



**Parsec Electronic Access Control System  
integration service description  
(PNSoft and PNOffice software)**

**2021/12/22 version**

## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>7</b>
General.....	7
How the service works .....	7
Client application functionality .....	7
Some recommendations for developers.....	8
Development Kit Contents.....	8
Integration service functions supported by PNSoft and PNOffice .....	8
<b>2. USED OBJECTS.....</b>	<b>12</b>
BaseResult .....	12
GuidIdResult .....	12
SessionResult .....	12
StringResult .....	12
ObjectResult.....	13
EventsHistoryResult.....	13
Session.....	13
BaseObject.....	13
BaseOrgUnit .....	13
OrgUnit.....	14
BasePerson .....	14
Person .....	14
PersonWithPhoto .....	14
PersonExtraFieldTemplate .....	15
PersonScheduleFix .....	15
ExtraFieldValue .....	15
VisitorRequest .....	15
Schedule.....	16
AccessSchedule .....	16
WorktimeSchedule .....	17
ScheduleDay .....	17
ScheduleFix.....	17
TimeInterval.....	18
WorktimeInterval .....	18
Holiday .....	18
BaseIdentifier .....	18
Identifier.....	18
IdentifierTemp .....	19
StockIdentifier .....	19
IdentifierExData.....	19
PassageRole .....	20

BaseTerritory .....	20
Territory .....	20
TerritoryWithComponent .....	20
AccessGroup.....	21
SubAccessGroup .....	21
Event.....	22
EventsHistory .....	22
EventObject .....	22
Domain.....	22
EventHistoryQueryParams .....	22
HardwareState.....	24
TransactionClass .....	24
TransactionType.....	24
<b>3. FUNCTIONS .....</b>	<b>25</b>
VERSION.....	25
GetVersion function .....	25
SESSION.....	25
GetDomains function .....	25
OpenSession function .....	25
OpenSessionWithInLocale function .....	25
ContinueSession function .....	26
CloseSession function .....	26
CheckRole function.....	26
GetObjectName function .....	27
EQUIPMENT.....	27
SendHardwareCommand function.....	27
SendVerificationCommand function .....	28
GetHardwareState function .....	28
SUBDIVISIONS .....	30
GetRootOrgUnit function .....	30
GetOrgUnitsHierarhy function .....	30
GetOrgUnitsHierarhyWithPersons function .....	30
GetOrgUnitsHierarhyWithVisitors function.....	30
GetOrgUnitsHierarhyWithVehicle function .....	30
GetOrgUnitSubItems function .....	31
GetOrgUnitSubItemsHierarhyWithPersons function .....	31
GetOrgUnitSubItemsHierarhyWithVisitors function.....	31
GetOrgUnitSubItemsHierarhyWithVehicle function .....	31
GetOrgUnit function.....	32

EDITING A DEPARTMENT .....	32
CreateOrgUnit function .....	32
OpenOrgUnitEditingSession function .....	32
CloseOrgUnitEditingSession function .....	32
SaveOrgUnit function.....	33
DeleteOrgUnit function.....	33
PERSONNEL.....	33
GetPersonExtraFieldTemplates function.....	33
GetVisitorExtraFieldTemplates Function.....	33
GetVehicleExtraFieldTemplates function .....	34
FindPeople function .....	34
FindVisitors function .....	34
FindVehicle function .....	34
FindPersonByIdentifier function.....	35
PersonSearch function .....	35
GetPerson function .....	36
GetMultiplePersons function.....	36
GetPersonsChangedAfter function .....	37
GetPersonExtraFieldValue function .....	37
GetPersonExtraFieldValues function .....	37
GetPersonExtraFieldValueString function .....	37
ValidateExtraFieldValue function .....	38
GetPersonScheduleFixes function .....	38
AddPersonScheduleFix function.....	38
SavePersonScheduleFix function .....	39
DeletePersonScheduleFix function .....	39
GetPersonWorktimeSchedule function.....	39
SetPersonWorktimeSchedule function .....	39
EDITING THE ACCESS SUBJECT .....	40
CreatePerson function.....	40
CreateVisitor function .....	40
CreateVehicle function .....	40
OpenPersonEditingSession function .....	40
ClosePersonEditingSession function .....	41
SavePerson function .....	41
SetPersonPhoto function .....	41
SetPersonOrgUnit function .....	41
SetPersonExtraFieldValue function.....	42
SetPersonExtraFieldValues function .....	42

DeletePerson function .....	43
BlockPerson function .....	43
UnblockPerson function.....	43
TOPOLOGY .....	44
GetRootTerritory function.....	44
GetTerritoriesHierarhy function .....	44
GetTerritorySubItems function.....	44
GetTerritory function .....	44
IDENTIFIERS AND ACCESS.....	44
GetPersonIdentifiers function.....	44
DeleteIdentifier function.....	45
AddPersonIdentifier function.....	45
ChangePersonIdentifier function.....	46
SetIdentifierPrivileges function.....	47
GetIdentifierExtraData function.....	47
SetIdentifierExtraData function .....	48
GetPassageRoles function.....	48
CreatePassageRole function.....	48
SavePassageRole function .....	48
DeletePassgeRole function.....	49
DeletePassgeRole function.....	49
GetUnique4bCardCode function.....	49
GetCardCodeFromUID function .....	49
GenerateParsecQRCode function .....	50
SCHEDULES AND ACCESS GROUPS .....	50
GetAccessSchedules function .....	50
GetWorktimeSchedules function.....	50
GetScheduleIntervals function .....	50
CreateAccessSchedule function .....	51
CreateWorktimeSchedule function .....	51
GetSchedule function.....	52
SaveSchedule function.....	52
DeleteSchedule function.....	52
GetScheduleDetails function .....	52
SetScheduleDays function .....	53
SetScheduleFix function .....	53
DeleteScheduleDays function.....	53
GetHolidays function.....	53
SetHolidays function .....	54

DeleteHolidays function .....	54
GetAccessGroups function .....	54
CreateTempAccessGroup function .....	54
CreateAccessGroup function .....	55
DeleteAccessGroup function .....	55
AddSubAccessGroup function.....	55
DeleteSubAccessGroup function .....	56
GetSubAccessGroups function.....	56
GetInheritedAccessGroups function .....	56
SetInheritedAccessGroups function.....	56
WORK WITH THE PASS OFFICE REQUESTS.....	57
GetAcceptedVisitorRequests function .....	57
FindVisitorRequest function .....	57
ActivateVisitorRequest function .....	57
CreateVisitorRequest function .....	57
GetVisitorRequest function .....	58
SaveVisitorRequest function .....	58
DeleteIssuedVisitorRequest Function .....	58
GetIssuedVisitorRequests function.....	58
GetVisitorRequests function .....	59
CloseAllActiveVisitorRequests function .....	59
CloseVisitorRequest function.....	59
GetPersonVisitorRequests function .....	60
SYSTEM EVENTS .....	60
GetEvents function .....	60
OpenEventHistorySession function .....	60
CloseEventHistorySession function .....	61
GetEventHistoryResultCount function.....	61
GetEventHistoryResult function .....	61
GetHardwareEvents function.....	63
GetHardwareEventsResolved function .....	66
GetTransactionClasses function .....	68
GetTransactionTypes function .....	68
<b>4. HISTORY OF CHANGES.....</b>	<b>69</b>
<b>5. ALPHABETIC INDEX .....</b>	<b>70</b>

# 1. INTRODUCTION

## General

This document describes the functionality of the integration service (hereinafter simply "service") of the professional access control system (ACS) ParsecNET.

The service allows third-party software packages to access data and events of the ParsecNET system to implement specific functionality that is absent in the system, for example, such external applications as personnel management, pass bureaus, and many others that need access to information about ParsecNET personnel (users) and authorized access events.

The service is installed automatically on the server of the ParsecNET system and is included in the basic version of the software.

## How the service works

The service runs as a Windows service on the computer, which is the server of the ParsecNET system. It is implemented as an XML / SOAP WEB service, which allows you to access it from any computer of the network not just from the ACS server. Accordingly, an external application can be created in any development environment that supports WEB services technology and, in particular, SOAP. The service is maintained by MDO.Parsec.ParsecIntegrationServiceHost.exe application.

The service location address on the ACS server is specified in the configuration file MDO.Parsec.ParsecIntegrationServiceHost.exe.config (by default - <http://localhost:10101/IntegrationService/IntegrationService.asmx>).

WSDL document can be accessed with URL:  
<http://localhost:10101/IntegrationService/IntegrationService.asmx?wsdl>

## Client application functionality

The client application is equivalent to the operator of one specific organization of the system. To gain access to the service in the system, you need to create an operator with the necessary rights and scopes. The operator's rights determine the result of performing the functions of the integration service in the Parsec system, with which the session is open.

The service is operated at the session level: the client application opens a session after login and uses the service functions within this session. The service will automatically close the session if the client has not used any calls to the service functions for more than five minutes.

The client application using the service can access the structure of the personnel of the ParsecNET system, including the entire hierarchy of departments within a particular organization.

The service provides enough functions to create, edit or delete a user of the ParsecNET system. At the same time, as when working with system applications, any change made without the participation of the client is replicated to all objects of the system, including controllers.

The service allows you to create both temporary and regular (permanent) access groups. The service can provide a complete list of access groups in order to assign to the personnel the necessary access rights to the territory of the object in the client application.

In addition to working with personnel, the service provides access to system events.

Its unique key accompanies each object received from the system, which is necessary to identify this object. Windows GUIDs are used as keys.

## Some recommendations for developers

It is required that the ASP.NET 4.x component must be enabled in Windows OS for the integration service to work.

Data from the system can be obtained using various functions of the service. The developer chooses the data collection strategy depending on the tasks facing him.

However, it should be borne in mind that some functions can return very large amounts of data in large systems (depending on the scale of the system itself). Accordingly, the operation can take a long time and consume an unreasonable amount of resources.

Therefore, it is preferable to first obtain the general hierarchy of an entity (for example, a tree of departments), and then obtain data about personnel not for the entire system at once, but for the department with which the client application is actually working.

## Development Kit Contents

The development kit includes the following components:

- This manual
- An example of a client application in C # with sources and a working compiled example.
- An example of an Object Pascal client application with source code and a working compiled example (Delphi development environment, Borland Developer Studio 2007. Earlier Delphi versions do not fully support the required technologies).

## Integration service functions supported by PNSoft and PNOffice

The integration service included in the PNOffice (ParsecNET Office) software package supports a subset of the functions described in this document due to the lack of some functionality, for example, a pass office.

The table below provides information on the compatibility of functions in PNSoft and PNOffice software products.

Also the functions are grouped into sections in accordance with the functionality that they provide to the developer.

PNSoft	PNOffice
<b>Working with a session</b>	
<a href="#">CloseSession</a>	
<a href="#">ContinueSession</a>	
<a href="#">OpenSession</a>	
<a href="#">OpenSessionWithInLocale</a>	
<b>Working with access groups</b>	
<a href="#">AddSubAccessGroup</a>	not supported
<a href="#">CreateAccessGroup</a>	not supported
<a href="#">CreateTempAccessGroup</a>	
<a href="#">DeleteAccessGroup</a>	not supported
<a href="#">DeleteSubAccessGroup</a>	not supported
<a href="#">GetAccessGroups</a>	



<b>PNSoft</b>	<b>PNOffice</b>
<b>Access group inheritance</b>	
	not supported
	not supported
	not supported
<b>Group pass roles</b>	
<a href="#">CreatePassageRole</a>	not supported
<a href="#">SavePassageRole</a>	not supported
<a href="#">DeletePassageRole</a>	not supported
<a href="#">GetPassageRoles</a>	not supported
<b>Retrieving system events from the database archive</b>	
<a href="#">OpenEventHistorySession</a>	
<a href="#">GetEventHistoryResult</a>	
<a href="#">GetEventHistoryResultCount</a>	
<a href="#">CloseEventHistorySession</a>	
<a href="#">GetEvents</a> (obsolete one, not recommended for use)	
<b>Receiving operational events and statuses of access controllers</b>	
<a href="#">GetHardwareEvents</a>	not supported
<a href="#">GetHardwareEventsResolved</a>	not supported
<a href="#">GetHardwareState</a>	not supported
<b>Getting classes and types of events generated by the system</b>	
<a href="#">GetTransactionClasses</a>	not supported
<a href="#">GetTransactionTypes</a>	not supported
<b>Commands for direct control of access controllers</b>	
<a href="#">SendHardwareCommand</a>	not supported
<a href="#">SendVerificationCommand</a>	not supported
<b>Working with departments, employees, vehicles and identifiers</b>	
<b><i>Departments</i></b>	
<a href="#">OpenOrgUnitEditingSession</a>	
<a href="#">CloseOrgUnitEditingSession</a>	
<a href="#">CreateOrgUnit</a>	
<a href="#">SaveOrgUnit</a>	
<a href="#">GetOrgUnit</a>	
<a href="#">DeleteOrgUnit</a>	
<a href="#">GetOrgUnitSubItems</a>	
<a href="#">GetOrgUnitsHierarhy</a>	
<a href="#">GetRootOrgUnit</a>	
<b><i>Employees</i></b>	
<a href="#">OpenPersonEditingSession</a>	
<a href="#">ClosePersonEditingSession</a>	
<a href="#">AddPersonIdentifier</a>	
<a href="#">ChangePersonIdentifier</a>	
<a href="#">GetPersonIdentifiers</a>	
<a href="#">GetIdentifierExtraData</a>	not supported
<a href="#">SetIdentifierExtraData</a>	not supported
<a href="#">CreatePerson</a>	
<a href="#">SavePerson</a>	
<a href="#">DeletePerson</a>	
<a href="#">DeleteIdentifier</a>	
<a href="#">GetOrgUnitsHierarhyWithPersons</a>	

