Roger Gościszewo 25.07.2006
Last modified on: 14.07.2014

PR Master Application Programming Interface (API)

(for PR Master 4.5.12 or newer)

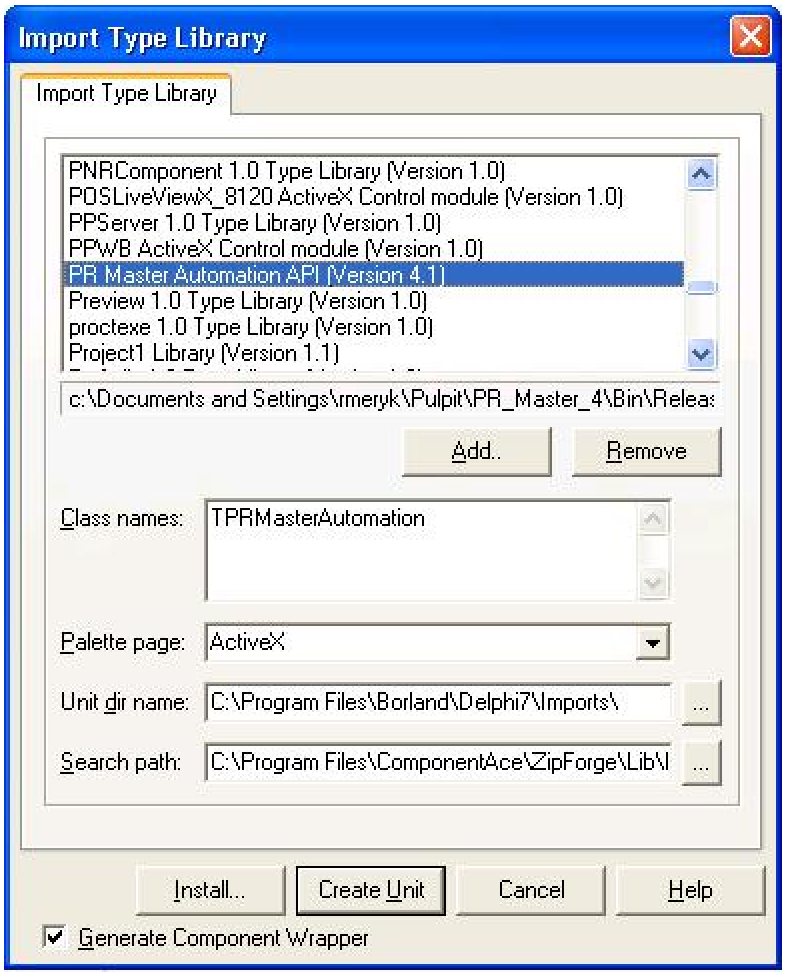
1. Introduction

The PR Master is a managing program for the *Roger Access Control System* (*RACS*). The PR Master includes API which enables following operations:

- creation, reading, updating and deleting in the RACS database
- import of events from RACS database
- sending commands to RACS system
- online (live) monitoring of the access system

The PR Master API was implemented in COM (Component Object Model) technology and operates as an *Ole Automation* unit. The PR Master API can be used only when main program (PR Master) is running.

After first installation of the PR Master the Type Library is registered in the PC's operating system, this library includes all defined API interfaces. In order to use the PR Master Automation API it must be imported to programming environment (below, you might see example how to import to Delphi).


Import Type Library

PNRComponent 1.0 Type Library (Version 1.0)
POSLiveViewX_8120 ActiveX Control module (Version 1.0)
PPServer 1.0 Type Library (Version 1.0)
PPWB ActiveX Control module (Version 1.0)
PR Master Automation API (Version 4.1)
Preview 1.0 Type Library (Version 1.0)
proctexe 1.0 Type Library (Version 1.0)
Project1 Library (Version 1.1)

c:\Documents and Settings\rmeryk\Pulpit\PR_Master_4\Bin\Releas

Class names: TPRMasterAutomation
Palette page: ActiveX
Unit dir name: C:\Program Files\Borland\Delphi7\Imports\
Search path: C:\Program Files\ComponentAce\ZipForge\Lib\I

☑ Generate Component Wrapper

The majority of the API functions are using parameters in XML format. Those XML parameters have the same structures as XML data generated by the PR Master when exporting database to XML files (PR Master menu File/Export...). Below you will find two examples of XML data.

