**Revision history**

<table>
<thead>
<tr>
<th>Date</th>
<th>Changed</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>January 2015</td>
<td>Added keydef for product name keyword</td>
<td>BA</td>
</tr>
<tr>
<td>September 2014</td>
<td>First version</td>
<td>AA</td>
</tr>
</tbody>
</table>
Contents

About this manual
General about this manual.................................................................5

Introducing the Danfoss Telematics Web Services API
General introduction ..............................................................................6
Authentication and security services......................................................6
Runtime services....................................................................................6
Status services......................................................................................6
File services..........................................................................................6
Geo mapping tasks................................................................................7

General Information about Headers being used with Danfoss Telematics Web Services
General information..............................................................................8

Authentication and Security - The AuthServices Interface
General..................................................................................................9
The authentication process.................................................................9

The ClientLogin Interface
General...............................................................................................10
Overview.............................................................................................10
Example scenario................................................................................11
Parameters...........................................................................................11
Raw SOAP Samples............................................................................12
clientLogin Error Codes.......................................................................12
Using ClientLogin................................................................................13

The Runtime Entities
The Runtime Entities Service...............................................................14
Overview.............................................................................................14
RuntimeEntities Service Operations................................................14
getAllVisibleRuntimeDeviceGroups....................................................14
getAllVisibleRuntimeDeviceGroups Parameter...................................14
getAllVisibleRuntimeDeviceGroups Examples...................................15
getAllVisibleRuntimeDevices..............................................................15
getAllVisibleRuntimeDevices Parameter..........................................15
getAllVisibleRuntimeDevices Examples..........................................16
getAllVisibleRuntimeMachineGroups...............................................17
getAllVisibleRuntimeMachineGroups Parameter................................17
getAllVisibleRuntimeMachineGroups Examples................................17
getAllVisibleRuntimeMachines Parameter........................................18
getAllVisibleRuntimeMachines Examples........................................19
get DeviceRuntimeDetails....................................................................19
getDeviceRuntimeDetails Parameter.................................................19
getDeviceRuntimeDetails Examples..................................................20

Monitoring Devices and Vehicles - Status
The Status Service................................................................................22
Overview.............................................................................................22
Status Service Operations...................................................................22
getStatusByDeviceIds.........................................................................22
getStatusByMaschinelnIds.................................................................22
getDeviceStates..................................................................................22
getDeviceState Parameter...................................................................23
getDeviceState Examples...................................................................23
getMachineStates...............................................................................24
getMachineState Examples...............................................................25

File Handling (Transferring Files)
The File Services................................................................................27
Overview.............................................................................................27
File Service Operations.......................................................................28
Contents

Geo Mapping Service

- getLastKnownPositionForObject ................................................................. 39
  - getLastKnownPositionForObject Parameter ........................................... 39
  - Sample getLastKnownPositionForObject Request .................................. 39
  - Sample getLastKnownPositionForObject Response .............................. 39
- getLastKnownPositionForObject Error Cades............................................. 39

Devices Service

- The Devices Services ................................................................................. 40
- Overview .................................................................................................... 40
- Device Service Operations ......................................................................... 40
  - changeDeviceModeTo .............................................................................. 40
    - changeDeviceModeTo Parameter ......................................................... 40
    - changeDeviceModeTo Examples ......................................................... 41

Service and support

- Telematics service and support ................................................................. 42
About this manual

**General about this manual**

This document is part of the Danfoss Telematics Solutions Portal and provides important information on the intended use, and operation of the portal described below.
Introducing the Danfoss Telematics Web Services API

General introduction

This document is aimed at programmers who are developing client applications that access Danfoss services and who want to implement a programmatic way to access Danfoss services. You will need to know some service-specific details when incorporating the web service functions. In addition, you will need to know how to make WS-I compliant SOAP requests and handle responses. This documentation gives an overview of Danfoss web services. All the web services functionality is divided into groups. There are web services for

- **Authentication and Security Services**
- **Runtime Services** – Runtime Entities
- **Status Services** – Monitoring Devices and Vehicles - Status
- **File Services** – File Handling (Transferring Files)
- **Geo Mapping Services**

Authentication and security services

**clientLogin:** This service must be used for login. The service returns a security token which must be used for subsequent service requests. The token has a limited lifetime. The default lifetime is 30 minutes. The lifetime may vary for different users. Please ask Danfoss staff if you need to know more details.

Runtime services

**getAllVisibleRuntimeDeviceGroups:** This service returns all device groups to which the user has access to. The result depends on the token which has been generated for a given user with specific rights and roles.

**getAllVisibleRuntimeDevices:** Query by example. Same as above, but returns devices.

**getAllVisibleRuntimeMachineGroups:** Query by example. Same as above, but returns machine groups.

**getAllVisibleRuntimeMachines:** Query by example. Same as above, but returns machines.

Status services

This service can be used to retrieve information about the status of devices and machines, e.g. vehicles.

**getStatusByDeviceIds:** Returns the status of devices. Unique primary keys are being used to request the status for specific devices. You need to find the unique primary keys first, by calling the appropriate services at first.

**getStatusByMachineIds:** Return the status of machines. Unique primary keys are being used to request the status for specific machines. You need to find the unique primary keys first, by calling the appropriate services at first.

File services

**downloadFile:** Downloads files.

**markFetchedFile:** This service marks files. You should mark files as soon as they are not needed anymore. Then Danfoss will delete them, which means that they will not be available anymore.

**searchFiles:** Query by example. One must search files before they can be downloaded.

**deleteUploadFiles:** Deletes a set of upload files associated by the file's primary key.

**uploadFileToMachine:** uploads one file to a machine.

**uploadFirmwareToMachine:** uploads firmware to a machine.

**uploadDbsConfigToMachine:** uploads Dashboard Configurations to a machine.
Introducing the Danfoss Telematics Web Services API

**Geo mapping tasks**

- `getLastKnownPositionForAllObjects`: Return a list with positions for all objects (user view). 
  Object is a legacy term for device which represents a GSM modem.

- `getLastKnownPositionForObject`: Return a list with the last known position for one given object. 
  Object is a legacy term for device which represents a GSM modem.
General Information about Headers being used with Danfoss Telematics Web Services

General information

With the exception for the authentication and security services, every request to the Danfoss Web Services must enclose a header. Each web service function has an input parameter pH of type pHType.

The information obtained from the header is used for authenticating the request and associating it with a Danfoss Session.

The header is enclosed with the requests and with the responses for all Danfoss web services.

The header looks as follows:

The Header pHType Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proemionId</td>
<td>The proemionId is a key for authenticating. Danfoss web service Requests. This identifier will be available for users being registered as a Danfoss user (customers).</td>
</tr>
<tr>
<td>clientId</td>
<td>This identifier must be generated and supplied by the client. The client identifier is being used by Danfoss to determine from which client the request is coming from. Each client must use a unique clientId. Actually the utilization of this attribute is disabled. Nevertheless client applications should fill in this attribute. Currently the authenticationKey is used alternatively.</td>
</tr>
<tr>
<td>authenticationKey</td>
<td>A security token. This is a key being generated by Danfoss's web service clientLogin. One must call the clientLogin service before accessing any other services. This token may only be used by one single and unique client. Danfoss uses the authenticationKey as a session identifier. This means that only the authentication may be empty for the clientLogin only.</td>
</tr>
<tr>
<td>requestId</td>
<td>This is being used by Danfoss for internal purposes.</td>
</tr>
<tr>
<td>responseld</td>
<td>This is being used by Danfoss for internal purposes.</td>
</tr>
<tr>
<td>customerKeyValuePair</td>
<td>Contains data that will not be used for any operations within the Danfoss services but be returned with the header in the response</td>
</tr>
</tbody>
</table>
Authentication and Security - The AuthServices Interface

General

Here at Danfoss, we welcome the development of applications that rely on and communicate with Danfoss services. However, many of these services require that users log into their Danfoss Accounts, and authentication by outside applications has been – until now – somewhat cumbersome. To improve this experience, we are pleased to offer alternatives for Danfoss Account authentication. Our goals in providing these alternatives are to enhance both performance and security, as well as to streamline the process for developers of client applications. As new options are developed, we’ll post them. All of Danfoss’s web services require at least the usage of the clientLogin function. Web service clients must request a security token to be able to call subsequent web services methods. The AuthServices interface provides several methods for acquiring and managing authentication tokens. Once a web application has received a token, it can request access to a Proemion service. For information on forming the access request to a Danfoss service, see the documentation for the service. Danfoss Accounts authentication for web-based applications allows the application to access a Danfoss service protected by a user’s Danfoss account. The authentication process flows as follows:

1. When the web service client application needs to access the user’s Danfoss service data, it makes a clientLogin request call. This request contains the login data as user’s name, password and customer number. The application logs into the Danfoss account and Danfoss then decides whether to grant or deny access to the web service.

2. If the application successfully logs in and access is granted, Danfoss Accounts responds to the application. The response will contain a security token with a given durability; it might be that the token can only be used one time.

3. The web service client application contacts the Danfoss service, using the authentication token to act as an agent for the user.

4. If the Danfoss service recognizes the token, it will supply the requested data.

The Authentication Proxy diagram shown below illustrates interactions between the two entities involved: a web service client application and Danfoss servers.