<b>PNSoft</b>	<b>PNOffice</b>
<a href="#">GetOrgUnitSubItemsHierarhyWithPersons</a>	
<a href="#">GetPerson</a>	
<a href="#">GetMultiplePersons</a>	not supported
<a href="#">SetPersonOrgUnit</a>	
<a href="#">GetPersonWorktimeSchedule</a>	
<a href="#">SetPersonWorktimeSchedule</a>	
<a href="#">SetPersonPhoto</a>	
<a href="#">SetIdentifierPrivileges</a>	not supported
<b><i>Search for access subjects</i></b>	
<a href="#">FindPeople</a>	
<a href="#">FindPersonByIdentifier</a>	not supported
<a href="#">FindVehicle</a>	not supported
<a href="#">PersonSearch</a>	not supported
<a href="#">GetPersonsChangedAfter</a>	
<b><i>Vehicles</i></b>	
<a href="#">CreateVehicle</a>	not supported
<a href="#">GetOrgUnitsHierarhyWithVehicle</a>	not supported
<a href="#">GetOrgUnitSubItemsHierarhyWithVehicle</a>	not supported
<b><i>Generating QR codes and working with ID</i></b>	
<a href="#">GenerateParsecQRCode</a>	not supported
<a href="#">GetCardCodeFromUID</a>	not supported
<a href="#">GetUnique4bCardCode</a>	not supported
<b><i>Extra fields of the access subject</i></b>	
<a href="#">GetPersonExtraFieldTemplates</a>	
<a href="#">GetPersonExtraFieldValue</a>	
<a href="#">GetPersonExtraFieldValueString</a>	
<a href="#">GetPersonExtraFieldValues</a>	
<a href="#">SetPersonExtraFieldValue</a>	
<a href="#">SetPersonExtraFieldValues</a>	
<a href="#">ValidateExtraFieldValue</a>	
<a href="#">GetVehicleExtraFieldTemplates</a>	not supported
<b><i>Access blocking/unblocking</i></b>	
<a href="#">BlockPerson</a>	
<a href="#">UnblockPerson</a>	
<b><i>Working with topology</i></b>	
<a href="#">GetRootTerritory</a>	
<a href="#">GetTerritoriesHierarhy</a>	
<a href="#">GetTerritory</a>	
<a href="#">GetTerritorySubItems</a>	
<b><i>Schedules</i></b>	
<a href="#">CreateAccessSchedule</a>	not supported
<a href="#">CreateWorktimeSchedule</a>	not supported
<a href="#">DeleteSchedule</a>	not supported
<a href="#">DeleteScheduleDays</a>	not supported
<a href="#">GetAccessSchedules</a>	
<a href="#">GetWorktimeSchedules</a>	
<a href="#">GetSchedule</a>	not supported

<b>PNSoft</b>	<b>PNOffice</b>
<a href="#">GetScheduleDetails</a>	not supported
<a href="#">GetScheduleIntervals</a>	
<a href="#">SaveSchedule</a>	not supported
<a href="#">SetScheduleDays</a>	not supported
<a href="#">SetScheduleFix</a>	not supported
<b>Holidays</b>	
<a href="#">GetHolidays</a>	not supported
<a href="#">SetHolidays</a>	not supported
<a href="#">DeleteHolidays</a>	not supported
<b>Working time corrections</b>	
<a href="#">AddPersonScheduleFix</a>	not supported
<a href="#">DeletePersonScheduleFix</a>	not supported
<a href="#">GetPersonScheduleFixes</a>	not supported
<a href="#">SavePersonScheduleFix</a>	not supported
<b>Pass office &amp; visitors</b>	
<a href="#">ActivateVisitorRequest</a>	not supported
<a href="#">CloseAllActiveVisitorRequests</a>	not supported
<a href="#">CloseVisitorRequest</a>	not supported
<a href="#">CreateVisitor</a>	not supported
<a href="#">CreateVisitorRequest</a>	not supported
<a href="#">DeleteIssuedVisitorRequest</a>	not supported
<a href="#">FindVisitors</a>	not supported
<a href="#">FindVisitorRequest</a>	not supported
<a href="#">GetAcceptedVisitorRequests</a>	not supported
<a href="#">GetIssuedVisitorRequests</a>	not supported
<a href="#">GetPersonVisitorRequests</a>	not supported
<a href="#">GetVisitorRequest</a>	not supported
<a href="#">SaveVisitorRequest</a>	not supported
<a href="#">GetOrgUnitsHierarhyWithVisitors</a>	not supported
<a href="#">GetOrgUnitSubItemsHierarhyWithVisitors</a>	not supported
	not supported
<b>Security</b>	
<a href="#">CheckRole</a>	not supported
<a href="#">GetDomains</a>	
<b>Getting the SDK version (software version)</b>	
<a href="#">GetVersion</a>	
<b>Other</b>	
<a href="#">GetObjectName</a>	not supported

## 2. USED OBJECTS

### BaseResult

The base class is used as the result of the operation.

<code>int</code> Result	The result of the operation.
<code>string</code> ErrorMessage	A description of the error that occurred during the operation.

The result of the operation can be as follows:

- 0 - the operation was successful;
- -1 - the operation was performed with an error.

Values > 0 are planned to be used for specific error codes.

### GuidResult

Base class: [BaseResult](#).

The class is used as a result in functions that return Guid.

<code>int</code> Result	The result of the operation. 0 - the operation was successful, -1 - the operation was completed with an error. Values > 0 are planned to be used for specific error codes.
<code>string</code> ErrorMessage	A description of the error that occurred during the operation.
<code>Guid</code> Value	Resulting ID.

### SessionResult

Base class: [BaseResult](#).

The class is used as the result of the OpenSession function.

<code>int</code> Result	The result of the operation. 0 - the operation was successful, -1 - the operation was completed with an error. Values > 0 are planned to be used for specific error codes.
<code>string</code> ErrorMessage	A description of the error that occurred during the operation.
<code>SessionValue</code>	The result of the operation.

### StringResult

Base class: [BaseResult](#).

The class is used as a result in functions that return String.

<code>int</code> Result	The result of the operation. 0 - the operation was successful, -1 - the operation was completed with an error. Values > 0 are planned to be used for specific error codes.
-------------------------	--

<code>String</code> ErrorMessage	A description of the error that occurred during the operation.
<code>String</code> Value	The result of the operation.

## ObjectResult

Base class: [BaseResult](#).

The class is used as a result in functions that return Object.

<code>int</code> Result	The result of the operation. 0 - the operation was successful, -1 - the operation was completed with an error. Values > 0 are planned to be used for specific error codes.
<code>string</code> ErrorMessage	A description of the error that occurred during the operation.
<code>Object</code> Value	The result of the operation.

## EventsHistoryResult

Base class: [BaseResult](#).

The class is used as the result of a function GetEvents.

<code>int</code> Result	The result of the operation. 0 - the operation was successful, -1 - the operation was completed with an error. Values > 0 are planned to be used for specific error codes.
<code>String</code> ErrorMessage	A description of the error that occurred during the operation.
<code>EventsHistory</code> Value	The result of the operation.

## Session

Class containing information for the session.

<code>Guid</code> SessionID	A unique session key used for further operations.
<code>Guid</code> RootOrgUnitID	ID of the root element of the personnel tree.
<code>Guid</code> RootTerritoryID	ID of the root element of the territories tree.

## BaseObject

The class used as the base for the main service classes. It has no own members.

## BaseOrgUnit

Base class: [BaseObject](#).

The class used to describe the division.

<code>Guid</code> ID	The unique division key.
<code>String</code> NAME	Division name.

<code>string</code> DESC	Description of the division.
--------------------------	------------------------------

## OrgUnit

Base class: [BaseOrgUnit](#).

The class used to describe the unit.

<code>Guid</code> ID	The unique division key.
<code>String</code> NAME	Division name.
<code>String</code> DESC	Description of the division.
<code>Guid</code> PARENT_ID	The unique key of the parent division.

## BasePerson

Base class: [BaseObject](#).

The class used to describe the employee.

<code>Guid</code> ID	Employee's unique key.
<code>string</code> LAST_NAME	Surname.
<code>string</code> FIRST_NAME	Name.
<code>string</code> MIDDLE_NAME	Middle name.
<code>string</code> TAB_NUM	Personnel Number.

## Person

Base class: [BasePerson](#).

The class used to describe the employee.

<code>Guid</code> ID	Employee's unique key.
<code>string</code> LAST_NAME	Surname.
<code>string</code> FIRST_NAME	Name.
<code>string</code> MIDDLE_NAME	Middle name.
<code>string</code> TAB_NUM	Personnel Number.
<code>Guid</code> ORG_ID	The unique department key for the employee.

## PersonWithPhoto

Base class: [Person](#).

The class used to describe the employee.

<code>Guid</code> ID	Employee's unique key.
<code>string</code> LAST_NAME	Surname.
<code>string</code> FIRST_NAME	Name.
<code>string</code> MIDDLE_NAME	Middle name.

<code>string</code> TAB_NUM	Personnel Number.
<code>Guid</code> ORG_ID	The unique department key for the employee.
<code>byte[]</code> PHOTO	Photo of the employee.

## PersonExtraFieldTemplate

The class used to define the additional personnel data field.

<code>Guid</code> ID	The unique key of the template.
<code>XmlTypeCode</code> TYPE	The type of data presented.
<code>string</code> NAME	Template name.

## PersonScheduleFix

The class used to determine the employee's working time correction.

<code>Guid</code> FIX_ID	The unique correction key.
<code>Guid</code> PERSON_ID	Employee's unique key.
<code>int</code> TYPE_ID	Correction type; it can take values: 4 - Sick leave; 5 - Business trip; 6 - Vacation; 7 - Hired; 8 - Fired; 9 - Vacation without pay; 10 - Full time; 11 - Presence (only in this correction a non-zero time can be transmitted, the rest of the corrections must be with the time 00:00).
<code>DateTime</code> START	Date and time of the start of the correction interval.
<code>DateTime</code> END	Date and time of completion of the correction interval.
<code>string</code> COMMENT	Commentary on the amendment of working hours.

## ExtraFieldValue

Base class: [BaseObject...](#)

The class used to describe the value of the employee extra field.

<code>Guid</code> TEMPLATE_ID	The unique key of the template.
<code>Object</code> VALUE	Field value.

## VisitorRequest

The class used to describe the requests of the Pass Office

<code>Guid</code> ID	The unique request key.
----------------------	-------------------------

<code>int</code> NUMBER	The unique request number.
<code>DateTime</code> DATE	Date of creation of the request.
<code>Guid</code> ORGUNIT_ID	The unique key of the department for which the request was created.
<code>Guid</code> PERSON_ID	Visitor unique key.
<code>string</code> PERSON_INFO	Additional information about the visitor.
<code>string</code> PURPOSE	Purpose of the visit.
<code>int</code> STATUS	Request status.
<code>DateTime</code> ADMIT_START	The start date of the permitted visit.
<code>DateTime</code> ADMIT_END	The end date of the permitted visit.

The status of the request can take values:

- 0 – Issued (waiting for approval);
- 1 – Accepted (you can issue an identifier to a visitor);
- 2 – Declined (you can only close the request);
- 3 – Active (the identifier is in the hands of the visitor);
- 4 – Complete (request is closed).

## Schedule

Base class: [BaseObject](#).

Class used to represent the system schedule.

<code>Guid</code> ID	A schedule unique key.
<code>string</code> NAME	Name.

## AccessSchedule

Base class: [Schedule](#).

The class used to represent the access schedule.

<code>Guid</code> ID	A schedule unique key.
<code>string</code> NAME	Name.
<code>string</code> DESC	Description of the schedule.
<code>bool</code> IS_WEEK	Indicates whether the schedule is weekly.
<code>int</code> HOLIDAYS_ACTION	Method of applying holidays to the schedule: 0 - Apply with replacement 1 - Apply with insert 2 - Do not apply



## WorktimeSchedule

Base class: [AccessSchedule](#).

The class used to represent the work schedule.

<code>Guid</code> ID	A schedule unique key.
<code>string</code> NAME	Name.
<code>string</code> DESC	Description of the schedule.
<code>bool</code> IS_WEEK	Indicates whether the schedule is weekly.
<code>int</code> HOLIDAYS_ACTION	Method of applying holidays to the schedule: 0 - Apply with replacement; 1 - Apply with insert; 2 - Do not apply.
<code>int</code> HOURS_PER_WEEK	Work rate per week (hours).
<code>int</code> HOURS_PER_DAY	Work rate per day (hours).

## ScheduleDay

Base class: [BaseObject](#).

Class used to describe the pattern of a day in a schedule cycle.

<code>DateTime</code> DATE	The start date of the schedule cycle.
<code>int</code> INDEX	Day number in the cycle of the schedule. The index of the first day in the cycle has a value of "1".
<code>TimeInterval[]</code> INTERVALS	Array of time intervals in the day pattern.

## ScheduleFix

Base class: [ScheduleDay](#).

The class used to describe the correction day.

<code>DateTime</code> DATE	Date of correction.
<code>int</code> INDEX	Not used.
<code>TimeInterval[]</code> INTERVALS	An array of time intervals in a correction day.
<code>int</code> ACTION	Method of applying a correction day to the schedule: 0 - Apply with replace; 1 - Apply with insert. The value is always "0" for weekly schedules.