Example 1:

function GetSchedulesXml([out] **xml**: String): Hresult;

This function returns data which contains definitions of all time schedules defined in the access system. Returned data is composed of two tables (Schedules and Time Zones) which also exist in PR Master database. For example function may return following result:

```
<tables>
        <table name="Schedules"> //beginning of a "Schedules" table
                <fields> //list of table's columns (name and type)
                        <field name="ScheduleID" type="Integer" />
                        <field name="Name" type="String" />
                        <field name="Custom1" type="String" />
                        <field name="Custom2" type="String" />
                </fields>
                <rows> //list of records in Schedules table
                        <row>
                                <f>0</f>
                                <f>-</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                        <row>
                                <f>1</f>
                                <f>Zawsze</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                        <row>
                                <f>2</f>
                                <f>Nigdy</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                        <row>
                                <f>1000</f>
                                <f>Zawsze w domyślnym Trybie Rejestracji RCP</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                        <row>
                                <f>1010</f>
                                <f>New T&amp;A Mode Schedule (1)</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                        <row>
                                <f>2000</f>
                                <f>Nigdy</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                        <row>
                                <f>2010</f>
                                <f>Nowy harmonogram Zerowania APB (1)</f>
                                <f>null</f>
                                <f>null</f>
                        </row>
                </rows>
        </table> //End of Schedule's table
        <table name="Time Zones"> //Beginning of the Time Zones table
                <fields> //List of columns (column's name and type)
                        <field name="ScheduleID" type="Integer" />
                        <field name="Day" type="Integer" />
```

```
                    <field name="StartTime" type="Time" />
                    <field name="EndTime" type="Time" />
                    <field name="Type" type="Integer" />
               </fields>
               <rows> //lista rekordów tabeli Time Zones
                    <row>
                         <f>1010</f>
                         <f>1</f>
                         <f>00:00:00</f>
                         <f>12:00:00</f>
                         <f>16</f>
                    </row>
                    <row>
                         <f>1010</f>
                         <f>1</f>
                         <f>13:00:00</f>
                         <f>14:00:00</f>
                         <f>0</f>
                    </row>
                    <row>
                         <f>2020</f>
                         <f>1</f>
                         <f>13:13:00</f>
                         <f>13:14:30</f>
                         <f>0</f>
                    </row>
               </rows>
          </table> //Endo of Time Jones table
</tables>
```

Those two tables can be exported from PR Master to external file and displayed using standard Window’s XML viewer/editor.

Schedules

ScheduleID	**Name**	**Custom1**	**Custom2**
0	-	null	null
1	Always	null	null
2	Never	null	null
1000	Always in Default T&A Mode	null	null
1010	New T&A Mode Schedule (1)	null	null
2000	Never	null	null
2010	New APB Reset schedule	null	null

Time Zones

ScheduleID	**Day**	**StartTime**	**EndTime**	**Type**
1010	1	00:00:00	12:00:00	16
1010	1	13:00:00	14:00:00	0
2020	1	13:13:00	13:14:30	0

Example 1:

function GetScheduleXml([in] scheduleID: Integer; [out] **xml**: String): Hresult;

This function returns data which contains definition of a time schedule with given scheduleID. Returned data is composed of two tables (Schedules and Time Zones) which also exist in PR Master database. For example function may return following result:

```
<tables>
    <table name="Schedules">
        <fields>
            <field name="ScheduleID" type="Integer" />
            <field name="Name" type="String" />
            <field name="Custom1" type="String" />
            <field name="Custom2" type="String" />
        </fields>
        <rows>
            <row>
                <f>1010</f>
                <f>New T&amp;A Mode Schedule (1)</f>
                <f>null</f>
                <f>null</f>
            </row>
        </rows>
    </table>
    <table name="Time Zones">
        <fields>
            <field name="ScheduleID" type="Integer" />
            <field name="Day" type="Integer" />
            <field name="StartTime" type="Time" />
            <field name="EndTime" type="Time" />
            <field name="Type" type="Integer" />
        </fields>
        <rows>
            <row>
                <f>1010</f>
                <f>1</f>
                <f>00:00:00</f>
                <f>12:00:00</f>
                <f>16</f>
            </row>
            <row>
                <f>1010</f>
                <f>1</f>
                <f>13:00:00</f>
                <f>14:00:00</f>
                <f>0</f>
            </row>
        </rows>
    </table>
</tables>
```

Those two tables can be also exported from PR Master to external file and viewed using standard Window's XML viewer/editor.