The authentication process

Developers can opt to handle authentication with either secure tokens or non-secure tokens. The use of secure tokens requires that the web service client application to be registered with Danfoss and file a certificate; if registered, the web application can secure all requests referencing an authentication token with a digital signature. Secure tokens are not available, in case of need contact DTS Team.

This document provides information on how to incorporate use of the authentication proxy into your web-based application.
The ClientLogin Interface

General

Developers of installed applications have the option of using a programmatic login method to access Danfoss services protected by user login. An installed web service client application is installed on a device, such as a desktop computer or a cell phone, as opposed to a web service server application. With programmatic login implemented, an application's user can log into their Danfoss account from inside the application. The application then contacts Danfoss with the login data and requests access to a specified Danfoss service. Once access is authorized, the user can create, read, update, or delete service data as needed using the application interface. With programmatic login, Danfoss supplies the application with a token that can be referenced in all subsequent requests. This approach offers several advantages, including:

- Performance is improved because login data is validated only once per session instead of with each request.
- Security is tightened by minimizing the number of times login data is transmitted per session.
- The authentication process can incorporate additional security measures.
- Authentication measures can be more easily enhanced and extended as required.

Use this interface in your installed application to programmatically log into Danfoss accounts. After collecting login information from a user, call clientLogin to request access to the user's account. Once the login information has been successfully authenticated, Danfoss will return a token, which your application will reference each time it requests access to the user's account, such as to get or post data. The token remains valid for a set length of time, defined by whichever Danfoss service you're working with. clientLogin uses standard security measures to protect user account information. To block bots and other entities from breaking user passwords, Danfoss Accounts may add additional measures to the authentication process when the server suspects an illegal intrusion, such as after too many incorrect login attempts.

Note that clientLogin does not support service sign ups or other account maintenance tasks. Users must have an existing account before using this feature. If a user tries to log in without an account, this interface will return a "login failed" response. In addition, if other account maintenance steps – such as email verification or acceptance of terms – are not completed, the login attempt will fail.

Overview

WSDL Definition

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Endpoint</td>
</tr>
<tr>
<td><a href="https://services.proemion.com/proemion-ws/ws-proemion-auth/2009/01/15/AuthService">https://services.proemion.com/proemion-ws/ws-proemion-auth/2009/01/15/AuthService</a></td>
</tr>
<tr>
<td>WSDL URL</td>
</tr>
<tr>
<td>Namespace</td>
</tr>
<tr>
<td>urn:pAuthServices</td>
</tr>
<tr>
<td>Binding</td>
</tr>
<tr>
<td>AuthServiceBinding</td>
</tr>
<tr>
<td>SOAP Version</td>
</tr>
<tr>
<td>SOAP 1.1</td>
</tr>
<tr>
<td>customerKeyValuePair</td>
</tr>
<tr>
<td>Contains data that will not be used for any operations within the Danfoss services but be returned with the header in the response</td>
</tr>
<tr>
<td>Style</td>
</tr>
<tr>
<td>Document</td>
</tr>
<tr>
<td>WS-A version</td>
</tr>
<tr>
<td>NONE</td>
</tr>
</tbody>
</table>

Definition Parts

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
</tr>
<tr>
<td>AuthService.wsdl</td>
</tr>
<tr>
<td>AuthService.wsdl</td>
</tr>
<tr>
<td><a href="https://services.proemion.com/2009/01/15/pProcessingProblem.xsd">https://services.proemion.com/2009/01/15/pProcessingProblem.xsd</a></td>
</tr>
</tbody>
</table>
The ClientLogin Interface

Definition Parts (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthService.wsdl</td>
<td><a href="https://services.proemion.com/2009/01/15/pClientLogin.xsd">https://services.proemion.com/2009/01/15/pClientLogin.xsd</a></td>
</tr>
<tr>
<td>pAuthTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/01/15/pAuthTypes.xsd">https://services.proemion.com/2009/01/15/pAuthTypes.xsd</a></td>
</tr>
<tr>
<td>pBaseTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/01/15/pBaseTypes.xsd">https://services.proemion.com/2009/01/15/pBaseTypes.xsd</a></td>
</tr>
<tr>
<td>pHeaderTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/01/15/pHeaderTypes.xsd">https://services.proemion.com/2009/01/15/pHeaderTypes.xsd</a></td>
</tr>
</tbody>
</table>

Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
<th>One-Way</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>getStatusByDeviceIds</td>
<td>Literal</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>getStatusByMachineIds</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

Example scenario

In this scenario, you’re creating an installed application that communicates with Proemion’s File Transfer service. Users of Proemion File Transfer manage their files, add, update, or delete files, and do different other things. For your application, you want to be able download or upload Proemion File Transfer data in your interface and provide tools to manipulate the data.

To accomplish this, you need to get access to your account. Before you can access the files, you need to request authorization from Proemion. Once you’ve been successfully authenticated and received a token from Proemion, you can access your files, referencing the token in each request.

Parameters

The input parameter of clientLogin is of type ptClientLogin consisting of the following parameters

The clientLogin Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accountType</td>
<td>Type of account to be authenticated. Possible values are: proemion (authenticate as a Proemion account only). Other account types may be introduced in the future. The parameter is case insensitive.</td>
</tr>
<tr>
<td>user</td>
<td>User Name.</td>
</tr>
<tr>
<td>passwd</td>
<td>User’s password.</td>
</tr>
<tr>
<td>customerNr</td>
<td>User’s customer number. This parameter is for some user’s optional and for some user’s required. If possible, then supply this parameter.</td>
</tr>
<tr>
<td>email</td>
<td>User’s full email address. It must include the domain (i.e. <a href="mailto:johndoe@business.com">johndoe@business.com</a>). This parameter is for some user’s optional and for some user’s required. If possible, then supply this parameter.</td>
</tr>
<tr>
<td>service</td>
<td>Name of the Proemion service for which authorization is requested. Each service that uses Proemion accounts is assigned a name value; for example, the name associated with Proemion File Transfer is fil. This parameter is required when accessing Proemion File Transfer services. Refer to the Proemion documentation for specific services. You can define more than one service by using comma separated services.</td>
</tr>
<tr>
<td>source</td>
<td>Short string identifying your application, for logging purposes. This string should take the form: companyName-applicationName-versionID</td>
</tr>
<tr>
<td>loginToken</td>
<td>reserved for future purposes</td>
</tr>
<tr>
<td>loginCaptcha</td>
<td>reserved for future purposes</td>
</tr>
<tr>
<td><a href="mailto:johndoe@business.com">mailto:johndoe@business.com</a></td>
<td></td>
</tr>
</tbody>
</table>
The ClientLogin Interface

Raw SOAP Samples

A clientLogin Request

```xml
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:clientLogin"
xmlns:urn1="urn:pst">
<soapenv:Header/>
<soapenv:Body>
<urn:clientLogin>
<urn1:clientLogin>
<urn1:accountType>proemion</urn1:accountType>
<urn1:user>webserviceuser</urn1:user>
<urn1:password>mysecretpassword</urn1:password>
<urn1:email>webserviceuser@acme.com</urn1:email>
<urn1:customerId>acme</urn1:customerId>
<urn1:application>ws</urn1:application>
<urn1:source>mywebservice client v 3.4</urn1:source>
<urn1:loginToken>reserved for future</urn1:loginToken>
<urn1:loginCaptcha>reserved for future</urn1:loginCaptcha>
</urn1:clientLogin>
</urn:clientLogin>
</soapenv:Body>
</soapenv:Envelope>
```

