Web-services_old

Content

Web-services

Endpoints

PCS Integration Platform functions should be invoked by national systems as web services. The WSDL (Web Service Definition Language) – based list of functions is shown on the test and production system of PCS Integration Platform.

There are four major (non-backward compatible) PCS IP versions deployed:

- legacy 1.x version (supported by 1.x endpoints). Obsolete (needed for non TAF-TSI compliant dossiers only)
- legacy 2.x version (supported by 2.x endpoint). Obsolete (v4 replaces it).
- legacy 3.x version (supported by 3.x endpoint). Obsolete (v4 replaces it).
- 4.x version (supported by 4.x endpoint). Obsolete (v5 replaces it).
- 5.x version (supported by 5.x endpoint). This is the current version which should be used by integrators.

Valid version

In order to utilize the latest PCS features (Main/Subsidiary, handling of PaPs in the dossiers, Composite Relations), PCS IP v5 version should be used for new (or upgrade of existing) integration efforts. The dossiers which support all new features should belong to a certain combination of process type and timetable period (see table below).

Process type	Timetable Period
New	2019
Late	2019
Ad-Hoc	2018

The test system can be reached here (WSDL):

2.x version - https://pfiptest.railneteurope.info/pathfinderintegration/webservices/IProxyIntegration_v2?wsdl

3.x version - https://pfiptest.railneteurope.info/pathfinderintegration/webservices/IProxyIntegration_v3?wsdl

4.x version - https://pfiptest.railneteurope.info/pathfinderintegration/webservices/IProxyIntegration_v4?wsdl

5.x version - https://pcstest1.rne.eu/pcs/services/IProxyIntegrationService_v5_1?wsdl

The production system can be reached here (WSDL):

2.x version - https://ippf.railneteurope.info/pathfinderintegration/webservices/IProxyIntegration_v2?wsdl

3.x version - https://ippf.railneteurope.info/pathfinderintegration/webservices/IProxyIntegration_v3?wsdl

4.x version - https://ippf.railneteurope.info/pathfinderintegration/webservices/IProxyIntegration_v4?wsdl

5.x version - https://pcs-online.rne.eu/pcs/services/IProxyIntegrationService_v5_1?wsdl

There is a separated Web Service endpoint which corresponds to WS-Security standards – it can be found in the reference list in [10].

Authentication mechanism

In order to execute certain operations exposed through the web services, the user needs to be authenticated.

The exposed web services which needs authentication include:

- createDossier
- getDossier
- updateDossier (on RU-IM Pair basis)
- helper functions (getOperationPointByName etc.)

PCS Core already provides an authentication system which can be reused to achieve the desired functionality:

- Agency and User management still to be done with PCS Core
- Secure and robust authentication mechanism

The desired functionality is provided through the following operation:

- **authenticate**

Authenticate operation

The user includes his credentials (username, password).

The system checks user credentials. Based on the outcome of these operations, it returns the appropriate status code (success or error). On successful authentication, system responds with the value of JSESSIONID given in the tag <authenticateData>.

If the operation resulted in success, the user is logged in and can execute the protected operations. For every subsequent protected operation, the JSESSIONID has to be written in the <authenticateData><sessionid>.

Session timeout applies in the same way as in the normal HTTP/HTML system.

createDossier operation

XML Schema

XML Schema definition is available in [3].

Validation

The web service implementation relies on validation on two levels:

- XML schema validation - validates the SOAP request against the schema and throws SOAPFault with corresponding description if they don't match.
- Application-level validation - consistency check (e.g. matches <agency_id> against the database) and return response with status that indicate error code.

General remark: please look into the XSD description [9] to see the functional, application-side validation rules for the XML elements of the request.

Request

In order to make a valid SOAP request (based on the XML Schema), Integration Platform should have knowledge about the following PCS Core specifics:

- Agencies
  - Agency id used in <dossier_agency> element
  - All agencies (all companies that have an access to PCS) can be retrieved with the operation**getAllAgencies**
- Process and Dossier Types
  - The list of supported process types can be retrieved through

    getSupportedDossierProcessTypesMatrix operation.

    If the process type is not provided, the PCS Core system generates the appropriate type depending on the time of the year. Please look at the PCS Process types documentation [8] for more details about process types. Usually, the createDossier operation is executed by Railway Undertakings (RU-user identification) – the only case where Infrastructure Manager (IM-user identification) can execute this operation is creating of a Catalogue Path. Hence, the system will automatically set the process type to Catalogue, if the createDossier is executed with user credentials of an IM-user.
- Progress statuses of dossier agencies
  - The progress status in PCS is the acceptance indicator ("traffic lights" in the control panel of the PCS Core System) of a phase in the dossier. At the opening (dossier creation), only statuses "Being processed" (value 'P' in the request) and "Not yet processed" (value 'U' in the request) are supported.
- International Train Nr and Dossier Title
  - These two parameters are mandatory for dossier creation since PCS Core System provides dossier details only for the dossiers with international train number and dossier title.
- Leading RU Agency ID

- This is new since the PCS Core System version 2.1.2: leading RU ID has to be provided for all process and dossier types except “Catalog Path”.
- **NEW (Version 2.2.2)**: From version 2.2.2 leading_ru_uic_id can be provided in the dossier instead of leading_ru_id which is PCS-specific agency-id (please check the table below).

  Additionally, if the leading RU is not provided the system will set the agency of the user whose credentials were used for identification as the leading RU (in the case of Catalogue, IM).

- Brake Types
  - Brake type id used in <braketype_id> element (child of <path_section>)
  - All supported brake types can be retrieved by using of the operation**getAllBrakeTypes**
- Operation points
  - Operation point id used in <from_op_id> element (child of <path_section>)
  - Operation points can be retrieved upon their names with the operation getOperationPointByName and upon PCS-specific ID by getOperationPointById. Please note that you can use just the first letters of the location name for the retrieval upon name – the system will respond with the list of operation points. Please use only ASCII-7 character set, do not put any umlaut characters.
  - **NEW (Version 2.2.2)**: From version 2.2.2 operation points do not need to be initially set in the dossier with PCS-specific operation_point_id, the national reference_id with agency_uic_id of the Infrastructure Manager whom the station is belonging to is sufficient (please check the table below)!
- Path sections
  - Timetable is given under the <tt_ru> element for Railway Undertakings (PCS term: “Requested RU Timetable”) in the dossier. The path element in tt_ru element has to contain at least two path_section elements. The system responds with an error if no requested timetable or empty timetable is sent in the request for all process types except “Catalogue”.
- Calendar items
  - The calendar (bitmap stream) has to be indicated in the path request on the reference location. The calendar item in a dossier is identified with PCS specific code calendar_item_id. A calendar item id is a PCS specific identifier which represents only one bitfield. It basically represents a shortcut for the bitfield. PCS IP umbrella element <pathfinderintegration> heavily relies on the usage of calendar item id. The calendar item id for a particular bitfield can be retrieved via **getCalendarItemId** operation and used throughout other PCS IP invocations like in this case, within the operation **createDossier**.
- National IM parameters
  - For every path section belonging to RU user, validation for national im parameters is done. The algorithm looks like this:
    - Find corresponding IM agency for the path section (first use sort order of the involved agencies, then check for RU’s preferred IM configuration)
    - If IM agency is found, get all defined im parameters and perform validation
    - Validation is consisted of :
      - Making sure the mandatory (if any) parameters are present in the request
      - Making sure the parameters of list type (if any) in the request contain valid value(s).