Schedules

ScheduleID	Name	Custom1	Custom2
1010	New T&A Mode Schedule (1)	null	null

Time Zones

ScheduleID	Day	StartTime	EndTime	Type
1010	1	00:00:00	12:00:00	16
1010	1	13:00:00	14:00:00	0

2. Type Library Elements

- **Constance Definition (const.):**

Below there are Input Parameters used in API functions except *ErrorCodesEnum* which are used to indicate errors.

```
{
CancelControllerAlarm = 161,
OpenDoor = 162,
SwitchToDISARM = 163,
SwitchToARM = 164,
CheckDoorMode = 169,
SetNormalMode = 168,
SetUnlockMode = 165,
SetConditionalUnlockMode = 166,
SetBarriedMode = 167,
CheckControllerStatus = 170,
CheckNetworkStatus = 171,
CheckControllerFlags = 172
} ControllerCommandsEnum;

{
ZoneSwitchToDISARM = 143,
ZoneSwitchToARM = 144,
ZoneSetNormalMode = 148,
ZoneSetUnlockMode = 145,
ZoneSetConditionalUnlockMode = 146,
ZoneSetBarriedMode = 147
} ZoneCommandsEnum;

{
SystemSwitchToDISARM = 173,
SystemSwitchToARM = 174,
SystemSetNormalMode = 178,
SystemSetUnlockMode = 175,
SystemSetConditionalUnlockMode = 176,
SystemSetBarriedMode = 177,
SystemCancelAllAlarms = 171,
SystemSetSystemClocks = 150
} SystemCommandsEnum;

{
ConfigurationFile = 1000,
EventsHistoryFile = 1001
} FileTypesEnum;

{
FunctionNotImplementedJet = 200,
PRMasterNotInIdleState = 201,
PRMasterNotInMonitoringState = 202,
PRMasterIsBusy = 203,
NotLoggedIn = 204,
InvalidXmlSyntax = 205,
DatabaseUpdatingError = 206,
UnexpectedError = 207,
InvalidSQLSyntax = 208,
SystemUpdatingError = 209,
InvalidParamValue = 210,
ExecutingCommandError = 211
} ErrorCodesEnum;
```

```
{
StateIdle = 5000,
StateMonitoring = 5001,
StateBusy = 5002
} PRMasterStateEnum;

{
APBZones = 0,
Areas = 1,
ARMZones = 2,
Event_Names = 3,
F7Readers = 4,
FingerprintTemplates = 5,
FunCards = 6,
Groups = 7,
GroupZoneSched = 8,
Holidays = 9,
Input_Functions = 10,
Keyboard_Codes = 11,
Output_Functions = 12,
RdrsARMZone = 13,
RdrsEnterToAPBZone = 14,
RdrsExitFromAPBZone = 15,
Reader_Types = 16,
ReaderProgs = 17,
Readers = 18,
ReadersEnterToArea = 19,
ReadersExitFromArea = 20,
ReaderSwitchers = 21,
Schedules = 22,
Sections = 23,
Time_Zones = 24,
Transponder_Codes = 25,
Users = 26,
UsersOptions = 27,
Zones = 28,
Conditions = 29,
EventsCache = 30
} DBTableNamesEnum;

{
Master = 0,
SwitcherFull = 16,
SwitcherLimited = 32,
Normal = 48
Normal3999 = 64
} UserTypesEnum;

{
GeneralPurposeSchedule = 0,
TandAModeSchedule = 1,
APBResetSchedule = 2,
DoorModeSchedule = 3,
IdentificationModeSchedule = 4
} ScheduleTypesEnum;

{
None = 0,
ReadEvents = 1,
ClearEvents = 2
} EventsBuffersAction;
```

❑ IPRMasterAutomation Definition:

The IPRMasterAutomation is a set of functions which enables modification of the RACS system configuration, sending online commands to access system, reading of events stored in PR Master database. The result of each function is returned as HRESULT type, which specifies if the given function was performed successfully and if it failed then error code is returned. Error codes are defined as **ErrorCodesEnum.**

WinApi32: HResultCode(hr: HRESULT): Integer – returns error code
WinApi32: Succeeded(Status: HRESULT): BOOL –function is successful
WinApi32: Failed(Status: HRESULT): BOOL – function failed

Some programming environments (e.g. Delphi) use individual system which changes COM function's results into error exceptions. In such cases, when calling specific function, application must catch thread in section try-except type **EOleSysError** and using HResultCode get error code. The example of API functions implementation and interpretation of errors can be found in RogerAutomationTest.dpr demo application. All functions must be called synchronically).

interface IPRMasterAutomation: IDispatch
{
HRESULT _stdcall **GetPRMasterVersion**([out, retval] BSTR * version); - Returns version of the PR Master (e.g. 4.2.1.61).

