto-arming Granted
- 682 - Postpone Auto-arming Denied
- 683 - Disable Arming Granted
- 684 - Disable Arming Denied
- 685 - Disable Arming Cancel Granted
- 686 - Disable Arming Cancel Denied
- 691 - Lock Toggled On Granted
- 692 - Lock Toggled On Denied
- 693 - Toggle Lock Off Granted
- 694 - Toggle Lock Off Denied
- 695 - Toggle Lock Control Cancel Granted
- 696 - Toggle Lock Control Cancel Denied
- 697 - Door Forced Alarm Signaling Cancel Granted
- 698 - Door Forced Alarm Signaling Cancel Denied
- 701 - Occupancy Register Clear Granted
- 702 - Occupancy Register Clear Denied
- 703 - Pass-back Register Clear Granted
- 704 - Pass-back Register Clear 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 End
- 811 - Controller Restart Request
- 812 - Controller Restart Success
- 813 - Communication Watchdog Restart
- 814 - Communication Key Set
- 815 - Elevator Access Trouble
- 816 - Elevator Access Trouble End
- 817 - External Module Communication Lost
- 818 - External Module Communication Restored
- 819 - External Module Monitoring Stopped
- 820 - External Module Monitoring Resumed
- 821 - Time/Date Change Request
- 822 - Time/Date Changed
- 831 - Switch to New Configuration Request

- 832 - Switch to New Configuration Sucess
- 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 Statutes

- 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

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.

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

PartialCredentialSynchronization

Send the specified Access Credential to physical access controllers. This method 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:

000 : Successfully completed order

255 : Unspecified error occurred

257 : Configuration sending error (occurs if configuration sending ended with error on at least one MC16 controller). It could be caused by communication problems. If occurred, please try again.

258 : Configuration processing error (occurs if configuration process ended with error on at least one MC16 controller). If occurred, please try again. If that not help then full system synchronization is required (use VISO application for that).

PartialCredentialsSynchronization

Send the Access Credentials list to physical access controllers. This method 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:

000 : Successfully completed order

255 : Unspecified error occurred

257 : Configuration sending error (occurs if configuration sending ended with error on at least one MC16 controller). It could be caused by communication problems. If occurred, please try again.

258 : Configuration processing error (occurs if configuration process ended with error on at least one MC16 controller). If occurred, please try again. If that not help then full system synchronization is required (use VISO application for that).

General Integration

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

MakeKey

Allow you to create the RACS 5 user with specified credential data, permissions and proximity card number.

Description:

a. Deducts Access Credential to which Authentication Factor given in command belongs to.
b. Sets Access Credential Status property to Active.
c. Creates new Access User Person with name given in command.
d. Assigns newly created Access User Person to Access Credential.
e. Sets new Begin/Expiry time for Access Credential.
f. Sets new Authorisations for Access Credential.
g. Updates system settings.

Method signature:

```
int RegisterVisitor(Guid sessionToken, string firstName, string lastName,
bool clearExistingPermissionsToRoom, int roomId, int[] permissions,
DateTime from, DateTime to, string cardCode, int callType)
```

Where,

- sessionToken - Session token returned by Connect function
- firstName - Guest First Name, max 200 characters, cannot contain any control characters (e.g. end of line) and none of the characters: ,=;
- lastName - Guest Last Name, max 40 characters, cannot contain any control characters (e.g. end of line) and none of the characters: ,=;
- clearExistingPermissionToRoom - TRUE for a new key replacing previously created keys and FALSE for a copy, leaving previously created keys valid if they have not expired
- roomId - Room number
- permissions - Permission Numbers separated by commas
- from - Arrival date: yyyyMMddHHmm (201412160731)
- to - Departure date: yyyyMMddHHmm (201412160731)
- cardCode - Card code
- callType – Elevator call type (0-99)

Result:

000 : Successfully completed order
010 : Unknown card code
011: Authorization not exist
255 : Unspecified error occurred
256 : Data updating error
257 : Configuration sending error (occurs if configuration sending ended with error on at least one MC16 controller)
258 : Configuration processing error (occurs if configuration process ended with error on at least one MC16 controller)

Assumptions:
RoomToAuthorizationConfig.ini file content
105=a
1=b
9=c

Where a,b,c are identifiers of RACS5 Authorization objects that refers to room with ID=105, and permissions with ID=1 and ID=9

DisableKey

Allow to disable specified key. After execution of this function, key will be still available (it can be activated by Make Key function) but it will be inactive and not grant access to any physical location.

Description:

a. Deducts Access Credential to which the Authentication Factor given in command belongs to.
b. Removes all Authorisations from Access Credential.
c. Clear Access Credential Belongs To property.
d. Sets Access Credential Status property to Inactive.
e. Updates the system

Method signature:

```
int DisableCredential(Guid sessionToken, string cardCode)
```

Where,

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

Result:
000 : Successfully completed order.

010 : Unknown card code
020 : None assigned authorizations
255 : Unspecified error occurred
256 : Data updating error
257 : Configuration sending error (occurs if configuration sending ended with error on at least one MC16 controller)
258 : Configuration processing error (occurs if configuration process ended with error on at least one MC16 controller)

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)

Communication Process Types

000 : Unknown

001 : EventLog

002 : Watchdog

004 : Time Synchronization

008 : Perimeter Zone

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 get the last error details.

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 get the last error details.

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 RunGlobalCommand(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. Result:

Possible error codes:

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.

GetLastErrorMessage

Allow to get the last error details.

Method signature:

```
string GetLastErrorMessage()
```

Result:

Last error details.

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