  You can retrieve the national parameters with the operation getImParameters.

IMPORTANT: When using createDossier operation you should not provide the following IDs in xml payload, since they will be generated by the system automatically:

- Id for the dossier element
- Id for the path element
- Id for the path_section element

Generally, the content of a request for dossier creation via Web Service is the same as the content for dossier creation in user interface in the web application PCS Core. For all elements whose ID is needed in the request for dossier creation there is an appropriate helper Web Service operation which can be called in order to get the accurate data for the request construction (see subsequent read-only operations). All data items offered in the user interface (dossier types, status, agencies etc.) of PCS Core are callable via Web Service.

NEW SINCE VERSION 2.2.2: since the version 2.2.2 it is possible to use uic-code of the particular agency instead of agency_id and reference_id of the stations that are provided from national systems instead of operation_point_id. The system checks the input, recognizes automatically the existing operation points due to the reference_id, name and agency_uic_id – for those that are not existing in PCS, the system adds the new operation points to the database.

NEW SINCE VERSION 2.2.3: Since PFCore 2.2.3, there is a support for creation of Short Term Path Request for RUs. NOTE: the process type must be specified.

Parent element	**Existing element**	**Alternative element**
DossierData element (child of <dossier>)	<dossierdata> <...other elements...> <leading_ru_id /> <leading_im_id /> <...other elements...> </dossierdata>	<dossierdata> <...other elements...> <leading_ru_uic_id /> <leading_im_uic_id /> <...other elements...> </dossierdata>
DossierAgency element (child of <dossier>/<involved_agencies>)	<dossier_agency agency_id="…" >	<dossier_agency agency_uic_id="…" >

TrasseElement element (child of <tt_ru>/<path> and <tt_km>/<path>)	<path_section> <...other elements...> <from_op_id/> <...other elements...> </path_section>	<path_section> <...other elements...> <operationpointLocal> <name/> <reference_id/> <agency_uic_id/> </operationpointLocal> <...other elements...> </path_section>
TrasseElement element (child of <tt_ru>/<path> and <tt_km>/<path>)	<path_section> <...other elements...> <agency_id/> <...other elements...> </path_section>	<path_section> <...other elements...> <agency_uic_id/> <...other elements...> </path_section>

	<path_section>	<path_section>
	<...other elements...>	<...other elements...>
TrasseElement element	<calendaritem_id />	< trafficperiod_id/>
(child of <tt_ru>/<path> and <tt_km>/<path>)	<...other elements...>	<...other elements...>
	</path_section>	</path_section>
	<compositionitem>	<compositionitem>
	<...other elements...>	<...other elements...>
CompositionItem element	<agency_id/>	<agency_uic_id/>
(<traincomposition>/items/compositionitem)	<...other elements...>	<...other elements...>
	</compositionitem>	</compositionitem>
	<compositionitem_cargo>	<compositionitem_cargo>
	<...other elements...>	<...other elements...>
CompositionItemCargo element	<agency_id/>	<agency_uic_id/>
(<traincomposition>/items/compositionitem_cargo)	<...other elements...>	<...other elements...>
	</compositionitem_cargo>	</compositionitem_cargo>

		<compositionitem_cargo>
		<...other elements...>
	<compositionitem_cargo>	<section_local_from>
	<...other elements...>	<name/>
CompositionItemCargo element	<abschnitt_from/>	<reference_id/>
(<traincomposition>/items/compositionitem_cargo)	<...other elements...>	<agency_uic_id/>
	</compositionitem_cargo>	<section_local_from/>
		<...other elements...>
		</compositionitem_cargo>
		<compositionitem_cargo>
		<...other elements...>
	<compositionitem_cargo>	<section_local_to>
	<...other elements...>	<name/>
CompositionItemCargo element	<abschnitt_to/>	<reference_id/>
(<traincomposition>/items/compositionitem_cargo)	<...other elements...>	<agency_uic_id/>
	</compositionitem_cargo>	<section_local_to/>
		<...other elements...>
		</compositionitem_cargo>

CompositionItemCargo element

(<traincomposition>/items/compositionitem_cargo)

```
<compositionitem_cargo>
 <...other elements...>
 <comp_group_from/>
 <...other elements...>
</compositionitem_cargo>
```

```
<compositionitem_cargo>
 <...other elements...>
 <comp_group_local_from>
  <name/>
  <reference_id/>
  <agency_uic_id/>
 <comp_group_local_from/>
 <...other elements...>
</compositionitem_cargo>
```

CompositionItemCargo element

(<traincomposition>/items/compositionitem_cargo)

```
<compositionitem_cargo>
 <...other elements...>
 <comp_group_to/>
 <...other elements...>
</compositionitem_cargo>
```

```
<compositionitem_cargo>
 <...other elements...>
 <comp_group_to_from>
  <name/>
  <reference_id/>
  <agency_uic_id/>
 <comp_group_to_from/>
 <...other elements...>
</compositionitem_cargo>
```

CompositionItemCargo element

(<traincomposition>/items/compositionitem_cargo)

```
<compositionitem_cargo>
 <...other elements...>
 <vertrauensuebergang_from/>
 <...other elements...>
</compositionitem_cargo>
```

```
<compositionitem_cargo>
 <...other elements...>
 <changeover_local_from>
  <name/>
 <reference_id/>
 <agency_uic_id/>
 <changeover_local_from/>
 <...other elements...>
</compositionitem_cargo>
```

CompositionItemCargo element

(<traincomposition>/items/compositionitem_cargo)

```
<compositionitem_cargo>
 <...other elements...>
 <vertrauensuebergang_to/>
 <...other elements...>
</compositionitem_cargo>
```

```
<compositionitem_cargo>
 <...other elements...>
 <changeover_local_to>
  <name/>
 <reference_id/>
 <agency_uic_id/>
 <changeover_local_to/>
 <...other elements...>
</compositionitem_cargo>
```

CompositionFootnote element (<traincomposition>/footnotes/compositionfootnote)	<compositionfootnote> <...other elements...> <agency_id/> <...other elements...> </compositionfootnote>	<compositionfootnote> <...other elements...> <agency_uic_id/> <...other elements...> </compositionfootnote>
Through coach element (<dossier>/throughCoaches)	<throughCoach agency_id="…" >	<throughCoach agency_uic_id="…" >

NEW SINCE VERSION 4.0: **Check changes in Main Calendar Handling**

Response

createDossier operation returns a response consisted of the following:

- **dossierId** – if the operation runs successfully, the returned value is the dossier id that has been assigned from the PCS Core system. Otherwise it has INVALID VALUE (-1).
- status – description for the response. The list of error codes can be found in the**Appendix A** at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

Important: Since 2.2.1 version, dossier with Feasibility Study process type cannot be created.

getDossier operation

General

You should use this operation to get the full dossier xml payload in order to prepare the dossier for update.