HRESULT _stdcall **Login**([in] BSTR operator, [in] BSTR password, [out, retval] VARIANT_BOOL * result); - Login into a PR Master. Note, that Login is required to call any API functions.

HRESULT _stdcall **Logout**(void); - Logout from PR Master.

HRESULT _stdcall **CreateNewConfig**(void); - Clears PR Master database, same as File/New from PR Master menu.

HRESULT _stdcall **ImportFromXmlFile**([in] BSTR fileName, [in] FileTypesEnum fileType); - Import settings from XML file. Same as File/Import… from PR Master menu.

HRESULT _stdcall **ExportToXmlFile**([in] BSTR fileName, [in] FileTypesEnum fileType); - Export settings to XML file. Same as File/Export… from PR Master menu.

HRESULT _stdcall **GetGroupsXml**([out, retval] BSTR * xml); - Returns definitions of all groups of users.

HRESULT _stdcall **GetGroupXml**([in] long groupID, [out, retval] BSTR * xml); - Returns definition of a group of users with given groupID.

HRESULT _stdcall **AddNewGroup**([in] BSTR xml, [out, retval] long * groupID); - Creates new group of users and returns it groupID.

HRESULT _stdcall **UpdateGroup**([in] long groupID, [in] BSTR xml); - Updates definition of a group of users with given groupID.

HRESULT _stdcall **DeleteGroup**([in] long groupID); - Deletes group of users with given groupID.

HRESULT _stdcall **GetUsersXml**([out, retval] BSTR * xml); - Returns definitions of all users in system.

HRESULT _stdcall **GetUserXml**([in] BSTR userGUID, [out, retval] BSTR * xml); - Returns definition of user with given GUID number.

HRESULT _stdcall **AddNewUser**([in] UserTypesEnum userType, [in] BSTR xml, [out] long * userID, [out, retval] BSTR * userGUID); - Adds new user and returns his ID number and GUID.

HRESULT _stdcall **UpdateUser**([in] BSTR userGUID, [in] BSTR xml); - Updates user with given GUID.

HRESULT _stdcall **AddUserPinCode**([in] BSTR str, [in] BSTR userGUID, [out, retval] VARIANT_BOOL * result); – Adds PIN to user with given GUID

HRESULT _stdcall **DeleteUser**([in] BSTR userGUID); - Deletes user with given GUID, set field Deleted=True.

HRESULT _stdcall **DeleteAllUsers**(void); - Deletes all users and sets all fields Deleted=True.

HRESULT _stdcall **GetSchedulesXml**([out, retval] BSTR * xml); - Returns definitions of all Time Schedules.

HRESULT _stdcall **GetScheduleXml**([in] long scheduleID, [out, retval] BSTR * xml); - Returns definition of a Time Schedule with given ID.

HRESULT _stdcall **AddNewSchedule**([in] ScheduleTypesEnum scheduleType, [in] BSTR xml, [out, retval] long * scheduleID); - Adds new Time Schedule and returns its ID.

HRESULT _stdcall **UpdateSchedule**([in] long scheduleID, [in] BSTR xml); - Updates definition of a Time Schedule with given ID.

HRESULT _stdcall **DeleteSchedule**([in] long scheduleID); - Deletes Time Schedule with given ID.

HRESULT _stdcall **GetZonesXml**([out, retval] BSTR * xml); - Returns definitions of all Zones.

HRESULT _stdcall **GetZoneXml**([in] long zoneID, [out, retval] BSTR * xml); - Returns definition of a Zone with given ID.

HRESULT _stdcall **AddNewZone**([in] BSTR xml, [out, retval] long * zoneID); - Adds new Zone and returns its ID.

HRESULT _stdcall **UpdateZone**([in] long zoneID, [in] BSTR xml); - Updates Zone with given ID.

HRESULT _stdcall **DeleteZone**([in] long zoneID); - Deletes Zone with given ID.

HRESULT _stdcall **GetNetworksXml**([out, retval] BSTR * xml); - Returns definitions of all Networks.

HRESULT _stdcall **GetNetworkXml**([in] long networkID, [out, retval] BSTR * xml); - Returns definition of a Netwotk with given ID.

HRESULT _stdcall **AddNewNetwork**([in] BSTR xml, [out, retval] long * networkID); - Adds new Network and returns its ID.