A clientLogin Response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
<ns7:clientLoginResponse
xmlns="urn:pst"
xmlns:ns2="urn:pbt"
xmlns:ns3="urn:pProcessingFault"
xmlns:ns4="urn:authSubSessionToken"
xmlns:ns5="urn:authSubTokenInfo"
xmlns:ns6="urn:authSubRevokeToken"
xmlns:ns7="urn:clientLogin">
<clientLoginResponse>
<authToken>urn:uuid:0832dc1d-7d5f-40d6-92b6-20e097167866</authToken>
</clientLoginResponse>
</ns7:clientLoginResponse>
</S:Body>
</S:Envelope>
```

clientLogin Error Codes

<table>
<thead>
<tr>
<th>Error No.</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>BadAuthentication</td>
<td>The login request used a username or password or customer number that is not recognized</td>
</tr>
<tr>
<td>1001</td>
<td>NotVerified</td>
<td>The account email address has not been verified. The user will need to access their Proemion account directly to resolve the issue before logging in using a non-Proemion application.</td>
</tr>
<tr>
<td>1002</td>
<td>Unknown</td>
<td>The error is unknown or unspecified; the request contained invalid input or was malformed</td>
</tr>
<tr>
<td>1005</td>
<td>AccountDeleted</td>
<td>The user account has been deleted.</td>
</tr>
<tr>
<td>1006</td>
<td>AccountDeleted</td>
<td>The user account has been deleted.</td>
</tr>
</tbody>
</table>
The ClientLogin Interface

*clientLogin Error Codes (continued)*

<table>
<thead>
<tr>
<th>Error No.</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1007</td>
<td>ServiceDisabled</td>
<td>The user’s access to the specified service has been disabled. (The user account may still be valid.)</td>
</tr>
<tr>
<td>1008</td>
<td>ServiceUnavailable</td>
<td>The service is not available; try again later.</td>
</tr>
</tbody>
</table>

**Using ClientLogin**

Incorporating clientLogin into your application will require these tasks:

1. Create a web service client submitting login data.

2. Handle responses from Proemion. There are four possible responses to a login request:
   - success
   - failure with an explanatory error code
   - invalid request, generally resulting from a malformed request
   - failure with extended security measures

A success response contains an authentication token labeled authToken. This token must be included in all subsequent requests to the Proemion service for this account. Authentication tokens should be closely guarded and should not be given to any other application, as they represent access to the user’s account. The time limit on the token varies depending on which service issued it.

A failure response includes one or more error codes. Please note that clientLogin does not differentiate between a failure due to an incorrect password or one due to an unrecognized user name (for example, if the user has not yet signed up for an account). Your application will need to handle all possible error message as appropriate.
The Runtime Entities

The Runtime Entities Service

Overview

WSDL Definition

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Endpoint</td>
</tr>
<tr>
<td>WSDL URL</td>
</tr>
<tr>
<td>Namespace</td>
</tr>
<tr>
<td>urn:runtimeEntitiesServices</td>
</tr>
<tr>
<td>Binding</td>
</tr>
<tr>
<td>RuntimeEntitiesServiceBinding</td>
</tr>
<tr>
<td>SOAP Version</td>
</tr>
<tr>
<td>SOAP 1.1</td>
</tr>
<tr>
<td>Style</td>
</tr>
<tr>
<td>Document</td>
</tr>
<tr>
<td>WS-A version</td>
</tr>
<tr>
<td>NONE</td>
</tr>
</tbody>
</table>

Definition Parts

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>runtimeEntitiesServices.wsdl</td>
</tr>
<tr>
<td>pProcessingProblem.xsd</td>
</tr>
<tr>
<td><a href="https://services.proemion.com/2009/07/16/pProcessingProblem.xsd">https://services.proemion.com/2009/07/16/pProcessingProblem.xsd</a></td>
</tr>
<tr>
<td>pUserRead.xsd</td>
</tr>
<tr>
<td><a href="https://services.proemion.com/2009/07/16/pUserRead.xsd">https://services.proemion.com/2009/07/16/pUserRead.xsd</a></td>
</tr>
<tr>
<td>pAdminTypes.xsd</td>
</tr>
<tr>
<td><a href="http://services.proemion.com/2009/07/16/pAdminTypes.xsd">http://services.proemion.com/2009/07/16/pAdminTypes.xsd</a></td>
</tr>
<tr>
<td>pBaseTypes.xsd</td>
</tr>
<tr>
<td><a href="http://services.proemion.com/2009/07/16/pBaseTypes.xsd">http://services.proemion.com/2009/07/16/pBaseTypes.xsd</a></td>
</tr>
<tr>
<td>pHeaderTypes.xsd</td>
</tr>
<tr>
<td><a href="http://services.proemion.com/2009/07/16/pHeaderTypes.xsd">http://services.proemion.com/2009/07/16/pHeaderTypes.xsd</a></td>
</tr>
<tr>
<td>getAllVisibleAdminEntities.xsd</td>
</tr>
<tr>
<td><a href="http://services.proemion.com/2009/07/16/getAllVisibleAdminEntities.xsd">http://services.proemion.com/2009/07/16/getAllVisibleAdminEntities.xsd</a></td>
</tr>
</tbody>
</table>

Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
<th>One-Way</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>getAllVisibleRuntimeDeviceGroups</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>getAllVisibleRuntimeDevices</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>getAllVisibleRuntimeMachineGroups</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>getAllVisibleRuntimeMachines</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RuntimeEntities Service Operations

getAllVisibleRuntimeDeviceGroups

ggetAllVisibleRuntimeDeviceGroups Parameter

No input parameters needed. Just the header containing the security token.

The service will return a list with device. For each found device two attributes are returned as shown in the table below.
The Runtime Entities

The getAllVisibleRuntimeDeviceGroups Response Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceGroupPk</td>
<td>A unique primary key for a device group. Must be used for specific subsequent service call requests.</td>
</tr>
<tr>
<td>deviceGroupName</td>
<td>A human readable and name for a device group.</td>
</tr>
</tbody>
</table>

getAllVisibleRuntimeDeviceGroups Examples

A getAllVisibleRuntimeDeviceGroups request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:urn="urn:getAllVisibleRuntimeEntities"
     xmlns:urn1="urn:pHeader">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:getAllVisibleRuntimeDeviceGroupsRequest>
      <urn1:pHeader>
        ...
        <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
        ...
      </urn1:pHeader>
      <urn:getAllVisibleRuntimeDeviceGroupsRequest>
        </urn:getAllVisibleRuntimeDeviceGroupsRequest>
    </urn:getAllVisibleRuntimeDeviceGroupsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A getAllVisibleRuntimeDeviceGroups response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:pHeader">
  <S:Body>
    <ns5:getAllVisibleRuntimeDeviceGroupsResponse
      xmlns="urn:pHeader"
      xmlns:ns2="urn:pbt"
      xmlns:ns3="urn:pProcessingFault"
      xmlns:ns4="urn:pat"
      xmlns:ns5="urn:getAllVisibleRuntimeEntities">
      <pHeader>
        ...
      </pHeader>
      <authenticationKey>urn:uuid:...</authenticationKey>
      ...
      <ns4:getAllVisibleDeviceGroupsResponse>
        <ns4:deviceGroups>
        <ns2:result>
          <ns2:deviceGroupPk>683</ns2:deviceGroupPk>
          <ns2:deviceGroupName>Proemion Device Pool</ns2:deviceGroupName>
        </ns2:deviceGroups>
        ...
        </ns4:deviceGroups>
      </ns4:getAllVisibleDeviceGroupsResponse>
    </ns5:getAllVisibleRuntimeDeviceGroupsResponse>
  </S:Body>
</S:Envelope>
```

getAllVisibleRuntimeDevices

getAllVisibleRuntimeDevices Parameter

No input parameters needed. Just the header containing the security token.

The service will return a list with device. For each found device two attributes are returned as shown in the table below.
The Runtime Entities

The `getAllVisibleRuntimeDevices` Response Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devicePk</td>
<td>A unique primary key for a device. Must be used for specific subsequent service calls.</td>
</tr>
<tr>
<td>deviceName</td>
<td>A human readable and name for a device.</td>
</tr>
</tbody>
</table>

`getAllVisibleRuntimeDevices` Examples

A `getAllVisibleRuntimeDevices` request

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:getAllVisibleRuntimeEntities"
xmlns:urn1="urn:pHeader">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:getAllVisibleRuntimeDevicesRequest>
      <urn1:pHeader>
        ...
        <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
        ...
      </urn1:pHeader>
      <urn:getAllVisibleRuntimeDevicesRequest/>
    </urn:getAllVisibleRuntimeDevicesRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A `getAllVisibleRuntimeDevices` response

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns5:getAllVisibleRuntimeDevicesResponse
      xmlns="urn:pHeader"
      xmlns:ns2="urn:pbt"
      xmlns:ns3="urn:pProcessingFault"
      xmlns:ns4="urn:pat"
      xmlns:ns5="urn:getAllVisibleRuntimeEntities">
      <pHeader>
        ...
        <authenticationKey>urn:uuid:...</authenticationKey>
        ...
      </pHeader>
      <ns4:getAllVisibleDevicesResponse>
        <ns4:devices>
          <ns2:result>
            <ns2:devicePk>355633001000636</ns2:devicePk>
            <ns2:deviceName>FD RM 65 Defender</ns2:deviceName>
          </ns2:result>
          <ns2:result>
            <ns2:devicePk>355633001108611</ns2:devicePk>
            <ns2:deviceName>OEM Controls 1</ns2:deviceName>
          </ns2:result>
        </ns4:devices>
      </ns4:getAllVisibleDevicesResponse>
    </ns5:getAllVisibleRuntimeDevicesResponse>
  </S:Body>
</S:Envelope>
```
The Runtime Entities

**getAllVisibleRuntimeMachineGroups**

**getAllVisibleRuntimeMachineGroups Parameter**

No input parameters needed. Just the header containing the security token.

The service will return a list with device. For each found device two attributes are returned as shown in the table below.

**The getAllVisibleRuntimeMachineGroups Response Parameters**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machineGroupPk</td>
</tr>
<tr>
<td>machineGroupName</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A unique primary key for a machine group. Must be used for specific subsequent service call requests.</td>
</tr>
<tr>
<td>A human readable and usable name for a machine group.</td>
</tr>
</tbody>
</table>

**getAllVisibleRuntimeMachineGroups Examples**

A `getAllVisibleRuntimeMachineGroups` request

```xml
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
    xmlns:urn="urn:getAllVisibleRuntimeEntities"
    xmlns:urn1="urn:pHeader">
    <soapenv:Header/>
    <soapenv:Body>
        <urn:getAllVisibleRuntimeMachineGroupsRequest>
            <urn1:pHeader>
                ...
                <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
                ...
            </urn1:pHeader>
        </urn:getAllVisibleRuntimeMachineGroupsRequest>
    </soapenv:Body>
</soapenv:Envelope>
```
The Runtime Entities

A `getAllVisibleRuntimeMachineGroups` response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:pHeader="urn:pHeader"
    xmlns:pbt="urn:pbt"
    xmlns:pProcessingFault="urn:pProcessingFault"
    xmlns:pat="urn:pat"
    xmlns:getAllVisibleRuntimeEntities="urn:getAllVisibleRuntimeEntities">
    <S:Body>
        <ns5:getAllVisibleRuntimeMachineGroupsResponse>
            <pHeader>
                ...
            </pHeader>
            <ns4:getAllVisibleMachineGroupsResponse>
                <ns4:machineGroups>
                    <ns2:result>
                        <ns2:machineGroupPk>820</ns2:machineGroupPk>
                        <ns2:machineGroupName>Cummins USA Machine Group</ns2:machineGroupName>
                    </ns2:result>
                    <ns2:result>
                        <ns2:machineGroupPk>680</ns2:machineGroupPk>
                        <ns2:machineGroupName>Raymond Vehicle Group</ns2:machineGroupName>
                    </ns2:result>
                </ns4:machineGroups>
            </ns4:getAllVisibleMachineGroupsResponse>
        </ns5:getAllVisibleRuntimeMachineGroupsResponse>
    </S:Body>
</S:Envelope>
```

`getAllVisibleRuntimeMachines Parameter`

No input parameters needed. Just the header containing the security token.