Validation

The web service implementation relies on XML schema validation - validates the SOAP request against the schema and throws SOAPFault with corresponding description if they don't match.

Request

getDossier request is consisted of two elements:

- dossierId – mandatory parameter
- version – optional parameter (if not specified, the latest version number is used)

Response

getDossier operation returns a response consisted of the following:

- dossier – if the operation runs successfully, the returned value is the dossier data. It is expected that the returned data will be extended to adhere to updateDossier implementation.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

NEW SINCE VERSION 4.0: **Check changes in Main Calendar Handling.**

NEW SINCE VERSION 5.2:

1. In TrasseElement a new optional element **<responsible_ru_agency_id>** or **<responsible_ru_agency_uic_id>** is added, before <agency_id> or <agency_uic_id> element. It is the id of the RU agency responsible for the given path section. This field is not supported in the notification messages, only in getDossier operation.
2. The request type for the getDossier operation, **GetDossierRequest,** has new optional boolean parameter **pathsWithResponsibleRu**. When set to true it indicates that **<responsible_ru_agency_id>** element should be included in all elements of type TrasseElement for the dossier in the response.

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

updateDossierRUIMPair operation

General

updateDossierRUIMPair operation is the most frequently used operation of the Integration Platform.

updateDossierRUIMPair is used for the following actions:

- acceptance indicators change ("traffic light" change, path and production related)
- state transitions (switching from one phase to another in the dossier)

- update of dossier data (changes in timetable, train parameters, train composition)

The request can be any combination of the above scenarios. Appropriate access rights checks are triggered based on your authentication.

IMPORTANT: updateDossierRUIMPair operation has replaced the updateDossier web service operation which was used until PCS IP 2.0. In this variant, user works with the current RU-IM pair (input parameter for the operation) instead of the whole dossier. The following data is now updated per pair (instead of per dossier):

- dossierstatus_id
- progressstatus_id (for both RUs and IMs)
- commercialstatus_id (for RUs only)

The PCS system takes care about the following:

Only the data of “your” agency is updated, the data of other agencies is protected. Hence, if you send the dossier xml payload containing out-of-date information of other agencies and your newest changes, the system PCS will filter out your changes, get the latest version of the dossier from the core system and merge your data with the newest data from the core system. With this approach it is ensured that your dossier update is always consistent with the changes that are made by other agencies.

IMPORTANT: when using updateDossierRUIMPair operation, you always have to be sure that you use the proper id-s for the elements like ruim_pair_id, dossier, path, path_section etc.

Only if you want to add a path or path_section you can use an element without id – in that case, the system will generate the id accordingly. By the next execution of the getDossier operation for the particular dossier, you will have the corresponding ids in the dossier that is delivered in the response.

NEW SINCE VERSION 4.0: **Check changes in Main Calendar Handling.**

NEW SINCE VERSION 5.2:

The request type for the updateDossierRUIMPair operation, **UpdateDossierRUIMPairRequest,** has new optional boolean parameter **updateImPathBasedOnResponsibleRU**. By setting it to true you can indicate that the update of IM paths should be based on the responsible RU agency. This will work only if responsible RU agency is set in all elements from type TrasseElement for the dossier in the request (element **<responsible_ru_agency_id>** or **<responsible_ru_agency_uic_id>**) This element is needed to ensure that the sent block (the one that needs to be updated) is resolved correctly on both sides, the client and PCS IP. All path sections that do not have the same responsible RU agency as the one from the RU-IM pair in the request won’t be touched.

Examples:

```
 <xsd:element name="updateDossierRUIMPairRequest">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="v5_0:pathfinderintegration"/>
            <xsd:element ref="v3_0:dossierRUIMPair" minOccurs="1"/>
            <xsd:choice minOccurs="0">
                <xsd:element ref="tins:PathRequestMessage" minOccurs="0"/>
                <xsd:element ref="tins:PathDetailsMessage" minOccurs="0"/>
                <xsd:element ref="tins:PathNotAvailableMessage" minOccurs="0"/>
                <xsd:element ref="tins:PathConfirmedMessage" minOccurs="0"/>
                <xsd:element ref="tins:PathDetailsRefusedMessage" minOccurs="0"/>
                <xsd:element ref="tins:PathCanceledMessage" minOccurs="0"/>
                <xsd:element ref="tins:ReceiptConfirmationMessage" minOccurs="0"/>
                <xsd:element ref="tins:ObjectInfoMessage" minOccurs="0"/>
                <xsd:element ref="tins:PathCoordinationMessage" minOccurs="0"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute name="updateImPathBasedOnResponsibleRU" type="xsd:boolean" use="optional">
            <xsd:annotation>
                <xsd:documentation>Optional attribute that indicates that the update of an IM path should be based
                    on responsible RU i.e. only blocks with the responsible RU belonging to the given pair will be
                    updated, the rest of the path won't be modified.
                </xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
    </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="responsible_ru_agency_id" minOccurs="0">
    <xsd:annotation>
        <xsd:documentation>Responsible RU agency for the path section. List of valid entries can be
            retrieved through getAllAgencies operation.</xsd:documentation>
    </xsd:annotation>
    <xsd:simpleType>
        <xsd:restriction base="xsd:long">
            <xsd:totalDigits value="12" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:choice>
    <xsd:element name="agency_id" minOccurs="0">
        <xsd:annotation>
            <xsd:documentation>Responsible agency for the path section. List of valid entries can be
                retrieved through getAllAgencies operation.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleType>
            <xsd:restriction base="xsd:long">
                <xsd:totalDigits value="12" />
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:element>
```

Use cases:

1. Completely replacing a subsidiary that is owned by User
    a. updateImPathBasedOnResponsibleRU = false
    b. No need to send responsible_ru_agency_id elements
    c. User doesn't have to send path or path_section id attribute
2. Update the master or a sub only for one RU
    a. updateImPathBasedOnResponsibleRU = true
    b. User must send responsible_ru_agency_id elements
    c. User must send path id
    d. User don't have to send path_section id's
    e. User needs to send only blocks related to the RU user wants to update, everything else will be reconstructed by PCS
    f. Have in mind user needs to have the reference point in that block, if user wants to change the calendar
3. Removing an RU block from master or sub
    a. updateImPathBasedOnResponsibleRU = false
    b. User must send path id
    c. User don't have to send path or path_section id's

   d. Send only the blocks of the RU that should remain
   e. Make sure user still owns a reference point
   f. Though discouraged, user can use this option to achieve subsidiaries per pair
4. Updating subs managed by PCS or master that has PaP's
   a. Must contain path id
   b. Must contain path section id's and all catalogue related fields for path sections coming from PaP
   c. Must contain all blocks for all involved RU's

XML Schema

XML Schema definition is available in [3].