HRESULT _stdcall **UpdateNetwork**([in] long networkID, [in] BSTR xml); - Updates Network with given ID.

HRESULT _stdcall **DeleteNetwork**([in] long networkID); - Deletes Network with given ID.

HRESULT _stdcall **SendConfigToNetwork**([in] long networkID, [in] EventsBuffersAction bufferAction, [out, retval] BSTR * resultMessage); - Sends settings to all devices in a given Network. The parameter "buffer action" specifies whether the events buffers on controllers must be read or cleared before new settings will be transmitted to the Network.

HRESULT _stdcall **GetControllersXml**([out, retval] BSTR * xml); - Returns definitions of all controllers in a system.

HRESULT _stdcall **GetControllerXml**([in] long controllerID, [out, retval] BSTR * xml); - Returns definition of a controller with given ID.

HRESULT _stdcall **UpdateController**([in] long controllerID, [in] BSTR xml); - Send settings to a controller with given ID.

HRESULT _stdcall **DeleteController**([in] long controllerID); - Deletes controller with given ID.

HRESULT _stdcall **SendConfigToController**([in] long controllerID, [in] EventsBuffersAction bufferAction, [out, retval] BSTR * resultMessage); - Send settings to controller with given ID. Parameter "buffer action" specifies whether event buffer will be read or cleared before new setting will be uploaded into a controller.

HRESULT _stdcall **GetEventsHistoryPeriodXml**([in] DATE fromDate, [in] DATE toDate, [out, retval] BSTR * xml); - Returns list of events (EventsCache) from specified time period.

HRESULT _stdcall **ExecuteControllerCommand**([in] ControllerCommandsEnum command, [in] long controllerID, [out, retval] BSTR * resultMessage); - Sends given command to a controller.

HRESULT _stdcall **ExecuteZoneCommand**([in] ZoneCommandsEnum command, [in] long zoneID, [out, retval] BSTR * resultMessage); - Sends given command to all controllers which belong to given Zone.

HRESULT _stdcall **ExecuteSystemCommand**([in] SystemCommandsEnum command, [out, retval] BSTR * resultMessage); - Send given command to all controllers.

HRESULT _stdcall **ImportFromZIPFile**([in] BSTR fileName, [in] BSTR zipPassword); - Imports system setting from ZIP file, same as File/Import… in PR Master.

HRESULT _stdcall **ExportToZIPFile**([in] BSTR fileName, [in] VARIANT_BOOL includeConfigurationFile, [in] VARIANT_BOOL includeEventsHistoryFile, [in] BSTR zipPassword); - Exports system setting to a ZIP file, same as File/Export… in PR Master.

HRESULT _stdcall **StartMonitoring**(void); - Starts Online Monitoring.

HRESULT _stdcall **StopMonitoring**(void); - Closes Online Monitoring.

HRESULT _stdcall **GetPRMasterState**([out, retval] PRMasterStateEnum * currentState); - Returns status of the application PR Master.

HRESULT _stdcall **GetDBTableXml**([in] DBTableNamesEnum tableName, [in] BSTR filter, [out, retval] BSTR * xml); - Returns given database table filtered according to parameter "filter".

HRESULT _stdcall **ExecuteSelectSQL**([in] BSTR sql, [out, retval] BSTR * xml); - Returns data from SQL function (parameter sql).

HRESULT _stdcall **ExecuteChangeSQL**([in] BSTR sql); - Executes SQL command (DELETE, UPDATE, INSERT).

HRESULT _stdcall **ExecuteChangeSQL2**([in] BSTR sql, [out, retval] SYSINT * affectedRows); - Executes SQL command (DELETE, UPDATE, INSERT). Return the number of affected rows.

HRESULT _stdcall **SendUserConfigToSystem**([in] BSTR userGUID, [in] EventsBuffersAction bufferAction , [out, retval] BSTR * resultMessage); - Updates (sends) settings of a single user to all controllers. Same as Quick User Update command in PR Master.

HRESULT _stdcall **GetEventsHistoryXml**([in] BSTR filter, [out, retval] BSTR * xml); - Returns events history filtered according to parameter "filter".

HRESULT _stdcall **SendConfigToSystem**([in] EventsBuffersAction bufferAction, [out, retval] BSTR * resultMessage); - Updates (sends) settings to all Networks. Parameter "buffer action" specifies whether events buffers will be read or cleared before new settings will be uploaded into controllers.

HRESULT _stdcall **EncodeString**([in] BSTR str, [out, retval] BSTR * result); - Scrambles PIN-code given as a value of field Code in XML in functions AddNewUser and UpdateUser.