The service will return a list with machines. For each found machine two attributes are returned as shown in the table below.

**The `getAllVisibleRuntimeMachines` Response Parameters**

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machinePk</td>
</tr>
<tr>
<td>machineName</td>
</tr>
</tbody>
</table>
The Runtime Entities

getAllVisibleRuntimeMachines Examples

A getAllVisibleRuntimeMachines request

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:urn="urn:getAllVisibleRuntimeEntities"
  xmlns:urn1="urn:pHeader">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:getAllVisibleRuntimeMachinesRequest>
      <urn1:pHeader>
        ...
      </urn1:pHeader>
    </urn:getAllVisibleRuntimeMachinesRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A getAllVisibleRuntimeMachines response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns5:getAllVisibleRuntimeMachinesResponse
      xmlns="urn:pHeader"
      xmlns:ns2="urn:pbt"
      xmlns:ns3="urn:pProcessingFault"
      xmlns:ns4="urn:pat"
      xmlns:ns5="urn:getAllVisibleRuntimeEntities">
      <pHeader>
        ...
      </pHeader>
      <ns4:getAllVisibleMachinesResponse>
        <ns2:machines>
          <ns2:result>
            <ns2:machinePk>1234</ns2:machinePk>
            <ns2:machineName>Proemion Demo Machine</ns2:machineName>
          </ns2:result>
          <ns2:result>
            <ns2:machinePk>1235</ns2:machinePk>
            <ns2:machineName>ACME Machine</ns2:machineName>
          </ns2:result>
        </ns2:machines>
      </ns4:getAllVisibleMachinesResponse>
    </ns5:getAllVisibleRuntimeMachinesResponse>
  </S:Body>
</S:Envelope>
```

get DeviceRuntimeDetails

getDeviceRuntimeDetails Parameter

The getDeviceRuntimeDetails Request Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>imei</td>
<td>IMEI number of the device. A unique key for the device.</td>
</tr>
</tbody>
</table>
The Runtime Entities

### The `getDeviceRuntimeDetails` Response Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devicePk</td>
<td>A unique primary key for the device. Must be used for specific subsequent service call requests.</td>
</tr>
<tr>
<td>deviceName</td>
<td>A human readable and usable name for the device.</td>
</tr>
<tr>
<td>imei</td>
<td>A unique key for the device.</td>
</tr>
<tr>
<td>hasGps</td>
<td>Deliver true, if device has GPS, otherwise false.</td>
</tr>
<tr>
<td>deviceGroups</td>
<td>Delivers all groups that contain the device.</td>
</tr>
<tr>
<td>controllers</td>
<td>Delivers all attached controller for this device.</td>
</tr>
<tr>
<td>machinePk</td>
<td>A unique primary key for the machine.</td>
</tr>
<tr>
<td>machineName</td>
<td>A human readable and usable name for the machine.</td>
</tr>
<tr>
<td>functionSets</td>
<td>Delivers all function set for the device.</td>
</tr>
<tr>
<td>orgUnitPk</td>
<td>Delivers all organization units for the device.</td>
</tr>
<tr>
<td>comment</td>
<td>Description for this device.</td>
</tr>
</tbody>
</table>

### `getDeviceRuntimeDetails` Examples

A `getRuntimeDeviceDetails` request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:urn="urn:getRuntimeDeviceDetails" xmlns:urn1="urn:pHeader"
                  xmlns:urn2="urn:pat">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:getRuntimeDeviceDetailsRequest>
      <urn1:pHeader>
        <urn1:proemionId/>
        <urn1:clientId/>
        (!--Optional:-->
        (<urn1:authenticationKey>urn:uuid:477aba39-755b-4e85-9f26-bca272a3f665</urn1:authenticationKey>)
        <urn1:requestId/>
        (!--Optional:-->
        (<urn1:responseId/>
        (!--Zero or more repetitions:-->
        <urn1:customerKeyValuePair>
          <urn1:key/>
          <urn1:value/>
        </urn1:customerKeyValuePair>)
      </urn1:pHeader>
      <urn2:getRuntimeDeviceDetailsRequest>
        (!--Optional:-->
        (<urn2:imei>355633007604449</urn2:imei>)
      </urn2:getRuntimeDeviceDetailsRequest>
    </urn:getRuntimeDeviceDetailsRequest>
  </soapenv:Body>
</soapenv:Envelope>
```
The Runtime Entities

A getRuntimeDeviceDetails response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns7="urn:pProcessingFault"
  xmlns:ns2="urn:pHeader"
  xmlns:ns3="urn:pbt"
  xmlns:ns4="urn:pat"
  xmlns:ns5="urn:getAllVisibleRuntimeEntities"
  xmlns:ns6="urn:getDevices"
  xmlns:ns7="urn:getRuntimeDeviceDetails"
  xmlns:ns8="urn:pDeviceRead"
  xmlns:ns9="urn:getAllVisibleAdminEntities">
  <ns2:pHeader>
    <ns2:proemionId/>
    <ns2:clientId/>
    <ns2:authenticationKey>urn:uuid:477aba39-755b-4e85-9f26-bca272a3f665</ns2:authenticationKey>
    <ns2:requestId/>
    <ns2:responseId/>
  </ns2:pHeader>
  <ns4:getRuntimeDeviceDetailsResponse>
    <ns3:devicePk>355633007604449</ns3:devicePk>
    <ns3:deviceName>355633007604449</ns3:deviceName>
    <ns3:imei>355633007604449</ns3:imei>
    <ns3:hasGps>false</ns3:hasGps>
    <ns3:deviceGroups/>
    <ns3:controllers/>
    <ns3:machinePk>5396</ns3:machinePk>
    <ns3:machineName>355633007604449</ns3:machineName>
    <ns3:functionSets/>
    <ns3:result>
      <ns3:functionSetPk>101</ns3:functionSetPk>
    </ns3:result>
  </ns4:getRuntimeDeviceDetailsResponse>
</S:Body>
</S:Envelope>
```
Monitoring Devices and Vehicles - Status

The Status Service

Status service can be used with any application that can make an WS-I compliant SOAP request.

The service endpoint is: https://services.proemion.com/proemion-ws/ws-proemion-status/2009/07/16/StatusService

Overview

WSDL Definition

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name space</td>
<td>urn:statusServices</td>
</tr>
<tr>
<td>Binding</td>
<td>StatusServiceBinding</td>
</tr>
<tr>
<td>SOAP Version</td>
<td>SOAP 1.1</td>
</tr>
<tr>
<td>Style</td>
<td>Document</td>
</tr>
<tr>
<td>WS-A version</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Definition Parts

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>statusServices.wsdl</td>
<td>statusServices.wsdl</td>
</tr>
<tr>
<td>pProcessingProblem.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pProcessingProblem.xsd">https://services.proemion.com/2009/07/16/pProcessingProblem.xsd</a></td>
</tr>
<tr>
<td>pStatusGetStatus.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pStatusGetStatus.xsd">https://services.proemion.com/2009/07/16/pStatusGetStatus.xsd</a></td>
</tr>
<tr>
<td>pBaseTypes.xsd</td>
<td><a href="http://services.proemion.com/2009/07/16/pBaseTypes.xsd">http://services.proemion.com/2009/07/16/pBaseTypes.xsd</a></td>
</tr>
<tr>
<td>pHeaderTypes.xsd</td>
<td><a href="http://services.proemion.com/2009/07/16/pHeaderTypes.xsd">http://services.proemion.com/2009/07/16/pHeaderTypes.xsd</a></td>
</tr>
<tr>
<td>pStatusTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pStatusTypes.xsd">https://services.proemion.com/2009/07/16/pStatusTypes.xsd</a></td>
</tr>
</tbody>
</table>


Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
<th>One-Way</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>getStatusByDeviceIds</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>getStatusByMachineIds</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>getDeviceStates</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>getMachineStates</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

Status Service Operations

getStatusByDeviceIds

This operation is deprecated please use getDeviceStates!

getStatusByMachineIds

This operation is deprecated please use getMachineStates!

getDeviceStates

The getDeviceStates operation delivers status information for every item of a given list of devices. It takes a list of device IDs and returns a list of device states, that consists of the device ID and a list of string keys, which represent specific states. For information on the state keys look at this table. The list of state
keys always contains at least state.online or state.offline. If the state of a device cannot be determined, it is put to the list of failed devices. Possible reasons for this are if you have no access to the device or if the device does not exist. This list consists of items with the device ID and the fault reason.

The operation returns a PProcessingFault with the code accessdenied.E00100 if the authenticated user has no access to a machine overview feature. It also returns a PProcessingFault with the code generic.E00017 on any other unhandled error on the server side.

**State Key that can be returned**

<table>
<thead>
<tr>
<th>Key</th>
<th>Represented State</th>
</tr>
</thead>
<tbody>
<tr>
<td>state.offline</td>
<td>The device couldn't be found connected to any server</td>
</tr>
<tr>
<td>state.online</td>
<td>The device is connected to a server</td>
</tr>
<tr>
<td>mode.unknown</td>
<td>The proemion operation mode of the device couldn't be determined</td>
</tr>
<tr>
<td>mode.filetransfer</td>
<td>The device is in filetransfer mode</td>
</tr>
<tr>
<td>mode.realtime</td>
<td>The device is in realtime</td>
</tr>
<tr>
<td>state.realtime.clientconnected</td>
<td>The device is connected to a realtime client</td>
</tr>
<tr>
<td>mode.mixed</td>
<td>The device operates in more than one proemion operation mode in parallel</td>
</tr>
</tbody>
</table>

**getDeviceState Parameter**

The getDeviceState Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devices</td>
<td>List of device IDs</td>
</tr>
</tbody>
</table>

**getDeviceState Examples**

A getDeviceState request