In order to support updating of dossier attachments via MTOM [2], XML Schema is extended with a xsd:base64Binary element which represents the attached file.

Validation

updateDossier validation is consisted of:

- XML schema validation - validates the SOAP request against the schema and throws SOAPFault with corresponding description if they don't match.
- Application-level validation - consistency check regarding to the existing access rights in PCS Core system [1] as well as comparing request ids with corresponding ones from the database and national IM parameters validation. In case of error, the response should contain error code which indicates the problem.
- Changed since PCS Core 2.1.2: Leading RU is mandatory field

Request

PCS Core system always expects the entire dossier xml in the request. In case of dossier attachment update, the file should be added as attachment in the SOAP message.

MTOM needs also to be enabled on the client side. Please see soapUI testing framework configuration for an example.

Response

updateDossierRUIMPair operation returns a response consisted of the following:

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

getAllAgencies operation

General

getAllAgencies is a helper operation which exposes PCS-specific information for the defined agencies.

Response

getAllAgencies operation returns a response consisted of the following:

- agencies – if the operation runs successfully, list of agencies with the following structure is exposed: agency id, name, shortname, agencytype_id, agencysubtype_id, agencygroup_id.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAllBrakeTypes operation

General

getAllBrakeTypes is a helper operation which exposes brake types defined in PCS.

Response

getAllBrakeTypes operation returns a response consisted of the following:

- brake types – if the operation runs successfully, list of brake types with the following structure is exposed: braketype_id and braketype_shortname.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAllProgressStatuses operation

General

getAllProgressStatuses is a helper operation which exposes valid progress statuses for agencies in PCS..

Response

getAllProgressStatuses operation returns a response consisted of the following:

- progress statuses – if the operation runs successfully, list of progress statuses with the following structure is exposed: progressstatus_id and progressstatus_description in the preferred language of the user.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getSupportedDossierTypes operation (deprecated!)

General

getSupportedDossierTypes is deprecated since PCS Core version 2.1.2, at the moment it returnes the same results as getSupportedDossierProcessTypesMatrix operation – the latest operation has to be used instead..

IMPORTANT: decommissioned in PFIP 2.0, use getSupportedDossierProcessTypesMatrix instead.

getAllDossiers operation

General

getAllDossiers operation returns dossiers for the search criteria: dossierstatus_id, dossiertype_id and ttp_id.

Request

getAllDossiers request is consisted of the following information:

- dossierstatus_id
  - The list of valid dossier status ids looks like:

    - D (Open)
    - C (Harmonization)
    - M (Path Study Request)
    - N (Path Study Elaboration)
    - I (Path Study Response)
    - E (Path Request)
    - F (Conference)
    - T (Path Elaboration)

- O (Draft Timetable)
- V (Verification, legacy status, not used anymore)
- X (Observations)
- J (Post-processing)
- A (Final offer)
- P (Active Timetable)
- R (Closed)
- B (Open CT)
- S (Path Elaboration CT)
- K (Catalog CT)
- G (Closed CT)

- dossiertype_id
  - Supported dossier types for the web services can be retrieved using getSupportedDossierTypes operation
- ttp_id – timetable period (e.g. 2010)

Response

getAllDossiers operation returns a response consisted of the following:

- dossiers – if the operation runs successfully, list of dossiers with the following structure is exposed: dossier_id, changenr(version), dossierstatus_id, dossiertype_ids, ttp_id. The list can be empty if no dossier match the search criteria.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getUserAgency operation

General

getUserAgency is a helper operation which retrieves agency information for the logged in user.

Response

getUserAgency operation returns a response consisted of the following:

- agency – if the operation runs successfully, the following information for the logged in user's agency is retrieved: name, shortname, agencytype_id, agencysubtype_id, agencygroup_id.

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getOperationPointByName operation

General

getOperationPointByName is a helper operation which retrieves operation points based on search criteria. It is basically a web service version of an already present operation point search in PCS Core.

Request

Search criteria is consisted of only one field: search_value which is used for search by operation point name. Wildcards can also be used in the search_value.

Response

getOperationPointByName operation returns a response consisted of the following:

- operationpoints – if the operation runs successfully, a list of operation points with the following structure is retrieved: id, name, enee_id (if the operation point has a valid enee_id). The list is empty if there's no match for the search criteria.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getIMParameters operation (deprecated!)

General

getImParameters is deprecated since PCS Core version 2.2.4, at the moment it returns the same results as getImParametersForTimetablePeriod operation – the latest operation has to be used instead..

IMPORTANT: decommissioned in PFIP 2.0, use getImParametersForTimetablePeriod instead.

Request

Search criteria is consisted of only one field: agencyId, which is a PCS-specific identificator for the agency,

Response

getImParameters operation returns a response consisted of the following:

- progress statuses – if the operation runs successfully, list of im parameters with the following structure is exposed: parameter name, parameter type in the preferred language of the user. (string, single-choice list, multiple-choice list), mandatory attribute and valid values (for list types).
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

importOperationPoints operation

General

importOperationPoints operation allows Infrastructure Manager agencies to adjust their data in PCS. The list of the Infrastructure Manager agencies responsible for the locations in the particular states can be obtained on demand by RNE.

Update of locations will be done automatically by the system after the rollout of TAF-TSI location reference files (for more details about TAF-TSI, please contact RNE).

Validation

The web service implementation relies on validation on two levels:

- XML schema validation - validates the SOAP request against the schema and throws SOAPFault with corresponding description if they don't match.
- Application-level validation - consistency check (e.g. matches <agency_id> against the database) and return response with status that indicate error code.

Request

The request contains the following information:

- reference_id – (mandatory) national reference id: this id has to be recognizable by the national system for unique national identification of the particular operation point,
- name – (mandatory) name of the operation point in ASCII
- enee_id – (not mandatory) the id given by ENEE database. PCS Core generates PCS-specific enee_id by adding the corresponding UIC Net Code at the beginning of the ID-string. It uses the Net Code which is related to the agency that is issuing the request. This approach forbids that a foreign agency can change the operation points of

other agency.
- valid_from – (not mandatory) the date from which the Operation Point record is valid (begin of validity period). If not provided the system puts the current timestamp date
- valid_to – (not mandatory) the date that signifies the end of validity period for the particular Operation Point. If not provided, the system puts the timestamp 2999-12-31.
- is_passenger flag (not mandatory) - boolean value. Default value is true.
- is_cargo flag (not mandatory) – boolean value. Default value is true.