HRESULT _stdcall **StartReadingCardCode** ([in] long controllerID); - Runs in the PR Master the process of reading the card code. When reading the card code is returned by calling the event OnReadCardCode, therefore it is possible to read multiple codes cards in one session.

HRESULT _stdcall **StopReadingCardCode**(void); - Exits from the process of reading the card code.

HRESULT _stdcall **ReadEventsBufferNow**([out, retval] BSTR * result)- Reads events buffer. Returns PR Master messages in text form;

HRESULT _stdcall **ToggleControllerFlag**([in] long controllerID, [in] long flagNo, [out, retval] BSTR * result) - Toggles flag flagno on the indicated controller. Returns xml describing the current state of flags. This function is available only for some controllers (PR621-CH);
};

Flags controllered by **ToggleControllerFlag** function		
flagNo	Flag name	Notes
0	LIGHT	
1	TAMPER	
2	AUX1	
3	AUX2	
4	INTRUDER	
5	DURESS	
6	TROUBLE	
7	ENTRY DELAY	
9	POWER SUPPLY	
16	AIR CONDITION	
17	DO NOT DISTURB	
20	CHECK-IN STATUS	
21	MAKE UP ROOM	
22	ASSISTANCE	
24	POWER SUPPLY	
25	ARMED	Can be toggled only if the option 'Enable anti-burglary functions' is activated in the properties of controller by means of PR Master software.
26	GUEST BLOCKING	
32	FLAG1	if activated then FLAG31=1
33	FLAG2	if activated then FLAG31=2
34	FLAG3	if activated then FLAG31=4
35	FLAG4	if activated then FLAG31=8
36	FLAG5	if activated then FLAG31=16
37	FLAG6	if activated then FLAG31=32
38	FLAG7	if activated then FLAG31=64
39	FLAG8	if activated then FLAG31=128

1. For example if FLAG 2 and FLAG3 are activated then the value of FLAG31 equals to 2+4 =6
2. All flags except for FLAG31 when activated equal to >0 and if deactivated then equal to 0

- **IPRMasterAutomationEvents Definition:**

The IPRMasterAutomationEvents interface is used to return events which occur in system when it operates in Online Monitoring mode (Online Monitoring window in PR Master). All functions must be called synchronically.

dispinterface IPRMasterAutomationEvents
{
HRESULT **OnMonitoringEvent**([in] DATE date, [in] DATE time, [in] long networkID, [in] long readerID, [in] long userID, [in] long groupID, [in] long eventCode, [in] long zoneID, [in] long TandAID, [in] BSTR strEvent, [in] BSTR strAccessPoint, [in] BSTR strUserSource, [in] BSTR strGroup, [in] BSTR strNetwork, [in] BSTR strZone, [in] BSTR strTandAMode); - Returns event which occurred during Online Monitoring.

HRESULT **OnChangePRMasterState**([in] PRMasterStateEnum state); - Returns event which occurred after status of the PR Master program is changed.

HRESULT **OnEventsCacheTableNewRecord**([in] long ECID, [in] DATE ECDate, [in] DATE ECTime, [in] long ECCode, [in] long ECUserID, [in] BSTR ECUserGUID, [in] long ECReaderID, [in] long ECDoorType); - This event appears after new record has been added to events table (EventsCache).

HRESULT **OnProblemsWithCommunication**([in] DATE date, [in] DATE ECTime; [in] BSTR Info) - Event occurs aftre the connection with particular network is lost or restored. The Info parameters conveys an information that communication with a given system was lost or restored, i.e: 'Network A: Connection lost' or 'Network A: Connection restored'.
};

3. Hints and tips

- **User enrollment:**

The XML structure for user settings can be easily acquired after enrolling new user by means of PR Master software and then using the function **GetUsersXML**. The example of full XML structure for John Smith with card number 19DEBD01C7 (HEX) is given below. This structure can be used with **AddNewUser** function to save user in PR Master database.