```xml
xmlns:ns0="urn:pStatusGetStatus" xmlns:ns1="urn:pHeader"
xmlns:ns2="urn:ps">  
<SOAP-ENV:Header/>
<SOAP-ENV:Body>
<ns0:getDeviceStates>
<ns1:pHeader>
...
<ns1:authenticationKey>urn:uuid:...</ns1:authenticationKey>
...
</ns1:pHeader>
<ns2:psDeviceStatesRequest>
<ns2:devices>
<ns2:devicePk>123</ns2:devicePk>
<ns2:devicePk>789</ns2:devicePk>
<ns2:devicePk>000</ns2:devicePk>
</ns2:devices>
</ns2:psDeviceStatesRequest>
</ns0:getDeviceStates>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Monitoring Devices and Vehicles - Status

A getDeviceState response

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns6="urn:pHeader"
  xmlns:ns2="urn:pst"
  xmlns:ns3="urn:pProcessingFault"
  xmlns:ns4="urn:pStatusHeartbeat"
  xmlns:ns5="urn:pbt"
  xmlns:ns6="urn:pStatusGetStatus">
  <pHeader>
    ...
  </pHeader>
  <ns2:psDeviceStatesResponse>
    <ns2:deviceStates>
      <ns2:deviceState>
        <ns2:devicePk>789</ns2:devicePk>
        <ns2:states>
          <ns2:state>mode.filetransfer</ns2:state>
          <ns2:state>state.online</ns2:state>
        </ns2:states>
      </ns2:deviceState>
      <ns2:deviceState>
        <ns2:devicePk>123</ns2:devicePk>
        <ns2:states>
          <ns2:state>mode.realtime</ns2:state>
          <ns2:state>state.online</ns2:state>
        </ns2:states>
      </ns2:deviceState>
    </ns2:deviceStates>
    <ns2:failedDevices>
      <ns2:device>
        <ns2:pk>000</ns2:pk>
        <ns2:fault>accessdenied.E00100</ns2:fault>
      </ns2:device>
    </ns2:failedDevices>
  </ns2:psDeviceStatesResponse>
</ns6:getDeviceStatesResponse>
</soap:Body>
</soap:Envelope>
```

getMachineStates

The getMachineStates operation delivers status information for every item of a given list of machines. It takes a list of machine IDs and returns a list of machine states, that consists of the machine ID and a list of device states. For more information on device states have a look at the section for getDeviceStates. The device states list consists of all devices that are assigned to the machine. So you have to decide for your needs how to interpret the states of different devices in a machine. If the state of a machine cannot be determined, it is put to the list of failed machine. Possible reasons for this are if you have no access to the machine or if the machine does not exist. Another example is if the machine has no devices assigned. This list consists of items with the machine ID and the fault reason. The operation returns a PProcessingFault with the code accessdenied.E00100 if the authenticated user has no access to a machine overview feature. It also returns a PProcessingFault with the code generic.E00017 on any other unhandled error on the server side.

The getMachineState Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machines</td>
<td>List of machine IDs</td>
</tr>
<tr>
<td>includeStatusInfo</td>
<td>Boolean flag to include status info into response (optional - default is false)</td>
</tr>
</tbody>
</table>
Monitoring Devices and Vehicles - Status

**getMachineState Examples**

A `getMachineState` request

```xml
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns0="urn:pStatusGetStatus"
  xmlns:ns1="urn:pHeader"
  xmlns:ns2="urn:ps">
  <SOAP-ENV:Header/>

  <SOAP-ENV:Body>
    <ns0:getMachineStates>
      <ns1:pHeader>
        ...
        <ns1:authenticationKey>urn:uuid:...</ns1:authenticationKey>
        ...
      </ns1:pHeader>

      <ns2:psMachineStatesRequest>
        <ns2:includeStatusInfo>true</ns2:includeStatusInfo>
        <ns2:machines>
          <ns2:machinePk>0</ns2:machinePk>
          <ns2:machinePk>09977</ns2:machinePk>
          <ns2:machinePk>09978</ns2:machinePk>
        </ns2:machines>
      </ns2:psMachineStatesRequest>
    </ns0:getMachineStates>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
Monitoring Devices and Vehicles - Status

A getMachineState response

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"

<soap:Body>
<ns6:getMachineStatesResponse xmlns="urn:pHeader"
xmlns:ns2="urn:pst"
xmlns:ns3="urn:pProcessingFault"
xmlns:ns4="urn:pStatusHeartbeat"
xmlns:ns5="urn:pbt"
xmlns:ns6="urn:pStatusGetStatus">

<pHeader>
...
</pHeader>
<ns2:psMachineStatesResponse>
<ns2:machineStates>
<ns2:machineState>
<ns2:statusInfo>STATUS_INFO_STRING</ns2:statusInfo>
<ns2:machinePk>09977</ns2:machinePk>
<ns2:deviceStates>
<ns2:deviceState>
<ns2:devicePk>123</ns2:devicePk>
<ns2:states>
<ns2:state>mode.realtime</ns2:state>
<ns2:state>state.online</ns2:state>
</ns2:states>
</ns2:deviceState>
<ns2:deviceStates>
</ns2:machineState>
<ns2:machineState>
<ns2:machinePk>09978</ns2:machinePk>
<ns2:deviceStates>
<ns2:deviceState>
<ns2:devicePk>353227020865855</ns2:devicePk>
<ns2:states>
<ns2:state>mode.unknown</ns2:state>
<ns2:state>state.online</ns2:state>
</ns2:states>
</ns2:deviceState>
<ns2:failedDevices/>
</ns2:machineState>
<ns2:machineState>
<ns2:machinePk>0</ns2:machinePk>
<ns2:statusInfo>accessdenied.E00100</ns2:statusInfo>
</ns2:machineState>
</ns2:machineStates>
</ns2:psMachineStatesResponse>
</soap:Body>
</soap:Envelope>
```
File Handling (Transferring Files)

The File Services

File service can be used with any application that can make an WS-I compliant SOAP request. The service endpoint is: https://services.proemion.com/proemion-ws/ws-proemion-file2/2011/02/16/FileService

Overview

WSDL Definition

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>urn:fileServices</td>
</tr>
<tr>
<td>Binding</td>
<td>FileServiceBinding</td>
</tr>
<tr>
<td>SOAP Version</td>
<td>SOAP 1.1</td>
</tr>
<tr>
<td>Style</td>
<td>Document</td>
</tr>
<tr>
<td>WS-A version</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Definition Parts

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fileServices.wsdl</td>
<td>fileServices.wsdl</td>
</tr>
<tr>
<td>pProcessingProblem.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pProcessingProblem.xsd">https://services.proemion.com/2009/07/16/pProcessingProblem.xsd</a></td>
</tr>
<tr>
<td>pFile2Types.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pFile2Types.xsd">https://services.proemion.com/2011/02/16/pFile2Types.xsd</a></td>
</tr>
<tr>
<td>pBaseTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pBaseTypes.xsd">https://services.proemion.com/2009/07/16/pBaseTypes.xsd</a></td>
</tr>
<tr>
<td>pHeaderTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pHeaderTypes.xsd">https://services.proemion.com/2009/07/16/pHeaderTypes.xsd</a></td>
</tr>
<tr>
<td>pSearchFiles.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pSearchFiles.xsd">https://services.proemion.com/2011/02/16/pSearchFiles.xsd</a></td>
</tr>
<tr>
<td>pSearchDownloadFiles.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pSearchDownloadFiles.xsd">https://services.proemion.com/2011/02/16/pSearchDownloadFiles.xsd</a></td>
</tr>
<tr>
<td>pSearchUploadFiles.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pSearchUploadFiles.xsd">https://services.proemion.com/2011/02/16/pSearchUploadFiles.xsd</a></td>
</tr>
<tr>
<td>pMarkFetchedFile.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pMarkFetchedFile.xsd">https://services.proemion.com/2011/02/16/pMarkFetchedFile.xsd</a></td>
</tr>
<tr>
<td>pDownloadFiles.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pDownloadFiles.xsd">https://services.proemion.com/2011/02/16/pDownloadFiles.xsd</a></td>
</tr>
<tr>
<td>pDeleteUploadFiles.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pDeleteUploadFiles.xsd">https://services.proemion.com/2011/02/16/pDeleteUploadFiles.xsd</a></td>
</tr>
<tr>
<td>pUploadFileToMachine.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pUploadFileToMachine.xsd">https://services.proemion.com/2011/02/16/pUploadFileToMachine.xsd</a></td>
</tr>
<tr>
<td>pUploadFirmwareToMachine.xsd</td>
<td><a href="https://services.proemion.com/2011/02/16/pUploadFirmwareToMachine.xsd">https://services.proemion.com/2011/02/16/pUploadFirmwareToMachine.xsd</a></td>
</tr>
<tr>
<td>pUploadDbsConfiguration.xsd</td>
<td><a href="https://services.proemion.com/2013/02/14/pUploadDbsConfiguration.xsd">https://services.proemion.com/2013/02/14/pUploadDbsConfiguration.xsd</a></td>
</tr>
</tbody>
</table>

Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
<th>One-Way</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>downloadFile</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>markFetchedFile</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>searchFiles</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>searchDownloadFiles</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>searchUploadFiles</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>deleteUploadFiles</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>uploadFileToMachine</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>
File Handling (Transferring Files)

Operations (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
<th>One-Way</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>uploadFirmwareToMachine</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
<tr>
<td>uploadDbsConfigToMachine</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

File Service Operations

DownloadFile

**DownloadFile Parameter**

To be done.

The `downloadFile` Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>??????</td>
<td></td>
</tr>
</tbody>
</table>

DownloadFile Examples

A `changeDeviceModeTo` request