Response

Operation returns a response consisted of the following:

- id – if the operation runs successfully, the returned value is the operation point id that has been assigned from the PCS Core system.
- reference_id – the given national reference id
- enee_id – if provided
- valid_from – begin of validity period
- valid_to – end of validity period
- is_passenger - flag
- is_cargo - flag
- is_infra - flag
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

For all imported operation points the flag “is_infra” is set to true.

updateOperationPoint operation

General

updateOperationPoint operation allows the responsible agencies to edit data of their operation points. The list of the Infrastructure Manager agencies responsible for the locations in the particular states can be obtained on demand by RNE.

Update of locations will be done automatically by the system after the rollout of TAF-TSI location reference files (for more details about TAF-TSI, please contact RNE).

Request

The request contains the following information:

- id – (mandatory) - operation point id

- All other parameters are not mandatory. They can be set arbitrary.
  - reference_id – the given national reference id
  - enee_id – if provided
  - name – operation point name
  - valid_from – begin of validity period
  - valid_to – end of validity period
  - is_passenger – boolean flag
  - is_cargo - – boolean flag
  - is_infra - – boolean flag

Response

Operation returns a response consisted of the following:

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

This operation is available only for the agencies that are registered in PCS as Infrastructure Managers and are responsible for operation point management in their country.

getOperationPointById operation

General

getOperationPointById operation retrieves the data of the operation point identified by the given id.

Request

The request contains the following information:

- id – (mandatory) operation point id

Response

Operation returns a response consisted of the following:

- id – operation point id
- reference_id – the given national reference id

- enee_id – if provided
- name – name of the operation point
- agency_id – responsible agency id
- valid_from – begin of validity period
- valid_to – end of validity period
- is_passenger – boolean flag
- is_cargo - boolean flag
- is_infra - boolean flag
- status (for errors see the **Appendix A**)
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getDossierPhases operation

General

getDossierPhases operation behaves the same as getDossierProcessPhases operation with the exception of assuming “New Path Request” process type as a default choice.

Request

The request contains the following information:

- dossier type – (mandatory) dossier type for which the list of states needs to be retrieved

Response

Operation returns a response consisted of the following:

- dossier phases – if the operation runs successfully, list of possible dossier phases for “New Path Request” process type with the following structure is exposed: dossierphase_id and dossierphase_description in the preferred language of the user.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getDossierProcessPhases operation (new since PF-Core 2.1.2)

General

getDossierProcessPhases operation retrieves the phases for the given combination of process and dossier type.

Request

The request contains the following information:

- processtype_id – (mandatory) process type
- dossiertype_id – (mandatory) dossier type

Response

Operation returns a response consisted of the following:

- Dossier phases– if the operation runs successfully, list of dossier phases for the given combination of process and dossier type
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getSupportedDossierProcessTypesMatrix (new since PF-Core 2.1.2)

General

getSupportedDossierProcessTypesMatrix operation retrieves the supported dossier types for all the process types.

Response

Operation returns a response consisted of the following:

- Dossier phases– if the operation runs successfully, list of supported dossier types per process type is returned.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

removeDossier operation (new since PF-Core 2.1.2)

General

removeDossier operation closes the dossier referenced by the given id.

The following rules apply:

- leading RU can close/remove dossiers in all process type except Catalogue
- leading IM can close/remove dossiers in Catalogue process type
- no restrictions in test (pfcoretest) environment
- only dossiers in Open/Open(CT) can be removed in production (pfcore) environment

Request

The request contains the following information:

- dossier_id – (mandatory) id of the dossier to be closed

Response

Operation returns a response consisted of the following:

- status
  - 100 – SUCCESS (if the operation runs well)

Example

The typical request looks as follows:

<removeDossierRequest>

<dossierId>11806</dossierId>

<authenticateData>

<sessionid>JSESSIONID= XXXXXXXXXXX…XXXX010020203</sessionid>

</authenticateData>

</removeDossierRequest>

The response looks as follows (if everything was ok):

<removeDossierResponse>

<status>100</status>

</removeDossierResponse>

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

The following operations are new since version PF-Core release 2.2.2

getCalendarItemId operation

General

getCalendarItemId operation retrieves the respective calendar item id to be used (in operations like createDossier and updateDossier) based on the provided bitfield.

Request

The request contains the following information:

- bitfield – (mandatory) id of the dossier to be closed

Response

Operation returns a response consisted of the following:

- calendaritem_id– if the operation runs successfully, valid PCS calendaritem_id for the current ttp_id
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getBanWagonTypes operation

General

getBanWaggonTypes operation retrieves the PCS-specific banwagon type ids accompanied with descriptions in the preferred language of the user.

Response

Operation returns a response consisted of the following:

- banwagon types– list of banwaggon types with the following structure: type_id and description
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getCargoBrakeTypes operation

General

getCargoBrakeTypes operation retrieves the PCS-specific cargo brake type ids accompanied with descriptions in the preferred language of the user.

Response

Operation returns a response consisted of the following:

- cargo brake types– list of cargo brake types with the following structure: type_id and description
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getCargoRemarks operation

General

getCargoRemarks operation retrieves the PCS-specific cargo remark ids accompanied with descriptions in the preferred language of the user.

Response

Operation returns a response consisted of the following:

- cargo remarks– list of cargo remarks with the following structure: id and description
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getGoodsTyoesByCargoTypeEneeId operation

General

getGoodsTypesByCargoTypeEneeId operation exposes PCS good type search feature (by cargotypeenee_id part).

Request

The request contains the following information:

- cargo_type_eneeid– (mandatory) search filter

Responses

Operation returns a response consisted of the following:

- good types– list of good types with the following structure: id, description and cargo_type_enee_id.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getGoodsTypesByDescriotion operation

General

getGoodsTypesByDescription operation exposes PCS good type search feature (by description part).

Request

The request contains the following information:

- description– (mandatory) search filter

Response

Operation returns a response consisted of the following:

- good types– list of good types with the following structure: id, description and cargo_type_enee_id.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getTimetablePeriods operation

General

getTimetablePeriods operation returns all timetable periods defined in PCS Core.

Response

Operation returns a response consisted of the following:

- timetable periods– list of timetable periods with the following structure: name, first date and last date.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getStopTypes operation

General

getStopTypes operation returns all stop types defined in PCS Core.

IMPORTANT: decommissioned in PFIP 2.0, use getActivityTypes instead.

Response

Operation returns a response consisted of the following:

- stop types– list of stop types with the following structure: id and description in the preffered language of the user
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

importTrafficPeriods operation

General

importTrafficPeriods operation exposes already existing importTrafficPeriods PCS Adaptor feature through the web services channel.

Request

The request contains the following information:

- trafficPeriods– (one or many),with the following structure:
  - localName - the name as defined in local planning system
  - bitField – specifies which days in the timetable period the train runs (1s and 0s)
  - ttp_id – timetable period
  - localTrafficPeriodId – the id as defined in local planning system

Response

Operation returns a response consisted of the following:

- trafficPeriods – if the operation is successful, list of traffic periods with the same structure as in the request with PCS id added.