Example 1
<tables><table name="Users"><fields><field name="UserID" type="Integer"/><field name="GUID" type="String"/><field name="FirstName" type="String"/><field name="LastName" type="String"/><field name="Evidence" type="String"/><field name="Status" type="Integer"/><field name="GroupID" type="Integer"/><field name="Official" type="Boolean"/><field name="Active" type="Boolean"/><field name="Photo" type="Graphic"/><field name="PhotoFileName" type="String"/><field name="Custom1" type="String"/><field name="Custom2" type="String"/><field name="Custom3" type="String"/><field name="Custom4" type="String"/><field name="ActivityPeriodOn" type="Boolean"/><field name="ActivitySince" type="DateTime"/><field name="ActivityUntil" type="DateTime"/><field name="UserFlags" type="Integer"/><field name="Deleted" type="Boolean"/><field name="Term0ScheduleID" type="Integer"/><field name="Term1ScheduleID" type="Integer"/><field name="LastLoginDate" type="Date"/><field name="LastLoginTime" type="Time"/><field name="LastLoginReaderID" type="Integer"/><field name="LastLoginEditedByOperator" type="Boolean"/><field name="LastLoginDoorType" type="Integer"/><field name="LastZoneID" type="Integer"/><field name="ActiveSinceLimit" type="Boolean"/><field name="ActiveUntilLimit" type="Boolean"/><field name="HotelGuest" type="Boolean"/><field name="HotelRoomID" type="Integer"/><field name="GuestID" type="Integer"/><field name="RoomService" type="Boolean"/><field name="RecCreatedBy" type="String"/><field name="RecCreatedDate" type="DateTime"/><field name="RecDeletedBy" type="String"/><field name="RecDeletedDate" type="DateTime"/><field name="Protest" type="Boolean"/><field name="WhoGotAccessToThisRec" type="String"/><field name="ExcludeFromADIntegration" type="Boolean"/><field name="NewlyInsertedFromAD" type="Boolean"/></fields><rows><row><f>100</f><f>{C0DF8DE0-9380-489C-B682-69A93EED384A}</f><f>John</f><f>Smith</f><f>null</f><f>48</f><f>0</f><f>False</f><f>True</f><f>null</f><f>null</f><f>null</f><f>null</f><f>null</f><f>null</f><f>False</f><f>2000-01-01T00:00:00</f><f>2099-12-31T23:59:59</f><f>0</f><f>False</f><f>null</f><f>null</f><f>null</f><f>null</f><f>null</f><f>False</f><f>null</f><f>null</f><f>False</f><f>False</f><f>False</f><f>-1</f><f>-1</f><f>False</f><f>ADMIN</f><f>2014-08-06T12:30:55</f><f>null</f><f>null</f><f>null</f><f>null</f><f>null</f><f>null</f></row></rows></table><t

able name="Transponder Codes"><fields><field name="Code" type="String"/><field name="UserGUID" type="String"/><field name="UserID" type="Integer"/><field name="TransCodeID" type="AutoInc"/><field name="UniqueNum" type="Integer"/></fields><rows><row><f>19DEBD01C7</f><f>{C0DF8DE0-9380-489C-B682-69A93EED384A}</f><f>100</f><f>65845</f><f>0</f></row></rows></table><table name="Keyboard Codes"><fields><field name="UserID" type="Integer"/><field name="UserGUID" type="String"/><field name="Code" type="String"/></fields></table><table name="FingerprintTemplates"><fields><field name="UserGUID" type="String"/><field name="FingerprintTemplate" type="Memo"/><field name="FingerIdx" type="SmallInt"/></fields></table></tables>

In practical applications usually it is enough to use **AddNewUser** function with shortened XML structure as in example 2, where new NORMAL type user John Smith with card number 124565833312 (DEC) and assigned to No Group (unlimited access rights) is added to the system. **AddNewUser** returns user GUID number (e.g. {0603A49E-26B9-410C-8EA8-906522E8392F}) which can be then used with other functions. After uploading new user to RACS 4 devices by means of such functions as **SendConfig toNetwork**, **SendConfigtoControlller** or **SendConfigtoSystem** the user is recognized by RACS 4 system.

Example 2
AddNewUser(48,’ <tables><table name="Users"><fields><field name="FirstName" type="String"/><field name="LastName" type="String"/><field name="GroupID" type="Integer"/></fields><rows><row><f>John</f><f>Smith</f><f>0</f></row></rows></table><table name="Transponder Codes"><fields><field name="Code" type="String"/></fields><rows><row><f>1D00B3C660</f></row></rows></table></tables>’, out uid)

❑ User enrollment by means of Quick User Update:

User enrollment by means of Quick User Update requires **AddNewUser** function and then **SendUserConfigToSystem** function with user GUID number returned by **AddNewUser** function. When **SendUserConfigToSystem** is to be used then PR Master software must be in online monitoring mode which can be achieved with **StartMonitoring** function. In the example 3, John Smith assigned to No Group (unlimited access rights) with 1D00B3C660 (HEX) card number is enrolled.