```xml
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:urn="urn:pDownloadFile"
 xmlns:urn1="urn:pHeader"
 xmlns:urn2="urn:pft">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:downloadFile>
   <urn1:pHeader>
    ...
    <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
    ...
   </urn1:pHeader>
   <urn2:pfDownloadFileRequest>
    <urn2:fileTransferPk>1234</urn2:fileTransferPk>
   </urn2:pfDownloadFileRequest>
  </urn:downloadFile>
 </soapenv:Body>
</soapenv:Envelope>
```
File Handling (Transferring Files)

A `changeDeviceModeTo` response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
>S:Body>
<ns9:downloadFileResponse
xmlns="urn:pProcessingFault"
xmlns:ns2="urn:pHeader"
xmlns:ns3="urn:pft"
xmlns:ns4="urn:pbt"
xmlns:ns5="urn:pSearchFiles"
xmlns:ns6="urn:pSearchDownloadFiles"
xmlns:ns7="urn:pSearchUploadFiles"
xmlns:ns8="urn:pMarkFetchedFile"
xmlns:ns9="urn:pDownloadFile">
<ns2:pHeader>
...<ns2:authenticationKey>urn:uuid:...</ns2:authenticationKey>
...
</ns2:pHeader>
<ns3:pfDownloadFileResponse>
<ns3:filePk>1234</ns3:filePk>
<ns3:content>VQANA/gAHEkxYnJpem9sIEoxOTM5IENvbmZpZ3VyYXRpb24BAQAR/wzwBADxsGbIwD/sUt6hKwBAQAR/zwAwDckDT///+C/0t6hKwBAQAR/xj+vwx0VX1b2zn///0t6hKwBAQAR/zwAwDcfcZv/QAR/xj+vwtsVH58dW7///0t6hOoBAQAR/xj+8gA0AxwE2wX///0t6hOoBAQAR/wzwbABDxuLg31wD/ueT6hOoBAQAR/zwAwDcpTv///+T/0t6hOqUxKo=</ns3:content>
<ns3:fileName>00018797</ns3:fileName>
<ns3:fileSuffix>clf</ns3:fileSuffix>
<ns3:fileSize>5120</ns3:fileSize>
<ns3:direction>download</ns3:direction>
</ns3:pfDownloadFileResponse>
</ns9:downloadFileResponse>
</S:Body>
</S:Envelope>
```

`markFetchedFile`

The `markFetchedFile` Parameter

To be done.

The `markFetchedFile` Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>??????</td>
<td></td>
</tr>
</tbody>
</table>
File Handling (Transferring Files)

**markFetchedFile Examples**

A *markFetchedFile* request

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:urn="urn:pMarkFetchedFile"
  xmlns:urn1="urn:pHeader"
  xmlns:urn2="urn:pft">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:markFetchedFile>
      <urn1:pHeader>
        ...
        <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
        ...
      </urn1:pHeader>
      <urn2:pfMarkFetchedFileRequest>
        <urn2:fileTransferPk>1234</urn2:fileTransferPk>
      </urn2:pfMarkFetchedFileRequest>
    </urn1:pHeader>
    </urn2:pfMarkFetchedFileRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A *markFetchedFile* response

```xml
    <ns8:markFetchedFileResponse
      xmlns="urn:pProcessingFault"
      xmlns:ns2="urn:pHeader"
      xmlns:ns3="urn:pft"
      xmlns:ns4="urn:pbt"
      xmlns:ns5="urn:pSearchFiles"
      xmlns:ns6="urn:pSearchDownloadFiles"
      xmlns:ns7="urn:pSearchUploadFiles"
      xmlns:ns8="urn:pMarkFetchedFile"
      xmlns:ns9="urn:pDownloadFile">
      <ns2:pHeader>
        ...
        <ns2:authenticationKey>urn:uuid:...</ns2:authenticationKey>
        ...
      </ns2:pHeader>
    </ns8:markFetchedFileResponse>
  </S:Body>
</S:Envelope>
```

**SearchFiles**

**SearchFiles Parameter**

Lists files, uploaded or downloaded by a machine, matching the given parameters.

The parameter direction has to be **download** or **upload**, if you choose **unknown** it will be used like **download**

**The searchFiles Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machineName</td>
<td>Look for machine name containing given String. (case insensitive)</td>
</tr>
<tr>
<td>status</td>
<td>List complete or incomplete files</td>
</tr>
<tr>
<td>fileName</td>
<td>Look for file name containing given String. (case insensitive)</td>
</tr>
<tr>
<td>fileSuffix</td>
<td>Look for file suffix containing given String. (case insensitive)</td>
</tr>
</tbody>
</table>
File Handling (Transferring Files)

The searchFiles Parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ignoreFetchedFiles</td>
<td>Ignore files that are marked as fetch by webservice</td>
</tr>
<tr>
<td>from</td>
<td>Define a date range beginning with given date. If no timezone offset is given, it is interpreted as UTC</td>
</tr>
<tr>
<td>to</td>
<td>Define a date range ending with given date. If no timezone offset is given, it is interpreted as UTC</td>
</tr>
<tr>
<td>machinePk</td>
<td>ID of a specific machine</td>
</tr>
<tr>
<td>direction</td>
<td>List upload or download files</td>
</tr>
<tr>
<td>pageSize</td>
<td>Define the number of search results to be listed</td>
</tr>
<tr>
<td>page</td>
<td>The result set corresponding to the pageSize to be returned.</td>
</tr>
</tbody>
</table>

SearchFiles Examples

A searchFiles request

```xml
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmns:urn="urn:pSearchFiles"
xmns:urn1="urn:pHeader"
xmns:urn2="urn:pft">
<soapenv:Header/>
<soapenv:Body>
<urn:searchFilesRequest>
<urn1:pHeader>
...
<urn1:authenticationKey>urn:uuid:...<urn1:authenticationKey>
...
</urn1:pHeader>
<urn2:pfSearchFilesRequest>
<urn2:machineName xsi:nil="true"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
<urn2:status>complete</urn2:status>
<urn2:fileName xsi:nil="true"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
<urn2:fileSuffix xsi:nil="true"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
<urn2:ignoreFetchedFiles>true</urn2:ignoreFetchedFiles>
<urn2:from xsi:nil="true"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
<urn2:to xsi:nil="true"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
<urn2:machinePk xsi:nil="true"
xmns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
<urn2:direction>download</urn2:direction>
<urn2:pageSize>10</urn2:pageSize>
<urn2:page>0</urn2:page>
<urn2:pfSearchFilesRequest>
</urn2:searchFilesRequest>
</urn1:searchFilesRequest>
</soapenv:Body>
</soapenv:Envelope>
```
File Handling (Transferring Files)