## TimeInterval

Base class: [BaseObject](#).

Class used to represent a time interval.

<code>DateTime</code> START	The beginning of the interval.
<code>DateTime</code> END	The end of the interval.

## WorkTimeInterval

Base class: [TimeInterval](#).

Class used to represent a time interval of a work schedule.

<code>DateTime</code> START	The beginning of the interval.
<code>DateTime</code> END	The end of the interval.
<code>int</code> TYPE	Time interval type: 0 - Working hours (access allowed); 1 - Night shift; 2 - Break (lunch); 3 - Mandatory working hours.

## Holiday

Base class: [BaseObject](#).

Class used to describe holidays.

<code>string</code> NAME	The name of the holiday.
<code>byte</code> MONTH	Month (holiday dates).
<code>byte</code> DAY	Date (holiday dates).

## BaseIdentifier

Base class: [BaseObject](#).

The class used to describe the identifier.

<code>string</code> CODE	Identifier code (a string containing a number in hexadecimal format, exactly 8 characters long).
<code>Guid</code> PERSON_ID	Employee's unique key.
<code>bool</code> IS_PRIMARY	Indicates whether the identifier is primary.

## Identifier

Base class: [BaseIdentifier](#).

The class used to describe the identifier.

<code>string</code> CODE	Identifier code (a string containing a number in hexadecimal format, exactly 8 characters long).
<code>Guid</code> PERSON_ID	Employee's unique key.

<code>bool</code> IS_PRIMARY	Indicates whether the identifier is primary.
<code>Guid</code> ACCGROUP_ID	The unique key of the ID access group.
<code>Long</code> PRIVILEGE_MASK	Privilege mask.
<code>int</code> IDENTIFYTYPE	Identifier type. (Values: 0 - "Parsec" access subsystem; 1 - License plate).
<code>string</code> NAME	Identifier name. Used for informational purposes.

## IdentifierTemp

Base class: [Identifier](#).

The class used to describe the temporary identifier.

<code>string</code> CODE	Identifier code (a string containing a number in hexadecimal format, exactly 8 characters long).
<code>Guid</code> PERSON_ID	Employee's unique key.
<code>bool</code> IS_PRIMARY	Indicates whether the identifier is primary.
<code>Guid</code> ACCGROUP_ID	The unique key of the ID access group.
<code>DateTime</code> VALID_FROM	The start date of the identifier.
<code>DateTime</code> VALID_TO	The expiration date of the identifier.

## StockIdentifier

Base class: [IdentifierTemp](#).

The class used to describe a pooled identifier.

<code>string</code> CODE	Identifier code (a string containing a number in hexadecimal format, exactly 8 characters long).
<code>Guid</code> PERSON_ID	Employee's unique key.
<code>bool</code> IS_PRIMARY	Indicates whether the identifier is primary.
<code>Guid</code> ACCGROUP_ID	The unique key of the ID access group.
<code>DateTime</code> VALID_FROM	The start date of the identifier.
<code>DateTime</code> VALID_TO	The expiration date of the identifier.

## IdentifierExData

Base class: [BaseObject](#).

A class used to describe additional properties of an identifier.

<code>Guid</code> PASSAGE_ROLE_ID	The unique key of the group passage role.
<code>int</code> ENTRY_LIMIT	Maximum allowed number of passes (value "-1" - unlimited number of passes; "0" - access denied; "127" - maximum possible restriction).
<code>Guid</code> OWNED_COMPONENT_ID	The unique key of the "Door" component owned by the identifier. (Used in the "Cabinet owner" functionality)

## PassageRole

Base class: [BaseObject](#).

The class used to describe the role of the group pass.

<code>Guid</code> ID	The role unique key.
<code>string</code> NAME	Role name.
<code>string</code> DESCRIPTION	Role description.

## BaseTerritory

The base class used to describe the territory.

<code>Guid</code> ID	The territory unique key.
<code>byte</code> TYPE	Territory object type.
<code>string</code> NAME	Territory name.
<code>string</code> DESC	Description of the territory.

The territory object type can take values:

- 0 - folder;
- 1 - door;
- 2 - part (not yet used);
- 3 - other components.

## Territory

Base class: [BaseTerritory](#).

The class used to describe the territory.

<code>Guid</code> ID	The territory unique key.
<code>byte</code> TYPE	Territory object type.
<code>string</code> NAME	Territory name.
<code>string</code> DESC	Description of the territory.
<code>Guid</code> PARENT_ID	The parent territory unique key.

## TerritoryWithComponent

Base class: [Territory](#).

A class used to describe a territory with information about its associated component.

<code>Guid</code> ID	The territory unique key.
<code>byte</code> TYPE	Territory object type.
<code>string</code> NAME	Territory name.
<code>string</code> DESC	Description of the territory.

<code>Guid</code> PARENT_ID	The parent territory unique key.
<code>Guid</code> COMPONENT_ID	The component unique key.
<code>long</code> FEATURE_MASK	Component properties bitmask. The meaning of the bits are described in the table below.

Component properties mask:

Bit number	Meaning
0	Door
1	2 readers
2	APB
3	supports complex schedules
4	used as a desktop reader
5	guard area
6	video stream source
7	turnstile
8	- Reserved -
9	does not support schedules
10	software controller
11	- Reserved -
12	not limited by license
13	elevator controller
14	- Reserved -
62	disabled by the operator
63	not serviced due to license restrictions

## AccessGroup

Base class: [BaseObject](#).

The class used to describe the access group.

<code>Guid</code> ID	The access group unique key.
<code>string</code> NAME	Access group name.
<code>int</code> IDENTIFYTYPE	Access group type. (Values: 0 - "Parsec" access subsystem; 1 - License plate).

## SubAccessGroup

Base class: [BaseObject](#).

The class used to describe the access subgroup.

<code>Guid</code> SubGroupID	The group of component unique key.
<code>Guid</code> ScheduleID	A schedule unique key.
<code>Guid[]</code> Territories	An array of territory keys.

## Event

Base class: [BaseObject](#).

A class used to describe a system event.

<code>DateTime</code> EventDate	Event date.
<code>int</code> EventType	Event type (0-entry; 1-exit).
<code>int</code> EventPersonIndex	The employee's index in EventsHistory.
<code>string</code> CODE	ID code.
<code>int</code> EventTerritoryIndex	The territory index in EventsHistory.

The event type can take values:

0 - entry;

1 - exit.

## EventsHistory

Class used to describe system events.

<code>Event[]</code> Events	Array of events.
<code>Guid[]</code> Persons	An array of employee unique keys.
<code>string[]</code> PersonFullNames	Array full name employees.
<code>Guid[]</code> Territories	An array of territory unique keys.
<code>string[]</code> TerritoryNames	Array of territory names.

## EventObject

Base class: [BaseObject](#).

Class used to describe event data.

<code>Object[]</code> Values	An array of field values describing the event.
------------------------------	--

## Domain

Base class: [BaseObject](#).

The class used to describe the organization.

<code>string</code> NAME	Name of company.
<code>string</code> DESCRIPTION	Description of the organization.
<code>bool</code> VISITOR_CONTROL	The Pass Office attribute (not used since version 3.2).
<code>bool</code> IS_SYSTEM	SYSTEM organization attribute.

## EventHistoryQueryParams

A class used to describe the criteria of the generated event report. Null can be passed as any parameter. In this case, this criterion will not be used when selecting events in the report.

Date and time parameters always transmitted in UTC.

<code>Guid[] IDs</code>	An array of event keys to be included in the report. If given, all other parameters are ignored.
<code>Guid[] ParentEventID</code>	Array of keys for "primary" events. All events related to primary events will be selected in the report. If given, all parameters (except <code>IDs</code> ) are ignored.  <code>ParentEventId = {Guid.Empty}</code> is interpreted in the same way as <code>{null}</code> (for Delphi compatibility).
<code>DateTime StartDate</code>	The start date of the time period for which the report is generated.
<code>DateTime EndDate</code>	The end date of the time period for which the report is generated.
<code>DateTime StartTime0</code>	The start time of the hour range within the time period. Only events that occurred within this range will be included in the report.
<code>DateTime EndTime0</code>	The end time of the hour range within the time period. Only events that occurred within this range will be included in the report.
<code>DateTime StartTime1</code>	The start time of the second hour range within the time period. It is used if the end of the day falls within the range. *
<code>DateTime EndTime1</code>	The end time of the second hour range within the time period. It is used if the end of the day falls within the range. *
<code>Guid[] Territories</code>	An array of territory keys for which the report will be generated.
<code>Guid[] Operators</code>	The parameter is intended for internal use.
<code>UInt[] TransactionTypes</code>	An array of transaction type keys. One or more transaction keys from the table below can be used as a parameter.
<code>Guid[] Organizations</code>	An array of company keys for which the report will be generated.
<code>Guid[] Users</code>	An array of user keys, for whose events the report will be generated.
<code>bool EventsWithoutUser</code>	The parameter is intended for internal use.
<code>int MaxResultSize</code>	The maximum number of events displayed in the report. When passing null, the default number of events (5000) is displayed.

\* If the moment of the end of the day (24.00) falls within the hour range, then such a range should be divided into two: from XX hours to 24.00 and from 00.00 to YY. In this case, the `StartTime0` and `EndTime0` parameters are used for the first part of the range, and `StartTime1` and `EndTime1` for the second part.

### Transaction Keys:

Key	Description of the transaction
590144	Normal key entry
590152	Actual entry
590145	Normal key exit
590153	Actual exit
590146	Exit out of time profile
590244	Normal visitor exit
590245	Actual exit of the visitor

### HardwareState

Base class: [BaseObject](#).

Class, used to describe the state of the territory.

<code>Guid</code> TerritoryID	The territory unique.
<code>ulong</code> State	The set of territory states is a 4-byte bit mask.

### TransactionClass

Base class: [BaseObject](#).

Class, used to describe the categories of system events.

Each event in the ParsecNET system belongs to one or more categories.

<code>Long</code> ID	The category ID of the system event.  Always equal to 1 bitwise left shifted by N positions, where N ranges from 0 to 64.  Thus, each category is 1 bit in the event bitmask.
<code>Bool</code> NAME	The name of the system event category.
<code>Bool</code> IS_USER	Custom category attribute: 0 - system category; 1 - a category created by the user.

### TransactionType

Base class: [BaseObject](#).

Class, used to describe the types of system events.

<code>Int</code> ID	The unique event identifier.
<code>long</code> CLASS_MASK	A bit mask that specifies the set of categories to which this event belongs.
<code>String</code> NAME	The name of the event.



### 3. FUNCTIONS

#### VERSION

##### GetVersion function

```
string GetVersion()
```

Parameters: no parameters.

Result: It returns the version of the integration service.

Description: The function returns the version of the integration service.

#### SESSION

##### GetDomains function

```
Domain[] GetDomains ()
```

Parameters: No

Result: It returns an array [Domain](#).

Description: It returns an array of organizations.

##### OpenSession function

```
SessionResult OpenSession( string domain, string userName, string password )
```

Options:

<code>string domain</code>	Organization name to login. An empty string can be used to enter the system organization.
<code>string userName</code>	Operator name.
<code>string password</code>	Operator password.

Result: It returns the class.

Description: This function used to authenticate the operator in the integration service. The resulting session key used in the future to perform all operations. Each session opens for 5 minutes, the session time extends when any operation is performed on the integration server.

##### OpenSessionWithInLocale function

```
SessionResult OpenSessionWithInLocale( string domain, string userName, string password, string locale )
```

Options:

<code>string domain</code>	Organization name to login. An empty string can be used to enter the system organization.
<code>string userName</code>	Operator name.
<code>string password</code>	Operator password.
<code>string locale</code>	Required language. Possible values are "ru-RU", "en-US" or "es-ES".

Result: It returns the class.

Description: This function used to authenticate the operator in the integration service specifying the interface language. The resulting session key used in the future to perform all operations. Each session opens for 5 minutes, the session time extends when any operation is performed on the integration server.

### ContinueSession function

```
int ContinueSession( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: The following value is returned as a result: 0 - if the operation was successful; 1 - if the operation was completed with errors.

Description: Used to extend the session.

### CloseSession function

```
void CloseSession( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: –

Description: Used to close the session.