- status
  - 100 – SUCCESS (if the operation runs well)
  - 237 – UNKNOWN TIMETABLE PERIOD (if the ttp_id is not valid)
  - 243 – UPDATE TRAFFIC PERIOD NOT ALLOWED ERROR (if a traffic period with the same local traffic period as in the request already exist for the agency group of the user)
  - 244 – TRAFFIC PERIOD NAME ALREADY IN USE ERROR (if a traffic period with the same localName as in the request already exist for the agency group of the user)
  - 245 – AGENCYGROUP_TRAFFIC_PERIOD_ALREADY_IN_USE_ERROR (if a traffic period with the same combination of ttpId and bitField already exist for the agency group of the user)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getOperationPointByReferenceId operation

General

getOperationPointByReferenceId operation retrieves the operation point identified by the provided reference id.

Request

The request contains the following information:

- reference_id – (mandatory) search filter

Response

Operation returns a response consisted of the following:

- operationPointDetail – if the operation is successful, operation point information with the following structure: id, reference_id, name, enee_id, agency_id, valid_from, valid_to, is_passenger, is_cargo, is_infra
- status
  - 100 – SUCCESS (if the operation runs well)

The operation has been modified (21.12.2016):

- **Start date of the operation point should be before the end date. The end date should be in the future or equal to current date.** With this approach, the integrators would get the operation points which are activated in the future.

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getImParametersForAgencyUicCode operation (deprecated!)

General

getImParametersForAgencyUicCode is deprecated since PCS Core version 2.2.4, at the moment it returns the same results as getImParametersForAgencyUicCode ForTimetablePeriod operation – the latest operation has to be used instead.

Request

The request contains the following information:

- agencyUic_id – (mandatory) search filter

Response

Operation returns a response consisted of the following:

- im parameters – if the operation runs successfully, list of im parameters with the following structure is exposed: parameter name, parameter type in the preferred language of the user. (string, single-choice list, multiple-choice list), mandatory attribute and valid values (for list types).
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAgencyTypes operation

General

getAgencyTypes operation retrieves all valid agency types defined in PCS Core.

Response

Operation returns a response consisted of the following:

- agency types – list agency types with the following structure: id and description in the preffered language of the user
- status

- 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAlignmentDirections operation

General

getAlignmentDirections operation retrieves all valid alignment directions defined in PCS Core.

Response

Operation returns a response consisted of the following:

- alignment directions – list of alignment directions with the following structure: code and description in the preffered language of the user
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getTrafficPeriodsForAgencyGroup operation

General

getTrafficPeriodsForAgencyGroup operation retrieves all defined traffic periods for the provided timetable period in PCS Core for the agency group of the user.

Response

Operation returns a response consisted of the following:

- traffic period details – list of traffic periods with the following structure: id, ttp_id, name, bitfield.
- status
  - 100 – SUCCESS (if the operation runs well)
  - 237 – UNKNOWN TIMETABLE PERIOD

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getDossierWithReferenceIds operation

General

getDossierWithReferenceIds operation represents alternative for getDossier web service operation. In this variant, PFCore attempts to:

- use reference ids instead of PF-specific operation point ids whenever possible (i.e. not all operation points have reference ids)
- use agency uic codes instead of PF-specific agency ids whenever possible (i.e. not all agencies have agency uic codes)

Request

getDossierWithReferenceIds request is consisted of two elements:

- dossierId – mandatory parameter
- version – optional parameter (if not specified, the latest version number is used)

Response

getDossierWithReferenceIds operation returns a response consisted of the following:

- dossier – if the operation runs successfully, the returned value is the dossier data
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

NEW SINCE VERSION 4.0: Check changes in Main Calendar Handling.

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

Non-Relevant operations

There are several additional web services offered by PCS Integration Platform, but they are used for RNE-internal applications and do not bring any added value to IM/RU national systems and their connections:

- getIMUserDataForDossier

- getProdAgencyUserDataForDossier
- getRegBodiesForDossier
- getRegBodiesForIMAgency
- getRUUserDataForDossier

You can ignore these web services during the implementation of the interface to PCS.

The following operations are new since version PF-Core release 2.2.3

getRUIMPairsForDossier operation

General

getRUIMPairsForDossier operation returns the respective RU-IM pairs for the given dossier id. This operation is needed for Short Term Path Request (both non-harmonized and harmonized) process types.

Request

getRUIMPairsForDossier request is consisted of one element:

- dossierId – mandatory parameter

Response

getRUIMPairsForDossier operation returns a response consisted of the following:

- ruImPairs – if the operation runs successfully, the returned value are the dossier RU-IM pairs
- status – description for the response. At the moment, it could take one of the following codes:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the HTTP Header (JSESSIONID cookie).

getRUIMPairsForDossierWithUicIds operation

General

getRUIMPairsForDossierWithUicIds operation represents alternative for getRUIMPairsForDossier web service operation. In this variant, PF-Core attempts to:

- use agency uic codes instead of PF-specific agency ids whenever possible (i.e. not all agencies have agency uic codes)

This operation is needed for Short Term Path Request (both non-harmonized and harmonized) process types.

Request

getRUIMPairsForDossierWithUicIds request is consisted of one element:

- dossierId – mandatory parameter

Response

getRUIMPairsForDossierWithUicIds operation returns a response consisted of the following:

- ruImPairs – if the operation runs successfully, the returned value are the dossier RU-IM pairs
- status – description for the response. At the moment, it could take one of the following codes:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the HTTP Header (JSESSIONID cookie).

The following operations are new since version PF-Core release 2.2.4

getImParametersForTimetablePeriod operation

General

getImParametersForTimetablePeriod is a helper operation which returns all defined national parameters for the given agency id and timetable period.

Request

Search criteria is consisted of two fields: agencyId (PCS-specific identificator for the agency) and ttpId (timetable period).

Response

getImParametersForTimetablePeriod operation returns a response consisted of the following:

- progress statuses – if the operation runs successfully, list of im parameters with the following structure is exposed: parameter name, parameter type in the preferred language of the user. (string, single-choice list, multiple-choice list), mandatory attribute and valid values (for list types). The list can be empty if no im parameters are defined for the timetable period.
- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getImParametersForAgencyUicCodeForTimetablePeriod operation

General

getImParametersForAgencyUicCodeForTimetablePeriod operation retrieves the national parameters defined for IM agency (identified by the provided agency uic code) for the given timetable period.