A searchFiles response

```xml
<envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/"
<body>
<searchFilesResponse
xmlns="urn:pProcessingFault"
xmlns="urn:pHeader"
xmlns="urn:pft"
xmlns="urn:pbt"
xmlns="urn:pSearchFiles"
xmlns="urn:pSearchDownloadFiles"
xmlns="urn:pSearchUploadFiles"
xmlns="urn:pMarkFetchedFile"
xmlns="urn:pDownloadFile">
<pHeader>
...
</pHeader>
<pfSearchFilesResponse>
<count>1</count>
<pageCount>1</pageCount>
<pageSize>10</pageSize>
<files>
<result>
<filePk>1234</filePk>
<machinePk>5678</machinePk>
<fileName>12345678</fileName>
<fileSuffix>clf</fileSuffix>
<fileSize>5120</fileSize>
<fileFetched>false</fileFetched>
<completed>true</completed>
<dc>2010-02-16T15:31:21.762+01:00</dc>
<dm>2010-03-15T12:57:24.026+01:00</dm>
<direction>download</direction>
</result>
...
</files>
</pfSearchFilesResponse>
</searchFilesResponse>
</body>
</envelope>
```

SearchDownloadFiles

This works exactly like searchFiles, except it predefines the direction parameter to download. Thus if you give the direction parameter, your choice will be ignored.
File Handling (Transferring Files)

**SearchDownloadFiles Examples**

A searchDownloadFiles request

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:urn="urn:pSearchFiles"
  xmlns:urn1="urn:pHeader"
  xmlns:urn2="urn:pft">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:searchDownloadFilesRequest>
      <urn1:pHeader>
        ...
        <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
        ...
      </urn1:pHeader>
      <urn2:pfSearchFilesRequest>
        ...
      </urn2:pfSearchFilesRequest>
    </urn:searchDownloadFilesRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

**searchUploadFiles**

This works exactly like searchFiles, except if predefines the direction parameter to upload. Thus if you give the direction parameter, your choice will be ignored.

**searchUploadFiles Examples**

A searchUploadFiles request

```xml
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
  xmlns:urn="urn:pSearchFiles"
  xmlns:urn1="urn:pHeader"
  xmlns:urn2="urn:pft">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:searchUploadFilesRequest>
      <urn1:pHeader>
        ...
        <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
        ...
      </urn1:pHeader>
      <urn2:pfSearchFilesRequest>
        ...
      </urn2:pfSearchFilesRequest>
    </urn:searchUploadFilesRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

**uploadFileToMachine**

**uploadFileToMachine Parameter**

Uploads a file to the device that is attached with the machine.

*If there are more than one devices attached to the machine than this method is not working and will throw an exception.*
The searchFiles Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machinePk</td>
<td>The primary key of the machine.</td>
</tr>
<tr>
<td>fileName</td>
<td>The file name (8 characters).</td>
</tr>
<tr>
<td>fileSuffix</td>
<td>The file suffix (3 characters).</td>
</tr>
<tr>
<td>fileSize</td>
<td>The file size.</td>
</tr>
<tr>
<td>content</td>
<td>The byte content of the file.</td>
</tr>
</tbody>
</table>

uploadFileToMachine Examples

A uploadFileToMachine request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:urn="urn:pUploadFileToMachine" xmlns:urn1="urn:pHeader"
 xmlns:urn2="urn:pft">
 <soapenv:Header/>
 <soapenv:Body>
  <urn:uploadFileToMachine>
   <urn1:pHeader>
   ...
   <urn1:authenticationKey>
    urn:uuid:1bb63c82-7227-447d-bcd9-7adcef14878g
   </urn1:authenticationKey>
   ...
   <urn1:pHeader>
    <urn2:pfUploadFileToMachineRequest>
     <urn2:machinePk>1234</urn2:machinePk>
     <urn2:file>
      <urn2:fileName>12345678</urn2:fileName>
      <urn2:fileSuffix>clf</urn2:fileSuffix>
      <urn2:fileSize>563</urn2:fileSize>
      <urn2:content>byte content</urn2:content>
     </urn2:file>
    </urn2:pfUploadFileToMachineRequest>
   </urn1:pHeader>
  </urn:uploadFileToMachine>
</soapenv:Body>
</soapenv:Envelope>
```

A uploadFileToMachine response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
 <S:Body>
  <ns13:uploadFileToMachineResponse xmlns="urn:pHeader"
   xmlns:ns2="urn:pft"
   xmlns:ns13="urn:pUploadFileToMachine" >
   <pHeader>
   ...
   </pHeader>
  <ns2:pfUploadFileToMachineResponse>
   <ns2:acknowledge>ACK</ns2:acknowledge>
  </ns2:pfUploadFileToMachineResponse>
 </ns13:uploadFileToMachineResponse>
 </S:Body>
</S:Envelope>
```

deleteUploadFiles
File Handling (Transferring Files)

**deleteUploadFiles Parameters**

Deletes all files for the submitted primary keys. This service returns a list of failed files.

**The deleteUploadFiles Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filePks</td>
<td>A list of file primary keys.</td>
</tr>
</tbody>
</table>

**deleteUploadFiles Examples**

A deleteUploadFiles request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:pDeleteUploadFiles" xmlns:urn1="urn:pHeader" xmlns:urn2="urn:pft">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:deleteUploadFiles>
      <urn1:pHeader>
        ...
      <urn1:authenticationKey>urn:uuid:55553dc1-9c8e-42e7-9700-15e267cbe5a5</urn1:authenticationKey>
        ...
      </urn1:pHeader>
      <urn2:pfDeleteUploadFilesRequest>
        <urn2:files>
          <urn2:filePks>601713</urn2:filePks>
          <urn2:filePks>601714</urn2:filePks>
          <urn2:filePks>601715</urn2:filePks>
        </urn2:files>
      </urn2:pfDeleteUploadFilesRequest>
    </urn:deleteUploadFiles>
  </soapenv:Body>
</soapenv:Envelope>
```

A deleteUploadFiles response

```xml
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns15:deleteUploadFilesResponse xmlns="urn:pProcessingFault" xmlns:ns2="urn:pHeader" xmlns:ns3="urn:pft" xmlns:ns15="urn:pDeleteUploadFiles">
      <ns2:pHeader>
        ...
      </ns2:pHeader>
      <ns3:pfDeleteUploadFilesResponse>
        <ns3:failedFilePks>
          <ns3:filePks>601714</ns3:filePks>
          <ns3:filePks>601715</ns3:filePks>
        </ns3:failedFilePks>
      </ns3:pfDeleteUploadFilesResponse>
    </ns15:deleteUploadFilesResponse>
  </S:Body>
</S:Envelope>
```

**uploadFirmwareToMachine**
File Handling (Transferring Files)

**uploadFirmwareToMachine Parameter**

Uploads a firmware file to a machine. Be advised that only one device can be attached to the machine. Else you get an exception.

**The uploadFirmwareToMachine Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machinePk</td>
<td>The primary key of the machine.</td>
</tr>
<tr>
<td>fileName</td>
<td>The file name (8 characters).</td>
</tr>
<tr>
<td>fileSuffix</td>
<td>The file suffix (3 characters).</td>
</tr>
<tr>
<td>fileSize</td>
<td>The file size.</td>
</tr>
<tr>
<td>content</td>
<td>The byte content of the file.</td>
</tr>
</tbody>
</table>

**uploadFirmwareToMachine Examples**

A `uploadFirmwareToMachine` request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:urn="urn:pUploadFirmwareToMachine" xmlns:urn1="urn:pHeader"
 xmlns:urn2="urn:pft">
 <soapenv:Header/>
 <soapenv:Body>
 <urn:uploadFirmwareToMachine>
 <urn1:pHeader>
 <urn1:authenticationKey>
 urn:uuid:ee550791-62bd-404f-aa38-5ca8f7078af2h
 </urn1:authenticationKey>
 </urn1:pHeader>
 <urn2:pfUploadFirmwareToMachineRequest>
 <urn2:machinePk>1234</urn2:machinePk>
 <urn2:firmwareFile>
 <urn2:fileName>firmware</urn2:fileName>
 <urn2:fileSuffix>clf</urn2:fileSuffix>
 <urn2:fileSize>573</urn2:fileSize>
 <urn2:content>byte content</urn2:content>
 </urn2:firmwareFile>
 </urn2:pfUploadFirmwareToMachineRequest>
 </urn:uploadFirmwareToMachine>
</soapenv:Body>
</soapenv:Envelope>
```

**A uploadFirmwareToMachine response**

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/
 xmlns:urn="urn:pUploadFirmwareToMachine">
 <soapenv:Body>
 <ns15:uploadFirmwareToMachineResponse xmlns="urn:pHeader"
 xmlns:ns2="urn:pft"
 xmlns:ns15="urn:pUploadFirmwareToMachine">
 <pHeader>
 ...
 </pHeader>
 <ns2:pfUploadFirmwareToMachineResponse>
 <ns2:acknowledge>ACK</ns2:acknowledge>
 </ns2:pfUploadFirmwareToMachineResponse>
 </ns15:uploadFirmwareToMachineResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

**uploadDbsConfigToMachine**
File Handling (Transferring Files)

**uploadDbsConfigToMachine Parameters**

Uploads all the Dashboard Configurations associated to a machine. Configurations are selected based on the granted access of the user.

**The uploadDbsConfigToMachine Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>machinePk</td>
<td>The primary key of the machine.</td>
</tr>
</tbody>
</table>

**uploadDbsConfigToMachine Examples**

A `uploadDbsConfigToMachine` request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:pUploadDbsConfiguration"
xmlns:urn1="urn:pHeader">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:dbsConfigRequest>
      <urn1:pHeader>
        <urn1:authenticationKey>urn:uuid:5e21ae95-60cc-4275-ab80-fa71ced2ec3f</urn1:authenticationKey>
        <urn1:requestId/>
        <urn1:responseId/>
        <urn1:customerKeyValuePair>
          <urn1:key/>
          <urn1:value/>
        </urn1:customerKeyValuePair>
      </urn1:pHeader>
      <urn:machinePk>23002</urn:machinePk>
    </urn:dbsConfigRequest>
  </soapenv:Body>
</soapenv:Envelope>
```
File Handling (Transferring Files)