### CheckRole function

```
BaseResult CheckRole( Guid sessionID, string roleName )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string roleName</code>	Name of the right to perform operations.

Results: Returns [BaseResult](#).

Description: Checks the availability of actions for the current operator. The following rights are used in the integration service:

"EmployeeReader"	The right to obtain information about the personnel and structure of departments.
"EmployeeWriter"	The right to delete / change information about the personnel and the structure of departments.
"PersonDelete"	The right to delete subjects of access.
"AccessGroupReader"	The right to get information about access groups.
"AccessGroupWriter"	The right to change access groups.
"GuestReader"	The right to receive information about visitors.
"VisitorRequestCreator"	The right to create, edit and delete requests and visitors.
"VisitorRequestCoordinator"	The right to approve requests for visitors.
"VisitorPassDistributor"	The right to issue visitor passes and close requests.

"TimesheetReader"	The right to view schedule details.
"TimesheetWriter"	The right to create, delete and modify schedules.
"HardwareControl"	The right to send commands to control access equipment.
"AlarmHardwareControl"	The right to send commands for arming and disarming.
"HardwareWriter"	The "Full access" right to the Hardware Editor (in part of managing the roles of the group passage).
"MonitorReader"	The right to receive system events and equipment statuses in real time.
"VideoVerification"	Right to Confirm Access.

### GetObjectName function

`StringResult` GetObjectName ( `Guid` sessionID, `Guid` objectID )

Options:

<code>Guid</code> sessionID	The session unique key.
<code>Guid</code> objectID	The object unique key.

Result: It returns [StringResult](#).

Description: It returns the name of any object in the system by its key.

## EQUIPMENT

### SendHardwareCommand function

`BaseResult` SendHardwareCommand( `Guid` sessionID, `Guid` territoryID, `int` command )

Options:

<code>Guid</code> sessionID	The unique session key.
<code>Guid</code> territoryID	The unique territory key.
<code>int</code> command	The code of the command to be sent.

Result: It returns [BaseResult](#).

Description: Sends a command to the device represented by the territory. Command codes and devices are described in the table below.

Commands		Parsec access controller	AS-08 security area	Bolid security section
Code	Description			
1	Turn on the entry relay * \ Open the door	✓		
2	Turn on the exit relay *	✓		
4	Close the door	✓		
8	Set a relative blocking	✓		
16	Cancel a relative blocking	✓		

32	Set an absolute blocking	✓		
64	Cancel an absolute blocking	✓		
128	Set to arm	✓	✓	✓
256	Disarm	✓	✓	✓
512	Enable additional relay	✓		
1024	Disable additional relay	✓		
2048	Reset anti-passback	✓		
* Commands work only for controllers in turnstile mode.				

## SendVerificationCommand function

`BaseResult` SendVerificationCommand( `Guid` sessionID, `Guid` territoryID, `Guid` personID, `bool` passAllow )

Options:

<code>Guid</code> sessionID	The unique session key.
<code>Guid</code> territoryID	The territory unique key.
<code>Guid</code> personID	The access subject's unique key.
<code>bool</code> passAllow	Parameter of allowing/prohibiting passage.

Result: It returns the `BaseResult` array.

Description: Sends a command to confirm or deny passage to the device represented by the territory. The software passage confirmation must be selected in the controller settings.

## GetHardwareState function

`HardwareState[]` GetHardwareState( `Guid` sessionID, `Guid[]` territoryIDs )

Options:

<code>Guid</code> sessionID	The unique session key.
<code>Guid[]</code> territoryIDs	An array of territory keys.

Result: It returns an array of Hardware State bitmasks (each mask is 8 bytes in size).

Description: Gives a set of states for the selected territories. The status codes are described in the tables below. The function is applicable only to the territories generated by the NC series access controllers, the AC-08 security controller and the Bolid security system.

### Territory Status Bits for NC Series Controllers

Bit №	Meaning	Status at bit value	
		0	1
0	Battery	Discharged	Norm
1	Mains supply	Disabled	Norm
2	Battery	Faulty	Norm

3	Casing	Open	Closed
4	Entry relay * \ Lock	Off * \ Closed	On * \ Opened
5	Exit relay *	Off *	On *
6	Additional relay	Off	On
7	Card collector relay	Off	On
8	Absolute blocking	Off	On
9	Relative blocking	Off	On
10	Emergency door opening	Off	On
11	Security	Disarmed	Armed
12	Security sensor	Norm	Activated
13	Entry sensor ** \ Door contact	Norm	Activated
14	Exit sensor **	Norm	Activated
28	Switched off	No	Yes
31	Not available	No	Yes
* Values are indicated for controllers in turnstile mode.			
** Values shown are for NC-100K controller only.			

**The Status Bits of the territory - security area of the AC-08 controller and the Bolid security system**

Bit №	Meaning	Status at bit value	
		0	1
0	Battery*	Discharged	Norm
1	Mains supply *	Disabled	Norm
3	Casing*	Open	Closed
11	Security	Disarmed	Armed
12	Security sensor	Norm	Activated
24	Needs attention	No	Yes
28	Switched off	No	Yes
31	Not available	No	Yes
* Values are indicated for the AC-08 controller.			

## SUBDIVISIONS

### GetRootOrgUnit function

`OrgUnit GetRootOrgUnit( Guid sessionID )`

Options:

`Guid sessionID`      The unique session key.

Result: It returns the root OrgUnit.

Description: this function is intended to get the root object of the subdivision tree.

### GetOrgUnitsHierarhy function

`OrgUnit[] GetOrgUnitsHierarhy( Guid sessionID )`

Options:

`Guid sessionID`      The unique session key.

Result: OrgUnit array.

Description: It returns the complete hierarchy of divisions.

### GetOrgUnitsHierarhyWithPersons function

`BaseObject[] GetOrgUnitsHierarhyWithPersons( Guid sessionID )`

Options:

`Guid sessionID`      The unique session key.

Result: BaseObject array, array elements can be OrgUnit or Person.

Description: It returns the complete hierarchy of departments along with the employees.

### GetOrgUnitsHierarhyWithVisitors function

`BaseObject[] GetOrgUnitsHierarhyWithVisitors( Guid sessionID )`

Options:

`Guid sessionID`      The unique session key.

Result: BaseObject array, array elements can be OrgUnit or Person.

Description: It returns the complete hierarchy of departments along with the visitors.

### GetOrgUnitsHierarhyWithVehicle function

`BaseObject[] GetOrgUnitsHierarhyWithVehicle( Guid sessionID )`

Options:

`Guid sessionID`      The unique session key.

Result: BaseObject array, array elements can be OrgUnit or Person.

Description: It returns the complete hierarchy of departments along with vehicles.

## GetOrgUnitSubItems function

```
BaseObject[] GetOrgUnitSubItems( Guid sessionID, Guid orgUnitID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: BaseObject array, array elements can be BaseOrgUnit or BasePerson.

Description: It returns an array of departments and employees belonging to the department with the specified key.

## GetOrgUnitSubItemsHierarchyWithPersons function

```
BaseObject[] GetOrgUnitSubItemsHierarchyWithPersons( Guid sessionID, Guid orgUnitID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: BaseObject array, array elements can be OrgUnit or Person.

Description: It returns the complete hierarchy of departments and their employees, starting with the department with the specified key.

## GetOrgUnitSubItemsHierarchyWithVisitors function

```
BaseObject[] GetOrgUnitSubItemsHierarchyWithVisitors( Guid sessionID, Guid orgUnitID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: BaseObject array, array elements can be OrgUnit or Person.

Description: It returns the complete hierarchy of departments, starting with the department with the specified key, and the visitors to those departments.

## GetOrgUnitSubItemsHierarchyWithVehicle function

```
BaseObject[] GetOrgUnitSubItemsHierarchyWithVehicle( Guid sessionID, Guid orgUnitID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: BaseObject array, array elements can be OrgUnit or Person.

Description: It returns the complete hierarchy of departments, starting with the department with the specified key, and the vehicles of these departments.

## GetOrgUnit function

`OrgUnit GetOrgUnit( Guid sessionID, Guid orgUnitID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: It returns the OrgUnit for the department with the specified key, or 'null' if the specified department was not found.

Description: It returns information about the department with the specified key.

## EDITING A DEPARTMENT

### CreateOrgUnit function

`GuidResult CreateOrgUnit( Guid sessionID, OrgUnit orgUnit )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>OrgUnit orgUnit</code>	Department parameters.

Result: It returns GuidResult.

Description: It creates a department with the specified data. It returns the key of the newly created unit. If the ID field in the OrgUnit structure is Guid.Empty (00000000-0000-0000-0000-000000000000), then the ID is automatically generated and returned. Otherwise, the specified ID value is used.

### OpenOrgUnitEditingSession function

`GuidResult OpenOrgUnitEditingSession( Guid sessionID, Guid orgUnitID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: It returns GuidResult.

Description: It opens a department editing session. It returns the key of the newly created edit session of the department.

### CloseOrgUnitEditingSession function

`void CloseOrgUnitEditingSession( Guid orgUnitEditSessionID )`

Options:

<code>Guid orgUnitEditSessionID</code>	A unique key for the department editing session.
--	--

Result: –

Description: It closes the department editing session.



## SaveOrgUnit function

```
BaseResult SaveOrgUnit( Guid orgUnitEditSessionID, BaseOrgUnit orgUnit )
```

Options:

<code>Guid orgUnitEditSessionID</code>	A unique key of the department editing session.
<code>BaseOrgUnit orgUnit</code>	Department parameters.

Result: It returns BaseResult.

Description: It changes the parameters of a department. The orgUnit parameter can be BaseOrgUnit or OrgUnit.

When passing the OrgUnit structure as unit parameters in a request, you must specify the *xsi: type = "OrgUnit"* attribute in the *orgUnit* element.

The editing session is closed after successfully saving the department.

## DeleteOrgUnit function

```
BaseResult DeleteOrgUnit( Guid sessionID, Guid orgUnitID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.

Result: It returns BaseResult.

Description: It deletes the department with the specified key.

# PERSONNEL

## GetPersonExtraFieldTemplates function

```
PersonExtraFieldTemplate[] GetPersonExtraFieldTemplates( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns an array of PersonExtraFieldTemplate.

Description: It returns a set of templates for additional fields for employees.

## GetVisitorExtraFieldTemplates Function

```
PersonExtraFieldTemplate[] GetVisitorExtraFieldTemplates( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns an array of PersonExtraFieldTemplate.

Description: It returns a set of templates for additional fields for visitors.

## GetVehicleExtraFieldTemplates function

```
PersonExtraFieldTemplate[] GetVehicleExtraFieldTemplates( Guid  
sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns an array of PersonExtraFieldTemplate.

Description: It returns a set of templates of additional fields for vehicles.

## FindPeople function

```
BasePerson[] FindPeople( Guid sessionID, string lastname, string  
firstname, string middlename )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string lastname</code>	The value of the surname to search for.
<code>string firstname</code>	The value of the name to search for.
<code>string middlename</code>	The value of the middle name to search for.

Result: It returns an array of Person.

Description: It returns a set of employees matching the passed criteria. The search can be performed either by any parameter separately or by all parameters. At least one parameter must be specified.

## FindVisitors function

```
BasePerson[] FindVisitors( Guid sessionID, string lastname, string  
firstname, string middlename )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string lastname</code>	The value of the surname to search for.
<code>string firstname</code>	The value of the name to search for.
<code>string middlename</code>	The value of the middle name to search for.

Result: It returns an array of Person.

Description: It returns a set of visitors that match the passed criteria. The search can be performed either by any parameter separately or by all parameters. At least one parameter must be specified.

## FindVehicle function

```
BasePerson[] FindVehicle( Guid sessionID, string number, string model,  
string color )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string number</code>	The value of the car plate number for the search.

<code>string</code> model	The value of the car model to search for.
<code>string</code> color	The value of the color to search for.

Result: It returns an array of Person.

Description: It returns a set of vehicles that match the passed criteria. The search can be performed either by any parameter separately or by all parameters. At least one parameter must be specified.

### FindPersonByIdentifier function

[Person](#) FindPersonByIdentifier( `Guid` sessionId, `string` cardCode )

Options:

<code>Guid</code> sessionId	The unique session key.
<code>string</code> cardCode	Card ID code in uppercase hexadecimal format (Example: A12345BCF).

Result: It returns Person.

Description: It returns information about the access principal with the specified identifier code.

### PersonSearch function

[Person](#)[] PersonSearch( `Guid` sessionId, `Guid` fieldID, `int` relation, `object` value, `object` value1 )

Options:

<code>Guid</code> sessionId	The unique session key.
<code>Guid</code> fieldID	The unique key of the field used for the search.
<code>int</code> relation	Search criterion.
<code>object</code> value	The value to search for or the first value for the search criterion "between".
<code>object</code> value1	The second value for the search criterion "between".

Result: It returns an array of Person.

Description: It returns information about access subjects matching the search parameters.

### Field keys used for searches

Guid	Description	Value type
0de358e0-c91b-4333-b902-000000000003	Surname / Licence plate	<code>string</code>
0de358e0-c91b-4333-b902-000000000001	Name / Vehicle Model	<code>string</code>
0de358e0-c91b-4333-b902-000000000002	Middle name / Vehicle color	<code>string</code>
0de358e0-c91b-4333-b902-000000000006	Report card	<code>string</code>

Guid	Description	Value type
0de358e0-c91b-4333-b902-000000000004	Department	string
0a679144-d5ce-476d-a56e-0a696f079b71	Department description	string
0de358e0-c91b-4333-b902-00000000000a	Access group	string
0de358e0-c91b-4333-b902-000000000005	Card code	string
644E1B95-E87B-415D-91BF-C3242B6C3AEA	Identifier name	string
0de358e0-c91b-4333-b902-000000000007	Time of action from	DateTime
0de358e0-c91b-4333-b902-000000000008	Time of action to	DateTime
6FCFA1BB-9624-4248-A2D5-AA84901C53C8	Blacklisted subject	bool
07AF86B3-23FC-44EF-8438-6EE601B2FCB0	Access is denied	bool

You can also use a unique key of additional field template with the appropriate value type.

### Search criteria

Value	Description
0	Equal to (=)
1	Less than or equal (<=)
2	Less (<)
3	More or equal (> =)
4	More (>)
5	Between
6	Contains
7	Empty
8	Not empty

### GetPerson function

`PersonWithPhoto` GetPerson( `Guid` sessionID, `Guid` personID )

Options:

`Guid` sessionID                      The unique session key.  
`Guid` personID                        The unique access subject key.

Result: It returns PersonWithPhoto.