Request

The request contains the following information:

- agencyUic_id – (mandatory) search filter
- ttpId – (mandatory) timetable period

Response

Operation returns a response consisted of the following:

- im parameters – if the operation runs successfully, list of im parameters with the following structure is exposed: parameter name, parameter type in the preferred language of the user. (string, single-choice list, multiple-choice list), mandatory attribute and valid values (for list types). The list can be empty if no im parameters are defined for the timetable period.
-
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getOperationPointByImAgency operation

General

getOperationPointByImAgency operation retrieves the operation points for which the given IM agency is responsible for.

Request

The request contains the following information:

- im_agency_id – (mandatory) search filter

Response

Operation returns a response consisted of the following:

- operationpoints – if the operation runs successfully, a list of operation points with the following structure is retrieved: id, name, enee_id (if the operation point has a valid enee_id). The list is empty if there's no match for the search criteria.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getOperationPointByCountry operation

General

getOperationPointByCountry operation retrieves the operation points from the country identified by the given ISO 3166 code.

Request

The request contains the following information:

- country_code – (mandatory) search filter

Response

Operation returns a response consisted of the following:

- operationpoints – if the operation runs successfully, a list of operation points with the following structure is retrieved: id, name, enee_id (if the operation point has a valid enee_id). The list is empty if there's no match for the search criteria.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAllCountryCodes operation

General

getAllCountryCodes operation retrieves all countries in the system with their ISO 3166 codes.

Response

Operation returns a response consisted of the following:

- Countries – if the operation runs successfully, list of countries defined in the system with the following structure: country name, ISO 3166 country code.
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

The following operations are new since version PF-Core release 3.0

getActivityTypes operation

General

getActivityTypes operation retrieves all activity types which are valid in PCS.

Response

Operation returns a response consisted of the following:

- list of defined activity types (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getLocationTypes operation

General

getLocationTypes operation retrieves all location types which are valid in PCS.

Response

Operation returns a response consisted of the following:

- list of defined location types (id, description)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getTrainTypes operation

General

getTrainTypes operation retrieves all train types which are valid in PCS.

Response

Operation returns a response consisted of the following:

- list of defined train types (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getTractionModes operation

General

getTractionModes operation retrieves all traction modes defined in PCS.

Response

Operation returns a response consisted of the following:

- list of defined traction modes (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getTrainCCSysCodes operation

General

getTrainCCSystCodes operation retrieves all train CC System codes defined in PCS.

Response

Operation returns a response consisted of the following:

- list of defined train CC System codes (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getTrainRadioSystemCodes operation

General

getTrainRadioSystemCodes operation retrieves all train Radio System codes defined in PCS.

Response

Operation returns a response consisted of the following:

- list of defined train Radio System codes (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getExceptionGaugingIdents operation

General

getExceptionGaugingIdents operation retrieves all exceptional gauging idents defined in PCS.

Response

Operation returns a response consisted of the following:

- list of defined exceptional gauging idents (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAllTiltingFunctions operation

General

getAllTiltingFunctions operation retrieves all tilting functions defined in PCS.

Response

Operation returns a response consisted of the following:

- list of defined tilting functions (id, descriptions)
- status

- 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

The following operations are new since version PCS Core release 3.1

getPathModificationTypeList operation

General

getPathModificationTypeList operation retrieves all reasons in PCS for triggering a Path Modification process.

This list is needed by the RU agencies because they can actually trigger the process.

Response

Operation returns a response consisted of the following:

- list of defined reasons for triggering Path Modification (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getPathModificationRequestRejectionCause operation

General

getPathModificationRequestRejectionCause operation retrieves all reasons in PCS for rejecting a Path Modification request.

This list is needed by the IM agencies because they can actually perform the rejection.

Response

Operation returns a response consisted of the following:

- list of defined rejection reasons for Path Modification request (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getPathModificationOfferRejectionCause operation

General

getPathModificationOfferRejectionCause operation retrieves all reasons in PCS for rejecting a Path Modification offer.

This list is needed by the RU agencies because they can actually perform the rejection.

Response

Operation returns a response consisted of the following:

- list of defined rejection reasons for Path Modification offer (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getPathAlterationTypeList operation

General

getPathAlterationTypeList operation retrieves all reasons in PCS for triggering a Path Alteration process.

This list is needed by the IM agencies because they can actually trigger the process.

Response

Operation returns a response consisted of the following:

- list of defined reasons for Path Alteration (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getPathAlterationRejectionCause operation

General

getPathAlterationRejectionCause operation retrieves all reasons in PCS for rejection of Path Alteration process.

This list is needed by the RU agencies because they can actually perform the rejection.

Response

Operation returns a response consisted of the following:

- list of defined rejection reasons for Path Alteration (id, descriptions)
- status
  - 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAffectedByProcessValues operation

General

getAffectedByProcessValues operation retrieves all statuses for affected pairs in Path Alteration and/or Path Modification processes.

Response

Operation returns a response consisted of the following:

- list of defined rejection reasons for Path Alteration (id, descriptions)
- status

- 100 – SUCCESS (if the operation runs well)

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

updatePath operation

General

updatePath operation is used to update existing path in the dossier.

Only the data of your “agency” is updated and the data of other agencies is protected (leading agency can add path_sections to other agencies regarding the access rights in different dossier phases). Hence, if you send the path xml payload containing out-of-date information of other agencies and your newest changes, the system will filter out your changes, get the latest version of the dossier from the PCS core system and merge your path data with the newest data from the core system (according to access rights checks based on your authentication). With this approach it is ensured that your path update is always consistent with the changes that are made by other agencies.

IMPORTANT: when using updatePath operation, you always have to be sure that you use the proper id-s for the elements like path, path_section etc.

Only if you want to add a path_section you can use an element without id – in that case, the system will generate the id accordingly and new path_section will be added to the provided path. By the next execution of the getDossier operation for the particular dossier, you will have the corresponding ids in the dossier that is delivered in the response.

Request

PCS Core system always expects the entire path xml in the request. Optionally, the ruim_pair_id can be sent with the request (the ruim_pair_id will be resolved by the system but, if the user have more than one pair, this filed must be fulfilled).

Response

updatePath operation returns a response consisted of the following

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 - SUCCESS

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

If the user have more than one pair for the dossier which path is trying to update, but the ruim_pair_id is not provided, the system will return error and the user will be forced to provide ruim_pair_id together with the path XML.

This operation does not support addition of new and deletion of existing paths.

updatePathSections operation

General

updatePathSections operation is used to update existing path sections in the dossier.

IMPORTANT: when using updatePathSections operation, you always have to be sure that you use the proper id-s for the elements like path_section, loco_ident, activity_type etc.

Request

PCS Core system always expects the entire path_section xml in the request. The user can enter different path sections from different paths but from the same dossier. The order of the path sections is not taken into account.

Optionally, the ruim_pair_id can be sent with the request (the ruim_pair_id will be resolved by the system but, if the user have more than one pair, this filed must be fulfilled).

Response

updatePathSections operation returns a response consisted of the following

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 - SUCCESS

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

If the user have more than one pair for the dossier which path_section(s) is trying to update and the ruim_pair_id is not provided, the system will return error and the user will be forced to enter ruim_pair_id together with the path_section XML.