Example 3
AddNewUser(48,’ <tables><table name="Users"><fields><field name="FirstName" type="String"/><field name="LastName" type="String"/><field name="GroupID" type="Integer"/></fields><rows><row><f>Jan</f><f>Kowalski</f><f>0</f></row></rows></table><table name="Transponder Codes"><fields><field name="Code" type="String"/></fields><rows><row><f>1D00B3C660</f></row></rows></table></tables>’, out uid)

SendUserConfigToSystem(‘{0603A49E-26B9-410C-8EA8-906522E8392F}’,0);

❑ Non-latin (e.g. Polish) characters in names (e.g. user name):

XML does not allow to use non-latin characters if their encoding is not specified. Therefore for XML fragments including non-latin characters it is necessary to add following string: <?xml version="1.0" encoding="windows-1250" ?>. Other encoding as for example “UTF-8” can also be applied.

Example 4
<?xml version="1.0" encoding="windows-1250" ?><tables><table name="Users"><fields><field name="UserID" type="Integer"/><field name="GUID" type="String"/><field name="FirstName" type="String"/><field name="LastName" type="String"/><field name="Evidence" type="String"/><field name="Status" type="Integer"/><field name="GroupID" type="Integer"/><field name="Official" type="Boolean"/><field name="Active" type="Boolean"/><field name="Photo" type="Graphic"/><field name="PhotoFileName" type="String"/><field name="Custom1" type="String"/><field name="Custom2" type="String"/><field name="Custom3" type="String"/><field name="Custom4" type="String"/><field name="ActivityPeriodOn" type="Boolean"/><field name="ActivitySince" type="DateTime"/><field name="ActivityUntil" type="DateTime"/><field name="UserFlags" type="Integer"/><field name="Deleted" type="Boolean"/><field name="Term0ScheduleID" type="Integer"/><field name="Term1ScheduleID" type="Integer"/><field name="LastLoginDate" type="Date"/><field name="LastLoginTime" type="Time"/><field name="LastLoginReaderID" type="Integer"/><field name="LastLoginEditedByOperator" type="Boolean"/><field name="LastLoginDoorType" type="Integer"/><field name="LastZoneID" type="Integer"/><field name="ActiveSinceLimit" type="Boolean"/><field name="ActiveUntilLimit" type="Boolean"/><field name="HotelGuest" type="Boolean"/><field name="HotelRoomID" type="Integer"/><field name="GuestID" type="Integer"/><field name="RoomService" type="Boolean"/><field name="RecCreatedBy" type="String"/><field name="RecCreatedDate" type="DateTime"/><field name="RecDeletedBy" type="String"/><field name="RecDeletedDate" type="DateTime"/><field name="Protest" type="Boolean"/><field name="WhoGotAccessToThisRec" type="String"/><field name="ExcludeFromADIntegration" type="Boolean"/><field name="NewlyInsertedFromAD" type="Boolean"/></fields><rows><row><f>102</f><f>{5A867578-D2E9-4B3C-

8DAE-
6D1B39E9F615}</f><f>Józef</f><f>Pączyński</f><f>null</f><f>48</f><f>0</f><f>False</f><f>True</f><f>n
ull</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f><f>False</f><f>2000-01-
01T00:00:00</f><f>2099-12-
31T23:59:59</f><f>0</f><f>False</f><f>null</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f><f>False</f><f
>null</f&gt;<f>null</f><f>False</f><f>False</f><f>False</f><f>-1</f><f>-
1</f><f>False</f><f>ADMIN</f><f>2014-07-08T09:07:44</f><f>null</f&
gt;<f>null</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f&gt;<f>null</f></row></rows></table><table name="Transponder
Codes"><fields><field name="Code" type="String"/><field name="UserGUID" type="String"/><field name="UserID"
type="Integer"/><field name="TransCodeID" type="AutoInc"/><field name="UniqueNum"
type="Integer"/></fields></table><table name="Keyboard Codes"><fields><field name="UserID"
type="Integer"/><field name="UserGUID" type="String"/><field name="Code"
type="String"/></fields><rows><row><f>102</f><f>{5A867578-D2E9-4B3C-8DAE-
6D1B39E9F615}</f><f>Y98LyThZv/XvwdJ+HsZ4pA==</f></row></rows></table><table
name="FingerprintTemplates"><fields><field name="UserGUID" type="String"/><field name="FingerprintTemplate"
type="Memo"/><field name="FingerIdx" type="SmallInt"/></fields></table></tables>