Description: It returns information about the access subject with the specified key.

### GetMultiplePersons function

`PersonWithPhoto[]` GetMultiplePersons( `Guid` sessionID, `Guid[]` personIDs )

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid[] personIDs</code>	An array of access subjects keys.

Result: It returns an array of `PersonWithPhoto`.

Description: It returns information about access subjects with the specified keys.

## GetPersonsChangedAfter function

`Person[]` `GetPersonsChangedAfter( Guid sessionID, Guid orgID, DateTime dateFrom, bool includeSubOrg )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgID</code>	The unique department key .
<code>DateTime dateFrom</code>	Start date for analysis.
<code>bool includeSubOrg</code>	Search attribute for nested divisions.

Result: It returns an array of `Person`.

Description: It returns a set of access subjects (employee, visitor, car), whose data has been changed since the specified date. The search is performed either in the specified department, or in the specified and all nested ones.

## GetPersonExtraFieldValue function

`ObjectResult` `GetPersonExtraFieldValue( Guid sessionID, Guid personID, Guid templateID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique access subject key.
<code>Guid templateID</code>	The unique template key.

Result: It returns `ObjectResult`.

Description: It returns the value of the specified additional access subject field. The type of the value is determined by the type of the field.

## GetPersonExtraFieldValues function

`ExtraFieldValue[]` `GetPersonExtraFieldValues( Guid sessionID, Guid personID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique access subject key.

Result: It returns an `ExtraFieldValue` array.

Description: It returns an array of values of additional fields of the access subject.

## GetPersonExtraFieldValueString function

`StringResult` `GetPersonExtraFieldValueString( Guid sessionID, Guid personID, Guid templateID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique access subjects key.
<code>Guid templateID</code>	The unique template key.

Result: It returns `StringResult`.

Description: It returns the value of the access subject additional field as a formatted string.

### **ValidateExtraFieldValue function**

```
BaseResult ValidateExtraFieldValue( Guid sessionID, ExtraFieldValue value )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>ExtraFieldValue value</code>	The value of the additional field.

Result: It returns `BaseResult`.

Description: Checks the validity of the additional field value.

In the request, in the *value* element, you must specify the type of the value in the *xsi:type* attribute. For example, *xsi:type="xsd:string"* (see example from the description of the function [SetPersonExtraFieldValue](#)).

### **GetPersonScheduleFixes function**

```
PersonScheduleFix[] GetPersonScheduleFixes( Guid sessionID, Guid personID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique employee key.

Result: It returns the `PersonScheduleFix` array.

Description: It returns a list of existing corrections to the employee's working time.

### **AddPersonScheduleFix function**

```
GuidResult AddPersonScheduleFix( Guid sessionID, PersonScheduleFix fix )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>PersonScheduleFix fix</code>	Working time correction parameters.

Result: It returns `GuidResult`.

Description: It adds an employee working time correction.

## SavePersonScheduleFix function

```
BaseResult SavePersonScheduleFix( Guid sessionID, PersonScheduleFix  
fix )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>PersonScheduleFix fix</code>	Working time correction parameters.

Result: It returns BaseResult.

Description: It changes the parameters of an existing employee working time correction.

## DeletePersonScheduleFix function

```
BaseResult DeletePersonScheduleFix( Guid sessionID, Guid personID,  
Guid fixID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique employee key.
<code>Guid fixID</code>	The unique correction key.

Result: It returns BaseResult.

Description: It removes an employee's working time correction.

## GetPersonWorktimeSchedule function

```
GuidResult GetPersonWorktimeSchedule( Guid sessionID, Guid personID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique employee key.

Result: It returns GuidResult.

Description: It returns the unique key of the working time schedule assigned to the employee. If no schedule is assigned, returns Guid.Empty.

## SetPersonWorktimeSchedule function

```
BaseResult SetPersonWorktimeSchedule( Guid personEditSessionID, Guid  
scheduleID )
```

Options:

<code>Guid personEditSessionID</code>	The unique employee editing session key.
<code>Guid scheduleID</code>	A unique working time schedule key.

Result: It returns BaseResult.

Description: It sets a personal working time schedule for an employee with the specified key.

## EDITING THE ACCESS SUBJECT

### CreatePerson function

```
GuidResult CreatePerson( Guid sessionID, Person person )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Person person</code>	Employee's data.

Result: It returns GuidResult.

Description: It creates an employee with the specified data. The 'person' parameter can be Person or PersonWithPhoto. It returns the key of the newly created employee. If the ID field in the 'person' structure is Guid.Empty (00000000-0000-0000-0000-000000000000) then the ID is automatically generated and returned. Otherwise, the specified ID value is used.

### CreateVisitor function

```
GuidResult CreateVisitor( Guid sessionID, Person person )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Person person</code>	Visitor's data.

Result: It returns GuidResult.

Description: It creates a visitor with the specified data. It returns the key of the newly created employee. If the ID field in the 'person' structure is Guid.Empty (00000000-0000-0000-0000-000000000000) then the ID is automatically generated and returned. Otherwise, the specified ID value is used.

### CreateVehicle function

```
GuidResult CreateVehicle( Guid sessionID, Person person )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Person person</code>	Vehicle's data.

Result: It returns GuidResult.

Description: It creates a car with the specified data. It returns the key of the newly created employee. If the ID field in the 'person' structure is Guid.Empty (00000000-0000-0000-0000-000000000000) then the ID is automatically generated and returned. Otherwise, the specified ID value is used.

### OpenPersonEditingSession function

```
GuidResult OpenPersonEditingSession( Guid sessionID, Guid personID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid personID</code>	The unique access subject key.

Result: It returns GuidResult.



Description: It opens an access subject editing session. It returns the key of the newly created session.

### ClosePersonEditingSession function

```
void ClosePersonEditingSession( Guid personEditSessionID )
```

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
---	---

Result: Empty.

Description: It closes the editing session of the access principal.

### SavePerson function

```
BaseResult SavePerson( Guid personEditSessionID, BasePerson person )
```

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
<code>BasePerson</code> <code>person</code>	Access subject data.

Result: It returns BaseResult.

Description: It changes the data of the access subject. BasePerson, Person, PersonWithPhoto can be used as a 'person' parameter.

When passing a Person or PersonWithPhoto structure as an access subject in a request, you must specify the class name in the *xsi:type* attribute of the *person* element.

In case of successful saving, the editing session of the access subject is closed.

### SetPersonPhoto function

```
BaseResult SetPersonPhoto( Guid personEditSessionID, byte[]  
photoByteArray )
```

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
<code>byte[]</code> <code>photoByteArray</code>	Photo of the access subject.

Result: It returns BaseResult.

Description: it saves a photo of the access subject. In case of successful saving, the editing session of the access subject is closed.

### SetPersonOrgUnit function

```
BaseResult SetPersonOrgUnit( Guid personEditSessionID, Guid orgUnitID )
```

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
<code>Guid</code> <code>orgUnitID</code>	The unique department key.

Result: It returns BaseResult.

Description: it sets the department for the access subject. In case of successful saving, the editing session of the access subject is closed.

### SetPersonExtraFieldValue function

`BaseResult SetPersonExtraFieldValue( Guid personEditSessionID, Guid templateID, object value )`

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
<code>Guid</code> <code>templateID</code>	The unique key of the addition field template.
<code>object</code> <code>value</code>	The value of the additional field.

Result: It returns BaseResult.

Description: It sets the new value for the specified additional field of the access subject.

In the request, you must specify a type of the *value* element in *xsi:type* attribute. For example, *xsi:type="xsd: string"*.

An example of a valid request:

```
<?xml version="1.0" encoding="utf-8"?>

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <soap:Body>

    <SetPersonExtraFieldValue
xmlns="http://parsec.ru/Parsec3IntergationService">

      <personEditSessionID>b2aea303-f736-4a29-b835-
670c2384551f</personEditSessionID>

      <templateID>4d7371ef-e856-4371-88da-
0365277c2032</templateID>

      <value xsi:type="xsd:string"> valid value</value>

    </SetPersonExtraFieldValue>

  </soap:Body>

</soap:Envelope>
```

### SetPersonExtraFieldValues function

`BaseResult SetPersonExtraFieldValues( Guid personEditSessionID, ExtraFieldValue[] values )`

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
<code>ExtraFieldValue[]</code> <code>values</code>	An array of additional field values.

Result: It returns BaseResult.

Description: It sets values for additional fields.

In the request, you must specify the value type in the *xsi:type* attribute in each *values* element of the array. For example, *xsi:type* = "*xsd:string*" (see example in the description of the function [SetPersonExtraFieldValue](#)).

## DeletePerson function

```
BaseResult DeletePerson( Guid sessionId, Guid personID )
```

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Guid personID</code>	The unique access subject key.

Result: It returns BaseResult.

Description: It removes the access subject with the specified key.

## BlockPerson function

```
BaseResult BlockPerson( Guid personEditSessionID, int typeBlock )
```

Options:

<code>Guid personEditSessionID</code>	The unique key of the access subject data editing session.
<code>int typeBlock</code>	Blocking type.

Result: It returns BaseResult.

Description: It blocks the access subject according to the specified blocking type.

The blocking type values are described in the table:

Values of typeBlock	Description
1	Setting the "Exit Denied" privilege.
2	Setting the "Entry Denied" privilege.
3	Setting the "Entry Denied" and "Exit Denied" privileges.

The access subject data editing session is closed in case of successful execution of the function.

## UnblockPerson function

```
BaseResult UnblockPerson( Guid personEditSessionID )
```

Options:

<code>Guid personEditSessionID</code>	The unique key of the access subject data editing session.
---------------------------------------	--

Result: It returns BaseResult.

Description: It clears all blocks from the previously blocked access subject.

The access subject data editing session is closed in case of successful execution of the function.

## TOPOLOGY

### GetRootTerritory function

`Territory` GetRootTerritory( `Guid` sessionID )

Options:

`Guid` sessionID                      The unique session key.

Result: It returns the root Territory.

Description: This function is intended to get the root object of the territory tree.

### GetTerritoriesHierarhy function

`Territory`[] GetTerritoriesHierarhy( `Guid` sessionID )

Options:

`Guid` sessionID                      The unique session key.

Result: It returns an array of Territory; values can be TerritoryWithComponent.

Description: It returns the complete hierarchy of territories.

### GetTerritorySubItems function

`BaseTerritory`[] GetTerritorySubItems( `Guid` sessionID, `Guid` TerraID )

Options:

`Guid` sessionID                      The unique session key.

`Guid` TerraID                        The unique territory key.

Result: It returns an array of BaseTerritory; values can be TerritoryWithComponent.

Description: It returns a list of territories belonging to a territory with the specified key.

### GetTerritory function

`Territory` GetTerritory( `Guid` sessionID, `Guid` territoryID )

Options:

`Guid` sessionID                      The unique open session key.

`Guid` territoryID                    The unique territory key.

Result: It returns Territory; the value can be TerritoryWithComponent.

Description: This function is intended to get a description of the territory by its key.

## IDENTIFIERS AND ACCESS

### GetPersonIdentifiers function

`Identifier`[] GetPersonIdentifiers( `Guid` sessionID, `Guid` personID )

Options:

`Guid` sessionID                      The unique session key.

`Guid` personID                        The unique access subject key.

Result: Massiv 'Identifier'; array elements can be Identifier or IdentifierTemp.

Description: It returns an array of identifiers for the specified access subject.

## DeleteIdentifier function

```
BaseResult DeleteIdentifier( Guid sessionID, string Code )
```

Options:

Guid sessionID	The unique session key.
string Code	ID code.

Result: It returns BaseResult.

Description: Deletes the identifier with the specified code.

## AddPersonIdentifier function

```
BaseResult AddPersonIdentifier( Guid personEditSessionID,  
BaseIdentifier identifier )
```

Options:

Guid personEditSessionID	The unique key of the access subject editing session.
BaseIdentifier identifier	Identifier parameters.

Result: It returns BaseResult.

Description: It adds an identifier to the access subject or modifies the data of an existing identifier. The 'identifier' parameter can be BaseIdentifier, Identifier, IdentifierTemp. The IS\_PRIMARY element is used only in a positive sense, i.e. when this feature is set, the corresponding identifier becomes primary, and when this feature is removed from the primary identifier, nothing changes when it is saved in the sense of the primary identifier. The PRIVILEGE\_MASK element is ignored.

When Identifier or IdentifierTemp structure is transmitting as identifier in a request, the *xsi:type="Identifier"* or *xsi:type="IdentifierTemp"* attribute must be specified in the element *identifier*, respectively.

An example of a valid request:

```
<?xml version="1.0" encoding="utf-8"?>  
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xmlns:xsd="http://www.w3.org/2001/XMLSchema">  
  <soap:Body>  
    <AddPersonIdentifier xmlns="http://parsec.ru/ParsecIntergationService">  
      <personEditSessionID>a0c8c558-0893-4fb6-b2ea-  
528d2f08edd2</personEditSessionID>  
      <identifier xsi:type="Identifier">  
        <CODE>11111112</CODE>  
        <PERSON_ID>be4d3ce8-d830-4796-9197-7fa65c78d13f</PERSON_ID>  
        <IS_PRIMARY>>false</IS_PRIMARY>  
        <ACCGROUP_ID>11111111-2222-3333-4444-  
555555555555</ACCGROUP_ID>
```

```

</identifier>
</AddPersonIdentifier>
</soap:Body>
</soap:Envelope>

```

## ChangePersonIdentifier function

```
BaseResult ChangePersonIdentifier( Guid personEditSessionID,
BaseIdentifier identifier )
```

Options:

<code>Guid</code> <code>personEditSessionID</code>	The unique key of the access subject editing session.
<code>BaseIdentifier</code> <code>identifier</code>	Identifier parameters.