If the user enters path_sections that belongs to different dossiers, the dossier will be resolved from the first path_section and other path_sections that belongs to other dossiers will be ignored.

This operation does not support addition of new and deletion of existing path_sections.

updateProgressStatus operation

General

updateProgressStatus operation is used to update the progress status of user's agency.

Request

The request contains the following information:

- progressstatus_id – new progress status
- comment – element which is mandatory when dossier is rejected. (in other cases if entered will be ignored)
- ruim_pair_id – the ru-im pair id

Response

updateProgressStatus operation returns a response consisted of the following

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 - SUCCESS

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

updateDossierStatus

General

updateDossierStatus operation is used to update the dossier status of the dossier (promote the dossier to next phase)

Request

The request contains the following information:

- dossierstatus_id – the new dossier status of the dossier
- ruim_pair_id – the ru-im pair id
- comment - optional element in case the dossier is rejected/withdrawn reason should be entered. If not entered, error code 634 (DOSSIER_REJECTION_CAUSE_EMPTY) will be returned

In **updateDossierStatusRequest** new optional element **comment** is added (21.12.2016)

```
<xsd:element name="comment" type="v3_0:RejectionCause" minOccurs="0">
  <xsd:annotation>
    <xsd:documentation>Comment that should be entered in case when dossier is withdrawn/rejected.
    In other cases if entered will be ignored.</xsd:documentation>
```

```
    </xsd:annotation>
  </xsd:element>
```

Response

updateProgressStatus operation returns a response consisted of the following

- status – description for the response. The list of error codes can be found in the Appendix A at the end of this document. If everything went ok, the return code is:
  - 100 - SUCCESS

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

Main Calendar Handling (Since version 4.0)

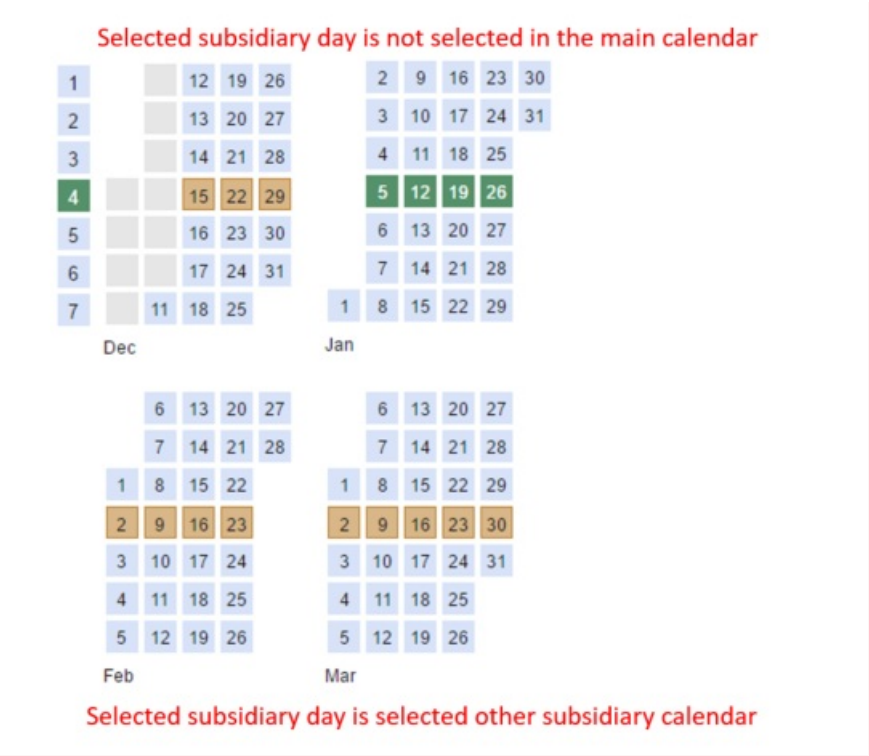
Calendar Consistency Checks (via Main/Subsidiary Timetables)

With the introduction of Main/Subsidiary Timetable Concepts, PCS provides the following two calendar consistency checks:

- Check whether **the subsidiary calendar is inside the main calendar frame**
- Check whether there is **NO overlapping of days for subsidiary calendars**.

It is important to stress that **in PCS Core the main calendar should actually be a frame of all days where the train is running (including the days from subsidiaries)**

PCS Core informs the users about existing calendar inconsistencies in the Calendar Editor (PCS Core GUI) in pop-up messages over the yellow warnings.



Main Calendar Handling via PCS IP

Starting with v4 endpoint, PCS IP contains some differences in the way Main Timetable Calendar is handled:

- It **always retrieves the complementary main calendar (without days already taken in the subsidiaries) via getDossier** Reason for this: integrators know the exact days when the train is running, **no duplication, no risk of double booking**.
- It is still able to do operations on the main calendar (e.g. add/deselect days). The complementary main calendar should always be sent in create/update dossier operations. PCS will take care of main calendar management (by using the complementary main calendar and the subsidiary calendars to reconstruct the main calendar).

Example: the attached (2.62.2... txt) reference point calendar in the main timetable in createDossier/updateDossierRUIMPair requests (NOTE**for readability purposes, only a portion of the calendar bitfield representing one week is rendered here**)...

The following operations are new since PCS Core version release 4.1

getDossierAttachment operation

General

getDossierAttachment operation retrieves the dossier attachment for the given attachment id. The attachment id can be retrieved from the dossier (getDossier operation).

Response

Operation returns a response consisted of the following:

- status
  - 100 – SUCCESS (if the operation runs well)
- attachment_file
  - The content of the attachment in xsd:base64Binary format.

Constraints

This operation is protected, so the user must be authenticated beforehand. The corresponding session id must be present in the request body, at the end of the XML request structure - <authenticateData><sessionid>JSESSIONID=...</sessionid></authenticateData>.

getAllDossiersExtended operation

getAllDossiersExtended operation is an extension of the existing**getAllDossiers** operations with a possibility to get the dossiers by:

- process type and
- train type

The response and the constraints are not changed.

The following operations are new since PCS Core version release 5.0 (PCS NG Release 1.0)

getHarmonizationStatuses operations

General

getHarmonizationStatuses operation is used to retrieve the valid values for the border harmonization statuses

Response

Operation returns a response consisted of the following:

- status
  - 100 – SUCCESS (if the operation runs well)
- list of defined border harmonization statuses

File

Document:
2.62.2_main_calendar_handling_via_pcs_ip.txt

Print

Printer-friendly version
Send by email
PDF version

Details

State: Draft
Topic: PCS Interface
Integration Platform
Area: Technical
Release: 5.x
Company All
Type:
Keywords: pcs ip
webservice
handbook

Translations

No translations

Source URL: https://cms.rne.eu/pcs/pcs-documentation-0/web-servicesold