Result: It returns BaseResult.

Description: Modifies the parameters of an existing identifier. The parameter 'identifier' can be BaseIdentifier, Identifier, IdentifierTemp. The IS\_PRIMARY element is used only in a positive sense, i.e. when this feature is set, the corresponding identifier becomes primary, and when this feature is removed from the primary identifier, nothing changes when it is saved in the sense of the primary identifier. The PRIVILEGE\_MASK element is ignored.

When Identifier or IdentifierTemp structure is transmitting as identifier in a request, the *xsi:type="Identifier"* or *xsi:type="IdentifierTemp"* attribute must be specified in the element *identifier*, respectively.

An example of a valid request:

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <ChangePersonIdentifier xmlns="http://parsec.ru/ParsecIntergationService">
      <personEditSessionID>a0c8c558-0893-4fb6-b2ea-
528d2f08edd2</personEditSessionID>
      <identifier xsi:type="Identifier">
        <CODE>11111112</CODE>
        <PERSON_ID>be4d3ce8-d830-4796-9197-7fa65c78d13f</PERSON_ID>
        <IS_PRIMARY>>false</IS_PRIMARY>
        <ACCGROUP_ID>11111111-2222-3333-4444-
555555555555</ACCGROUP_ID>
      </identifier>
    </ChangePersonIdentifier>
  </soap:Body>
</soap:Envelope>

```

## SetIdentifierPrivileges function

```
BaseResult SetIdentifierPrivileges( Guid sessionID, string cardCode,  
long privilegesMask )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string cardCode</code>	ID code.
<code>long privilegesMask</code>	Privilege bitmask.

Result: It returns BaseResult.

Description: The function sets a set of privileges using a bit mask. The mask bit values are described in the table:

Bit №	Description
0	Mute the door sound.
1	Security management.
2	Pass when blocking.
3	Receive alarm.
4	Arming.
5	Disarming.
6	Antipassback pass.
7	Guest card.
8	Privilege card.
9	The exit is prohibited.
10	Exit out of the time profile is allowed.
11	Access control.
12	-
13	Owner's card.
14	Do not use a pass counter.
15	No entry.
16	Unaccompanied passage is prohibited.
17	Strictly control the key return time (keyholder).

*Example:* to set the "Pass when blocking" and "Antipassback pass" privileges to identifier, the value of the *privilegesMask* parameter must be  $2^2 + 2^6 = 4 + 64 = 68_{10} = 1000100_2$

## GetIdentifierExtraData function

```
IdentifierExData GetIdentifierExtraData( Guid sessionID, string  
cardCode )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

`string` cardCode                      Card identifier code in hexadecimal format.

Result: It returns IdentifierExData.

Description: It gives information about an identifier additional properties.

### SetIdentifierExtraData function

```
BaseResult SetIdentifierExtraData( Guid sessionID, string cardCode,
IdentifierExData exData )
```

Options:

<code>Guid</code> sessionID	The unique session key.
<code>string</code> cardCode	Card identifier code in hexadecimal format.
<code>IdentifierExData</code> exData	Parameters of additional identifier properties.

Result: It returns BaseResult.

Description: It sets the values of the additional properties of the identifier.

### GetPassageRoles function

```
PassageRole[] GetPassageRoles( Guid sessionID )
```

Options:

<code>Guid</code> sessionID	The unique session key.
-----------------------------	-------------------------

Result: It returns a PassageRole array.

Description: Lists the roles of the group pass.

### CreatePassageRole function

```
GuidResult CreatePassageRole( Guid sessionID, PassageRole role )
```

Options:

<code>Guid</code> sessionID	The unique session key.
<code>PassageRole</code> role	The parameters of the role being created.

Result: It returns GuidResult.

Description: It creates a group passage role with the specified parameters. It returns the key of the created role. If the ID field in the PassageRole structure is Guid.Empty (00000000-0000-0000-0000-000000000000), then the ID is automatically generated and returned. Otherwise, the specified ID value is used.

### SavePassageRole function

```
BaseResult SavePassageRole( Guid sessionID, PassageRole role )
```

Options:

<code>Guid</code> sessionID	The unique session key.
<code>PassageRole</code> role	Role parameters.

Result: It returns BaseResult.

Description: It modifies the parameters of a role.



## DeletePassgeRole function

`BaseResult DeletePassageRole( Guid sessionID, Guid roleID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid roleID</code>	The unique role key.

Result: It returns BaseResult.

Description: it removes the group pass role with the specified key.

## DeletePassgeRole function

`BaseResult DeletePassageRole( Guid sessionID, Guid roleID )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid roleID</code>	The unique key for the role.

Result: It returns BaseResult.

Description: Removes the group pass role with the specified key.

## GetUnique4bCardCode function

`StringResult GetUnique4bCardCode( Guid sessionID )`

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns StringResult.

Description: It returns a unique unused identifier code in 4-byte size.

## GetCardCodeFromUID function

`StringResult GetCardCodeFromUID( Guid sessionID, string UID, bool reverseByteOrder )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string UID</code>	The card UID in hexadecimal format.
<code>bool reverseByteOrder</code>	Indication of the reverse byte order in the UID.

Result: It returns StringResult.

Description: It returns the Mifare card code based on the UID. The code returned depends on the parameter values specified in the "Configuring Desktop Readers" section.

## GenerateParsecQRCode function

```
StringResult GenerateParsecQRCode( Guid sessionID, string cardCode )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string cardCode</code>	ID code.

Result: It returns StringResult.

Description: It returns a string containing an encrypted identifier code for generating a QR code. The generated QR code can be used by Parsec readers.

## SCHEDULES AND ACCESS GROUPS

### GetAccessSchedules function

```
Schedule[] GetAccessSchedules( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns a Schedule array; the value can be AccessSchedule.

Description: It gives a set of system access schedules.

### GetWorktimeSchedules function

```
Schedule[] GetWorktimeSchedules( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns a Schedule array; the value can be WorktimeSchedule.

Description: It gives a set of working time schedules.

### GetScheduleIntervals function

```
TimeInterval[] GetScheduleIntervals( Guid sessionID, Guid scheduleID, DateTime from, DateTime to )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid scheduleID</code>	A unique key for the schedule.
<code>DateTime from</code>	Start date for analysis.
<code>DateTime to</code>	End date for analysis.

Result: It returns an array of TimeInterval; values can be WorktimeInterval.

Description: It gives a set of intervals of the specified schedule, limited by the start and end date. The function takes into account only the DATE part of the parameters.

## CreateAccessSchedule function

```
GuidResult CreateAccessSchedule( Guid sessionID, AccessSchedule  
schedule, ScheduleDay[] days )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>AccessSchedule schedule</code>	Schedule parameters.
<code>ScheduleDay[] days</code>	An array of templates for the days of the schedule cycle.

Result: It returns GuidResult.

Description: It creates a new access schedule with the specified parameters and the specified cycle in the system. It returns the key of the newly created access schedule.

*When creating a schedule, the following conditions must be strictly observed:*

- *For the weekly access schedule, the "Apply with replacement" holiday application type is required.*
- *The holiday application type "Apply with insert" cannot be set for the weekly work schedule.*
- *For weekly schedules, the 'days' template array must contain 7 elements.*
- *The elements of the 'days' array must have the indexes of the day in the schedule cycle. The index of the first day in the cycle has a value of "1". All elements of the cycle must have the same date.*
- *For weekly schedules, the cycle start date must be Monday.*
- *There should be no more than three unique day templates in the cycle of the weekly access schedule, one of which is weekend (does not contain time intervals).*

## CreateWorktimeSchedule function

```
GuidResult CreateWorktimeSchedule( Guid sessionID, WorktimeSchedule  
schedule, ScheduleDay[] days )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>WorktimeSchedule schedule</code>	Schedule parameters.
<code>ScheduleDay[] days</code>	An array of templates for the days of the schedule cycle.

Result: It returns GuidResult.

Description: It creates a new working time schedule in the system with the specified parameters and the specified cycle. It returns the key of the newly created work schedule. When creating, you must follow the rules described in the section functions [CreateAccessSchedule](#).

## GetSchedule function

`Shedule GetSchedule( Guid sessionId, Guid scheduleID )`

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Guid scheduleID</code>	A unique schedule key.

Result: It returns `Schedule`; the result can be `AccessSchedule` or `WorktimeSchedule`.

Description: It returns information about the schedule with the specified key.

## SaveSchedule function

`BaseResult SaveSchedule( Guid sessionId, Schedule schedule )`

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Schedule schedule</code>	Schedule parameters.

Result: It returns `BaseResult`.

Description: It changes the parameters of the schedule. The 'schedule' parameter can be `AccessSchedule` or `WorktimeSchedule`.

When `AccessSchedule` or `WorktimeSchedule` structure is passing as schedule data in a request, it is mandatory to specify the class name in the *xsi: type* attribute of the *schedule* element.

## DeleteSchedule function

`BaseResult DeleteSchedule( Guid sessionId, Guid scheduleID )`

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Guid scheduleID</code>	The unique key of the schedule to be deleted.

Result: It returns `BaseResult`.

Description: It deletes the schedule with the specified key.

## GetScheduleDetails function

`ScheduleDay[] GetScheduleDetails( Guid sessionId, Guid scheduleID )`

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Guid scheduleID</code>	A unique schedule key.

Result: It returns an array `ScheduleDay`. The array can contain `ScheduleFix`.

Description: It returns information about all templates of days and correction days contained in the schedule with the specified key.

## SetScheduleDays function

```
BaseResult SetScheduleDays( Guid sessionID, Guid scheduleID,  
ScheduleDay[] days )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid scheduleID</code>	A unique schedule key.
<code>ScheduleDay[] days</code>	An array of templates for the days of the schedule cycle.

Result: It returns BaseResult.

Description: It adds a new cycle or modifies an existing one (if the cycle start date matches) of the schedule with the specified key. The 'days' parameter must follow the rules described in the [CreateAccessSchedule](#) section of the function.

## SetScheduleFix function

```
BaseResult SetScheduleFix( Guid sessionID, Guid scheduleID,  
ScheduleFix[] fixes )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid scheduleID</code>	A unique schedule key.
<code>ScheduleFix[] fixes</code>	Array of schedule correction days.

Result: It returns BaseResult.

Description: It adds correction days or modifies existing ones (if the date matches) of the schedule with the specified key. It cannot be applied to weekly access schedule.

## DeleteScheduleDays function

```
BaseResult DeleteScheduleDays( Guid sessionID, Guid scheduleID,  
DateTime[] days )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid scheduleID</code>	A unique schedule key.
<code>DateTime[] days</code>	Array of dates.

Result: It returns BaseResult.

Description: It removes the schedule cycle and correction days if the date in the 'days' array is the same as the cycle start date or correction date. It is inadmissible to delete the last cycle in the schedule.

## GetHolidays function

```
Holiday[] GetHolidays( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns the holiday array.

Description: It lists the holidays that exist in the system.

### **SetHolidays function**

```
BaseResult SetHolidays( Guid sessionID, Holiday[] holidays )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Holiday[] holidays</code>	An array of holidays.

Result: It returns BaseResult.

Description: It adds holidays or modifies existing ones (if dates match).

### **DeleteHolidays function**

```
BaseResult DeleteHolidays( Guid sessionID, Holiday[] holidays )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Holiday[] holidays</code>	Array of holidays to delete.

Result: It returns BaseResult.

Description: it deletes holidays with the specified parameters.

### **GetAccessGroups function**

```
AccessGroup[] GetAccessGroups( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns an AccessGroup array.

Description: It returns an array of available access groups.

### **CreateTempAccessGroup function**

```
GuidResult CreateTempAccessGroup( Guid sessionID, Guid scheduleID,  
Guid[] territories )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid scheduleID</code>	A unique schedule key.
<code>Guid[] territories</code>	An array of unique territory keys.

Result: It returns GuidResult.

Description: It creates a temporary access group according to a specified schedule for access to specified territories. The group is temporary and has a limited "lifetime". And the group is invalid without reference to employee ID.

Only an access group of the "Parsec access subsystem" type is available for creation.

## CreateAccessGroup function

```
GuidResult CreateAccessGroup( Guid sessionID, string groupName, Guid  
scheduleID, Guid[] territories )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>string groupName</code>	Access group name. Must be unique.
<code>Guid scheduleID</code>	A unique schedule key.
<code>Guid[] territories</code>	An array of territory keys.

Result: It returns GuidResult.

Description: It creates an access group on a specified schedule to access the selected territories. The access group will have the name specified in the function.

Only an access group of the "Parsec access subsystem" type is available for creation.

## DeleteAccessGroup function

```
BaseResult DeleteAccessGroup( Guid sessionID, Guid accessGroupID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid accessGroupID</code>	The unique access group key.

Result: It returns BaseResult.

Description: It removes an access group with the specified key.

Only an access group of the "Parsec access subsystem" type is available for deletion.

## AddSubAccessGroup function

```
GuidResult AddSubAccessGroup( Guid sessionID, Guid accessGroupID, Guid  
scheduleID, Guid[] territories )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid accessGroupID</code>	The unique access group key.
<code>Guid scheduleID</code>	The unique schedule key.
<code>Guid[] territories</code>	An array of territory keys.

Result: It returns GuidResult.

Description: It adds a group of components with a specified schedule for access to the selected territories to an existing access group.

If the specified territory is already used in any group of components (within the same access group), then it will be removed from this group of components.

Only an access group of the "Parsec access subsystem" type is available for modification.

## DeleteSubAccessGroup function

```
BaseResult DeleteSubAccessGroup( Guid sessionID, Guid accessGroupID, Guid subGroupID )
```

Options:

Guid sessionID	The unique session key.
Guid accessGroupID	The unique access group key.
Guid subGroupID	The unique component group key.

Result: It returns BaseResult.

Description: It removes a group of components with the specified key.

Only an access group of the "Parsec access subsystem" type is available for modification.

## GetSubAccessGroups function

```
SubAccessGroup[] GetSubAccessGroups( Guid sessionID, Guid accessGroupID )
```

Options:

Guid sessionID	The unique session key.
Guid accessGroupID	The unique access group key.

Result: It returns an array SubAccessGroup.

Description: It gives a list of component groups (subgroups) for the specified access group.

## GetInheritedAccessGroups function

```
Guid[] GetInheritedAccessGroups( Guid sessionID, Guid accessGroupID )
```

Options:

Guid sessionID	The unique session key.
Guid accessGroupID	The unique access group key.

Result: It returns an array of access group keys.

Description: It gives a list of access group keys nested for the specified access group. Nested group keys are returned in descending order of inheritance priority.

## SetInheritedAccessGroups function

```
BaseResult SetInheritedAccessGroup( Guid sessionID, Guid accessGroupID, Guid[] inheritedGroups )
```

Options:

Guid sessionID	The unique session key.
Guid accessGroupID	The unique access group key.
Guid[] inheritedGroups	An array of nested access group keys

Result: It returns BaseResult.



Description: It sets the nested access groups for the specified group. Inheritance priority is determined by the order of the elements in the array.

To cancel inheritance, you must pass an empty array.

Only an access group of the "Parsec access subsystem" type is available for modification.

## WORK WITH THE PASS OFFICE REQUESTS

### GetAcceptedVisitorRequests function

```
VisitorRequest[] GetAcceptedVisitorRequests( Guid sessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns an array of VisitorRequest.

Description: It gives a set of the pass office applications with the "Accepted" status.

### FindVisitorRequest function

```
VisitorRequest FindVisitorRequest( Guid sessionID, int requestNumber )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

<code>int requestNumber</code>	Application number.
--------------------------------	---------------------

Result: It returns VisitorRequest, or null if the request with the specified number was not found.

Description: It searches for a request with the specified number among requests with the "Accepted" status.

### ActivateVisitorRequest function

```
BaseResult ActivateVisitorRequest( Guid sessionID, Guid requestID, string cardCode )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

<code>Guid requestID</code>	The unique request key.
-----------------------------	-------------------------

<code>String cardCode</code>	ID code.
------------------------------	----------

Result: It returns BaseResult.

Description: It gives an identifier with the specified code from the pool to the visitor. This function is applicable only to requests with the "Accepted" status. Transfers the request to the "Active" status.

### CreateVisitorRequest function

```
VisitorRequest CreateVisitorRequest( Guid sessionID, VisitorRequest request )
```

Options:

<code>Guid sessionID</code>	The unique session key.
-----------------------------	-------------------------

<code>VisitorRequest</code> <code>request</code>	Request parameters.
---	---------------------

Result: It returns VisitorRequest.

Description: It creates a request for a visitor and returns information about it. The parameter must contain PERSON\_ID, ORGUNIT\_ID, ADMIT\_START, ADMIT\_END. All other parameters are optional. The request is created with the status "Issued".

### GetVisitorRequest function

```
VisitorRequest GetVisitorRequest( Guid sessionId, Guid requestID )
```

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Guid requestID</code>	The unique request key.

Result: It returns VisitorRequest.

Description: It returns information about the desired request.

### SaveVisitorRequest function

```
BaseResult SaveVisitorRequest( Guid sessionId, VisitorRequest request )
```

Options:

<code>Guid sessionId</code>	The unique session key.
<code>VisitorRequest request</code>	Request parameters.

Result: It returns BaseResult.

Description: Modifies parameters in an existing request.

### DeleteIssuedVisitorRequest Function

```
BaseResult DeleteIssuedVisitorRequest( Guid sessionId, Guid requestID )
```

Options:

<code>Guid sessionId</code>	The unique session key.
<code>Guid requestID</code>	The unique request key.

Result: It returns BaseResult.

Description: It deletes a visitor request. Only a request with the "Issued" status can be deleted.

### GetIssuedVisitorRequests function

```
VisitorRequest[] GetIssuedVisitorRequests ( Guid sessionId )
```

Options:

<code>Guid sessionId</code>	The unique session key.
-----------------------------	-------------------------

Result: It returns an array VisitorRequest.

Description: It returns an array of requests with the "Issued" status. If no orders are found, an empty array is returned.

## GetVisitorRequests function

```
VisitorRequest[] GetVisitorRequests( Guid sessionID, Guid orgUnitID,  
DateTime from, bool issued, bool accepted, bool declined, bool active,  
bool completed )
```

### Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid orgUnitID</code>	The unique department key.
<code>DateTime from</code>	The starting date for the selection of requests.
<code>bool issued</code>	Requests with the "Issued" status.
<code>bool accepted</code>	Requests with the "Accepted" status.
<code>bool declined</code>	Requests with the "Declined" status.
<code>bool active</code>	Requests with the "Active" status.
<code>bool completed</code>	Requests with the "Complete" status.

Result: Array VisitorRequest.

Description: It gives out all requests in the specified department selected by date and status. The function takes into account only the DATE part of the parameter 'from'.

## CloseAllActiveVisitorRequests function

```
BaseResult CloseAllActiveVisitorRequests( Guid sessionID, Guid  
visitorID )
```

### Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid visitorID</code>	The unique visitor key.

Result: It returns BaseResult.

Description: It searches for and closes all active applications associated with the specified visitor. The identifier is unlinked from the visitor and returned to the card pool of the pass office after closing the request.

## CloseVisitorRequest function

```
BaseResult CloseVisitorRequest( Guid sessionID, Guid requestID )
```

### Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid requestID</code>	The unique request key.

Result: It returns BaseResult.

Description: It closes the request with the specified code. The closing is performed only for active requests (the "Active" status). After the application is closed, the associated identifier is unlinked from the visitor and returned to the card pool of the pass office.

## GetPersonVisitorRequests function

```
VisitorRequest[] GetPersonVisitorRequests( Guid sessionID,  
Guid visitorID, bool issued, bool accepted, bool declined,  
bool active, bool completed )
```

Options:

Guid sessionID	The unique session key.
Guid visitorID	The unique visitor key.
bool issued	Requests with the "Issued" status.
bool accepted	Requests with the "Accepted" status.
bool declined	Requests with the "Declined" status.
bool active	Requests with the "Active" status.
bool completed	Requests with the "Completed" status.

Result: It returns an array VisitorRequest.

Description: It returns an array of all requests, available by scope, made for the specified visitor, with the ability to select / filter for all types of request statuses. If no request are found, an empty array is returned.

## SYSTEM EVENTS

### GetEvents function

```
EventsHistoryResult GetEvents( Guid sessionID, Guid TerritoryID, Guid  
PersNodeID, DateTime dtFrom, DateTime dtTo )
```

Options:

Guid sessionID	The unique session key.
Guid TerritoryID	The unique territory key.
Guid PersNodeID	The unique personnel key (department or employee).
DateTime dtFrom	Date from...
DateTime dtTo	Date to...

Result: It returns EventsHistoryResult.

Description: It returns the history of events of the specified territories and personnel for the specified time period. You can use a special value for the territory key and personnel key - Guid.Empty (00000000-0000-0000-0000-000000000000). When you specify it, the root of the corresponding hierarchy will be taken as a criterion.

***The GetEvents function has been deprecated. It is recommended to use the package of functions instead [OpenEventHistorySession](#) (see below).***

### OpenEventHistorySession function

```
GuidResult OpenEventHistorySession( Guid sessionID,  
EventHistoryQueryParams parameters )
```

Options:

Guid sessionID	The unique session key.
----------------	-------------------------

<code>EventHistoryQueryParams</code> parameters	Parameters of formation\filtering of events.
--	--

Result: It returns `GuidResult`.

Description: It opens an event report session. It returns a key for further receiving events.

### CloseEventHistorySession function

```
void CloseEventHistorySession( Guid sessionID, Guid  
eventHistorySessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid eventHistorySessionID</code>	The unique key of the event report session.

Result: –

Description: It closes the event report session.

### GetEventHistoryResultCount function

```
int GetEventHistoryResultCount( Guid sessionID, Guid  
eventHistorySessionID )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid eventHistorySessionID</code>	The unique key of the event report session.

Result: It returns `int`.

Description: It returns the number of events in an open session.

### GetEventHistoryResult function

```
EventObject[] GetEventHistoryResult( Guid sessionID, Guid  
eventHistorySessionID, Guid[] fields, int offset, int count )
```

Options:

<code>Guid sessionID</code>	The unique session key.
<code>Guid eventHistorySessionID</code>	The unique key of the event report session.
<code>Guid[] fields</code>	An array of unique field keys whose values should be returned.
<code>int offset</code>	The number of events from the beginning to be ignored in the result.
<code>int count</code>	The number of events that should in the result.

Result: It returns an array [EventObject](#).

Description: It returns an array of events ([EventObject](#)) with the values of the requested fields. The requested fields list consists of one or more keys from the table below. The list may contain additional field keys.

Guid	Description
71b03d7b-2e11-47cd-bf47-adaf320aeb10	Event date
c7ad4f51-d8af-4944-bf92-23714715147e	Event time
2c5ee108-28e3-4dcc-8c95-7f3222d8e67f	Date / time of event
633904b5-971b-4751-96a0-92dc03d5f616	Event source (name of territory or operator)
42DAB9C6-5D30-4030-8CCD-2CAD6FCBC5F2	Event source (array of territory identifiers)
d1847aff-11aa-4ef2-aaaa-795ceefe5f9f	Event type (name)
9F7A30E6-C9ED-4E62-83E3-59032A0F8D27	Event identifier (Guid)
C4AE9465-8375-4169-BA61-EB7E365A7352	Event type (hexadecimal code)
57CA38E4-ED6F-4D12-ADCB-2FAA16F950D7	Event type (10-digit code)
68ef9fd3-a72d-4520-9c63-5c37b0ae8539	Subject (full name) - from 'dictionary'
7C6D82A0-C8C8-495B-9728-357807193D23	Subject ID (PERS_ID - Guid)
66C5B505-C3A7-4227-AAD3-6B7BA3D8E612	Subject ID (PERS_ID - Guid) <b>in ParsecNET Office</b>
4c5807cb-2c06-4725-9243-747e40c41d6c	Region (name)
2ab38696-1e30-4e04-a956-b951cb7c2033	Part (name)
89c9d5ac-6e13-4715-a524-7c3b34931385	Work station
fea92e1c-e07d-4932-a6a1-e8c53e3087d9	Operator
03ceb65f-dcad-4b56-94b8-be9fdb463988	Details
e5ac823f-c4f6-48e7-bebe-e6d44c57c7ad	Summary
66aa3a39-c866-4f34-9e99-e75f9918eae7	Operator comments
99914915-c882-4d11-80ff-57acdc6cc015	Heading
2f4a647e-4d9e-48ad-bf11-b1e49ffeac7f	Message
1bf8a893-7d21-4c0c-9a2d-2e333a2d769d	Full name of the subject
0de358e0-c91b-4333-b902-000000000003	Surname / Licence plate
0de358e0-c91b-4333-b902-000000000001	Name / Vehicle model
0de358e0-c91b-4333-b902-000000000002	Middle name / Vehicle color
0de358e0-c91b-4333-b902-000000000006	Report card
0de358e0-c91b-4333-b902-000000000009	Organization
0de358e0-c91b-4333-b902-000000000004	Department
0a679144-d5ce-476d-a56e-0a696f079b71	Department description
0de358e0-c91b-4333-b902-00000000000a	Access group

Guid	Description
0de358e0-c91b-4333-b902-000000000005	Card code
0de358e0-c91b-4333-b902-000000000007	Action time from
0de358e0-c91b-4333-b902-000000000008	Action time to
3ad06d24-43f6-45e0-8164-a98b4da955dc	Photo
6FCFA1BB-9624-4248-A2D5-AA84901C53C8	Blacklisted subject
68D13785-C708-4418-8683-678A3F74957B	Images attached to the event

## GetHardwareEvents function

```
string[] GetHardwareEvents( Guid sessionId, int fromIndex )
```

Options:

<code>Guid sessionId</code>	The unique session key.
<code>int fromIndex</code>	Events starting from the specified index value inclusively.

Result: It returns a set of events.

Description: It returns a set of the specified number of operational events. The data is returned as a JSON document.

*An example of an 'Event' object obtained as a result of executing a function*

```
{
  "ID": "e299479e-596d-4be0-9688-950918621783",
  "Index": 1,
  "ParentID": "00000000-0000-0000-0000-000000000000",
  "Code": 590193,
  "Component": "770ca0ee-4ecb-4648-bd3f-02b47140a670",
  "Territories": [
    "ee7b6630-bcee-4137-b759-61c90889bf39"
  ],
  "Date": "2015-06-10T11:43:30Z",
  "Items": [
    {
      "Instance": 0,
      "Key": 83886592,
      "Type": 5,
      "Value": "0084C471"
    },
    {
      "Instance": 0,
      "Key": 16778496,
      "Type": 1,
      "Value": 83886081
    }
  ]
}
```

```

    },
    {
      "Instance": 0,
      "Key": 67110912,
      "Type": 4,
      "Value": "eec6409a-c092-46b7-9d8f-d311fc45c66d"
    },
    {
      "Instance": 0,
      "Key": 67110144,
      "Type": 4,
      "Value": "770ca0ee-4ecb-4648-bd3f-02b47140a670"
    },
    {
      "Instance": 0,
      "Key": 50334464,
      "Type": 3,
      "Value": 65600
    },
    {
      "Instance": 0,
      "Key": 67109376,
      "Type": 4,
      "Value": "039b2bf8-5a66-4b22-be4d-fb3d42af14d5"
    },
    {
      "Instance": 0,
      "Key": 201329152,
      "Type": 4,
      "Value": "a16486f7-8356-4fa9-8a4a-79e5659de052"
    }
  ]

```

*Event object structure*

*Table 1.*

Tag	Physical/Logical type	Description
ID	string / Guid	Event ID
Index	int / int	Event index. It is counted from the moment the operator logs in to the integration service.
ParentID	string / Guid	The identifier of the parent event that was the "initiator"
Code	int / int	Event type code
Component	string / Guid	Event source-component identifier



Territories	string[] / Guid[]	Array of territory identifiers referencing the event source-component
Date	string / DateTime	Date and time of the event
Items	ItemData[]	An array of service data of the <i>ItemData</i> type (see <i>Table 2.</i> )

*'ItemData' object structure*

*Table 2.*

Tag	Physical/Logical type	Description
Instance	int / int	Index of the data structure in the event data array. It grouped by the identity of the values of the "Key" and "Type" tags
Key	int / int	Service data. Its contain the data class of the "Value" tag in terms of the ParsecNET system. The data class is isolated by performing an operation $((Key \gg 8) \& 0xffff)$ . (see <i>Table 3.</i> )
Type	int / int	Logical data type of the tag "Value" in terms of the ParsecNET system (see <i>Table 4.</i> )

*Possible values of the data class in the "Key" tag of the 'ItemData' structure*

*Table 3.*

Value	Description
0	Undefined value
2	User
4	Command
5	Component
6	State
7	Detail
8	Device
9	Fast transport recipient
10	Fast transport sender
11	Event mask
12	Work station
13	Operator
14	ParsecNET system object
15	Scope (area of visibility)
16	Automation task
17	Event interpreter
18	Server
19	Date / Time
20	Image
21	Licence plate
22	Image area
23	Operator comments
24	Channel
25	Sound
26	Related event
27	Text message
28	Text message recipient address

Value	Description
0	Undefined value
1	DWORD. 4-byte integer
2	double. 8-byte floating point number
3	long. 8-byte integer
4	guid. 16-byte globally unique identifier
5	char[]. A string value with a maximum length of 16 characters
6	datetime. Date-Time
7	byte[]. Data array with a maximum array length of 16 bytes
8	guid. Large binary data link
9	struct {int, int, int, int}. Rectangular area descriptor structure

## GetHardwareEventsResolved function

`string[] GetHardwareEventsResolved( Guid sessionID, int fromIndex )`

Options:

<code>Guid sessionID</code>	The unique session key.
<code>int fromIndex</code>	Events starting from the specified index value inclusive.

Result: It returns a set of events.

Description: It returns a set of the specified number of operational events. The data is presented in a user-friendly way.

*An example of an EventResolved object obtained as a result of executing a function (for the same event as in the example for the previous function):*

```
{
  "ID": "e299479e-596d-4be0-9688-950918621783",
  "Index": 1,
  "ParentID": "00000000-0000-0000-0000-000000000000",
  "Code": {
    "Raw": 590193,
    "Resolved": "Нет входа - нет карты в БД"
  },
  "Classes": [
    {
      "Raw": 64,
      "Resolved": "Отказ в доступе"
    },
    {
      "Raw": 65536,
      "Resolved": "Отказ в доступе на вход"
    }
  ],
  "Component": {
```

```

    "Raw": "770ca0ee-4ecb-4648-bd3f-02b47140a670",
    "Resolved": "Шлагбаум въезд всех и выезд посетителей (NC 5K) "
  },
  "Territories": [
    {
      "Raw": "ee7b6630-bcee-4137-b759-61c90889bf39",
      "Resolved": "Шлагбаум въезд всех и выезд посетителей (NC 5K) "
    }
  ],
  "Date": "2015-06-10T11:43:30Z",
  "CardCode": "0084C471",
  "User": {
    "Raw": "039b2bf8-5a66-4b22-be4d-fb3d42af14d5",
    "Resolved": "Иванов Иван Иванович"
  },
  "Operator": null,
  "Workstation": null,
  "Part": null,
  "Details": null
}

```

#### *EventResolved Object Structure*

Tag	Physical/Logical type	Description
ID	string / Guid	Event ID
Index	int / int	Event index. It is counted from the moment the operator logs in to the integration service.
ParentID	string / Guid	The identifier of the parent event that was the "initiator"
Code	ResolvedValue	An instance of an object of ResolvedValue type describing the type of event
Classes	ResolvedValue []	An array of an object of ResolvedValue type describing the categories to which the event belongs
Component	ResolvedValue	An instance of an object of ResolvedValue type describing the event source component
Territories	ResolvedValue []	An array of an object of ResolvedValue type describing the territories that reference the event source component
Date	string / DateTime	Date and time of the event
CardCode	string / string	ParsecNET system user card code (if applicable to a specific type of event)
User	ResolvedValue	An instance of an object of type ResolvedValue describing the user of the ParsecNET system (if applicable to a specific type of event)
Operator	ResolvedValue	An instance of an object of type ResolvedValue describing the operator of the ParsecNET system (if applicable to a specific type of event)

Workstation	ResolvedValue	An instance of an object of ResolvedValue type describing the workstation of the ParsecNET system (if applicable to a specific type of event)
Part	ResolvedValue	An instance of an object of ResolvedValue type describing a detail of the ParsecNET system (if applicable to a specific type of event)
Details	ResolvedValue	An instance of an object of ResolvedValue type describing the data of the ParsecNET system object associated with a specific event (if applicable to a specific type of event)

#### *ResolvedValue structure*

Tag	Physical/Logical type	Description
Raw	Any / Any	Data suitable for computer processing
Resolved	string / string	Human-readable data

### GetTransactionClasses function

```
TransactionClass[] GetTransactionClasses( Guid sessionID )
```

Options:

`Guid sessionID`                      The unique session key.

Result: It returns a set of event categories.

Description: It returns a set of all categories of events occurring in the system. An event in ParsecNET system can belong to one or more categories at the same time. Each category in the returned set corresponds to 1 bit of the bitmask that is assigned to events.

### GetTransactionTypes function

```
TransactionType[] GetTransactionTypes( Guid sessionID, long classMask )
```

Options:

`Guid sessionID`                      The unique session key.

`long classMask`                      Bitmask specifying categories of events. If a certain bit is set to 1, then events belonging to this category will be returned.

Result: It returns a set of event types.

Description: It returns a set of events using a category bitmask.

## **4. HISTORY OF CHANGES**

### **Version 3.11.629.15 (July 2021)**

Added functions: GetUnique4bCardCode, GetCardCodeFromUID, GenerateParsecQRCode.

### **Version 3.11.127.25 (March 2021)**

Added functions: GetObjectName, SendVerificationCommand, GetMultiplePersons, PersonSearch

Roles used in the CheckRole function have been added.

## 5. ALPHABETIC INDEX

AccessGroup .....	21	DeletePassgeRole .....	49
AccessSchedule .....	16	DeletePerson .....	43
ActivateVisitorRequest.....	57	DeletePersonScheduleFix .....	39
AddPersonIdentifier.....	45	DeleteSchedule .....	52
AddPersonScheduleFix.....	38	DeleteScheduleDays .....	53
AddSubAccessGroup .....	55	DeleteSubAccessGroup .....	56
BaseIdentifier.....	18	Domain .....	22
BaseObject .....	13	Event.....	22
BaseOrgUnit.....	13	EventHistoryQueryParams .....	22
BasePerson .....	14	EventObject .....	22
BaseResult.....	12	EventsHistory .....	22
BaseTerritory .....	20	EventsHistoryResult.....	13
BlockPerson .....	43	ExtraFieldValue .....	15
ChangePersonIdentifier .....	46	FindPeople .....	34
CheckRole.....	26	FindPersonByIdentifier .....	35
CloseAllActiveVisitorRequests .....	59	FindVehicle.....	34
CloseEventHistorySession .....	61	FindVisitorRequest .....	57
CloseOrgUnitEditingSession.....	32	FindVisitors .....	34
ClosePersonEditingSession .....	41	GenerateParsecQRCode .....	50
CloseSession .....	26	GetAcceptedVisitorRequests .....	57
CloseVisitorRequest .....	59	GetAccessGroups .....	54
ContinueSession .....	26	GetAccessSchedules .....	50
CreateAccessGroup .....	55	GetCardCodeFromUID.....	49
CreateAccessSchedule.....	51	GetDomains .....	25
CreateOrgUnit .....	32	GetEventHistoryResult .....	61
CreatePassageRole.....	48	GetEventHistoryResultCount.....	61
CreatePerson .....	40	GetEvents .....	60
CreateTempAccessGroup .....	54	GetHardwareEvents.....	63
CreateVehicle .....	40	GetHardwareEventsResolved .....	66
CreateVisitor .....	40	GetHardwareState.....	28
CreateVisitorRequest.....	57	GetHolidays.....	53
CreateWorktimeSchedule.....	51	GetIdentifierExtraData .....	47
DeleteAccessGroup .....	55	GetInheritedAccessGroups .....	56
DeleteHolidays .....	54	GetIssuedVisitorRequests.....	58
DeleteIdentifier .....	45	GetMultiplePersons .....	36
DeleteIssuedVisitorRequest.....	58	GetObjectName .....	27
DeleteOrgUnit .....	33	GetOrgUnit.....	32

GetOrgUnitsHierarhy .....	30	GuidResult .....	12
GetOrgUnitsHierarhyWithPersons ....	30	HardwareState.....	24
GetOrgUnitsHierarhyWithVehicle.....	30	Holiday .....	18
GetOrgUnitsHierarhyWithVisitors ....	30	Identifier.....	18
GetOrgUnitSubItems .....	31	IdentifierExData .....	19
GetOrgUnitSubItemsHierarhyWithPers ons .....	31	IdentifierTemp .....	19
GetOrgUnitSubItemsHierarhyWithVehi cle .....	31	ObjectResult.....	13
GetOrgUnitSubItemsHierarhyWithVisit ors.....	31	OpenEventHistorySession.....	60
GetPassageRoles .....	48	OpenOrgUnitEditingSession .....	32
GetPerson.....	36	OpenPersonEditingSession .....	40
GetPersonExtraFieldTemplates .....	33	OpenSession .....	25
GetPersonExtraFieldValue .....	37	OpenSessionWithInLocale .....	25
GetPersonExtraFieldValues.....	37	OrgUnit.....	14
GetPersonExtraFieldValueString.....	37	PassageRole .....	20
GetPersonIdentifiers.....	44	Person .....	14
GetPersonsChangedAfter .....	37	PersonExtraFieldTemplate .....	15
GetPersonScheduleFixes .....	38	PersonScheduleFix .....	15
GetPersonVisitorRequests .....	60	PersonSearch .....	35
GetPersonWorktimeSchedule.....	39	PersonWithPhoto.....	14
GetRootOrgUnit .....	30	SaveOrgUnit.....	33
GetRootTerritory.....	44	SavePassageRole .....	48
GetSchedule .....	52	SavePerson .....	41
GetScheduleDetails .....	52	SavePersonScheduleFix.....	39
GetScheduleIntervals .....	50	SaveSchedule .....	52
GetSubAccessGroups .....	56	SaveVisitorRequest .....	58
GetTerritoriesHierarhy .....	44	Schedule.....	16
GetTerritory .....	44	ScheduleDay .....	17
GetTerritorySubItems.....	44	ScheduleFix .....	17
GetTransactionClasses.....	68	SendHardwareCommand.....	27
GetTransactionTypes.....	68	SendVerificationCommand.....	28
GetUnique4bCardCode .....	49	Session .....	13
GetVehicleExtraFieldTemplates .....	34	SessionResult .....	12
GetVersion.....	25	SetHolidays .....	54
GetVisitorExtraFieldTemplates .....	33	SetIdentifierExtraData .....	48
GetVisitorRequest .....	58	SetIdentifierPrivileges.....	47
GetVisitorRequests.....	59	SetInheritedAccessGroups.....	56
GetWorktimeSchedules.....	50	SetPersonExtraFieldValue .....	42
		SetPersonExtraFieldValues .....	42
		SetPersonOrgUnit.....	41

SetPersonPhoto .....	41	TimeInterval.....	18
SetPersonWorktimeSchedule .....	39	TransactionClass .....	24
SetScheduleDays.....	53	TransactionType.....	24
SetScheduleFix.....	53	UnblockPerson .....	43
StockIdentifier.....	19	ValidateExtraFieldValue.....	38
StringResult .....	12	VisitorRequest .....	15
SubAccessGroup.....	21	WorktimeInterval .....	18
Territory .....	20	WorktimeSchedule .....	17
TerritoryWithComponent.....	20		