**A `uploadDbsConfigToMachine` response**

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:pHeader="urn:pHeader"
    xmlns:pUploadFirmwareToMachine="urn:pUploadFirmwareToMachine"
    xmlns:pUploadFilesToMachine="urn:pUploadFilesToMachine">
  <soap:Body>
    <ns4:dbsConfigResponse xmlns="urn:pHeader">
      <ns4:dbsConfigurationList>
        <ns2:dbsName>test3</ns2:dbsName>
        <ns2:configFile>
          <ns5:fileName>pro_geronimo-web.xml</ns5:fileName>
          <ns5:fileExtension xsi:nil="true"/>
          <ns5:direction xsi:nil="true"/>
          <ns5:content>PD94bWwgdmVyc2...=gdm</ns5:content>
          <ns5:dateOfCreation xsi:nil="true"/>
          <ns5:dateOfModification xsi:nil="true"/>
        </ns2:configFile>
        <ns2:dbsMedias>
          <ns2:dbsMediaType>image/jpeg</ns2:dbsMediaType>
          <ns2:dbsImage>
            <ns2:fileSuffix xsi:nil="true"/>
            <ns2:fileName xsi:nil="true"/>
            <ns2:fileSize>620888</ns2:fileSize>
            <ns2:content>/9j/4AAQSk....=Ssdw</ns2:content>
          </ns2:dbsImage>
        </ns2:dbsMedias>
        <ns2:comment>this is test 3</ns2:comment>
      </ns4:dbsConfigurationList>
    </ns4:dbsConfigResponse>
  </soap:Body>
</soap:Envelope>
```
Geo Mapping Service

`getLastKnownPositionForObject`

**`getLastKnownPositionForObject Request`**

GetLastKnownPositionForObject can be used with any application that can make a WS-I compliant SOAP request.

The service endpoint is: https://services.proemion.com/proemion-ws/ws-proemion-mapping/2009/01/15/MappingService?wsdl

**`getLastKnownPositionForObject Parameter`**

The `getLastKnownPositionForObject` Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>objectID</code></td>
<td>The unique identifier of the device (IMEI).</td>
</tr>
</tbody>
</table>

**Sample `getLastKnownPositionForObject` Request**

**`getLastKnownPositionForObject Response`**

In response to a `getLastKnownPositionForObject` request, Proemion will return the position for the one matching object.

**Sample `getLastKnownPositionForObject` Response**

**`getLastKnownPositionForObject Error Codes`**

Error codes are returned by a SOAP fault only. The standard SOAP fault for Proemion services is `pProcessingFault`.

See the documentation of `pProcessingFault`, to see on how the error codes are returned.

**The `getLastKnownPositionForObject` Error Codes**

<table>
<thead>
<tr>
<th>Error Number</th>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
</table>
The Devices Service can be used with any application that can make a WS-I compliant SOAP request.

The service endpoint is: 1)

Overview

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Namespace</td>
<td>urn: pDeviceStatus</td>
</tr>
<tr>
<td>Binding</td>
<td>DeviceServiceBinding</td>
</tr>
<tr>
<td>SOAP Version</td>
<td>SOAP 1.1</td>
</tr>
<tr>
<td>Style</td>
<td>Document</td>
</tr>
<tr>
<td>WS-A version</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Definition Parts

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pChangeDeviceModeTo.xsd</td>
<td><a href="https://services.proemion.com/2012/03/13/pChangeDeviceModeTo.xsd">https://services.proemion.com/2012/03/13/pChangeDeviceModeTo.xsd</a></td>
</tr>
<tr>
<td>pHeaderTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pHeaderTypes.xsd">https://services.proemion.com/2009/07/16/pHeaderTypes.xsd</a></td>
</tr>
<tr>
<td>pBaseTypes.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pBaseTypes.xsd">https://services.proemion.com/2009/07/16/pBaseTypes.xsd</a></td>
</tr>
<tr>
<td>pDeviceTypes.xsd</td>
<td><a href="https://services.proemion.com/2012/03/13/pDeviceTypes.xsd">https://services.proemion.com/2012/03/13/pDeviceTypes.xsd</a></td>
</tr>
<tr>
<td>pProcessingProblem.xsd</td>
<td><a href="https://services.proemion.com/2009/07/16/pProcessingProblem.xsd">https://services.proemion.com/2009/07/16/pProcessingProblem.xsd</a></td>
</tr>
</tbody>
</table>

Operations

<table>
<thead>
<tr>
<th>Name</th>
<th>Use</th>
<th>One-Way</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeDeviceModeTo</td>
<td>Literal</td>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

Device Service Operations

**changeDeviceModeTo**

**changeDeviceModeTo Parameter**

With the webservice method `changeDeviceModeTo` it is possible to switch the device mode from the selected device.

**The changeDeviceModeTo Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>deviceId</td>
<td>The object's (vehicle) unique identifier.</td>
</tr>
<tr>
<td>deviceMode</td>
<td>The object's (vehicle) name (note: that this is not unique)</td>
</tr>
</tbody>
</table>

1) https://services.proemion.com/proemion-ws/ws-proemion-device/2012/03/13/DeviceService
Devices Service

changeDeviceModeTo Examples

A changeDeviceModeTo request

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:changeDeviceModeTo"
xmlns:urn1="urn:pHeader"
xmlns:urn2="urn:pdt">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:changeDeviceModeToRequest>
      <urn1:pHeader>
        <urn1:proemionId></urn1:proemionId>
        <urn1:clientId></urn1:clientId>
        <urn1:authenticationKey>urn:uuid:...</urn1:authenticationKey>
        <urn1:requestId></urn1:requestId>
        <urn1:responseId></urn1:responseId>
        <!--Zero or more repetitions:-->
        <urn1:customerKeyValuePair>
          <urn1:key></urn1:key>
          <urn1:value></urn1:value>
        </urn1:customerKeyValuePair>
      </urn1:pHeader>
      <urn2:pdChangeDeviceModeToRequest>
        <urn2:devicePk>000000000000001</urn2:devicePk>
        <urn2:deviceMode>realtime</urn2:deviceMode>
      </urn2:pdChangeDeviceModeToRequest>
    </urn:changeDeviceModeToRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

A changeDeviceModeTo response

```xml
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns5:changeDeviceModeToResponse xmlns="urn:pHeader"
xmlns:ns2="urn:pdt"
xmlns:ns3="urn:pProcessingFault"
xmlns:ns4="urn:pbt"
xmlns:ns5="urn:changeDeviceModeTo">
      <pHeader>
        <proemionId/>
        <clientId/>
        <authenticationKey>urn:uuid:...</authenticationKey>
        <requestId/>
        <responseId/>
        <customerKeyValuePair>
          <key/>
          <value/>
        </customerKeyValuePair>
        <customerKeyValuePair>
          <key>version</key>
          <value>1.3.1</value>
        </customerKeyValuePair>
      </pHeader>
      <ns2:pdChangeDeviceModeToResponse>
        <ns2:acknowledge>ACK</ns2:acknowledge>
      </ns2:pdChangeDeviceModeToResponse>
    </ns5:changeDeviceModeToResponse>
  </soap:Body>
</soap:Envelope>
```
Service and support

Telematics service and support

For further information see our DPS website: http://www.danfoss.com or contact our support team plus1helpdesk@danfoss.com

The most recent WS driver, WS System tools and documentation are available for download from the DPS website Telematics section http://powersolutions.danfoss.com/solutions/telematics/.
Danfoss Power Solutions is a global manufacturer and supplier of high-quality hydraulic and electric components. We specialize in providing state-of-the-art technology and solutions that excel in the harsh operating conditions of the mobile off-highway market as well as the marine sector. Building on our extensive applications expertise, we work closely with you to ensure exceptional performance for a broad range of applications. We help you and other customers around the world speed up system development, reduce costs and bring vehicles and vessels to market faster.

Danfoss Power Solutions – your strongest partner in mobile hydraulics and mobile electrification.

Go to www.danfoss.com for further product information.

We offer you expert worldwide support for ensuring the best possible solutions for outstanding performance. And with an extensive network of Global Service Partners, we also provide you with comprehensive global service for all of our components.

Products we offer:
- DCV directional control valves
- Electric converters
- Electric machines
- Electric motors
- Hydrostatic motors
- Hydrostatic pumps
- Orbital motors
- PLUS+1® controllers
- PLUS+1® displays
- PLUS+1® joysticks and pedals
- PLUS+1® operator interfaces
- PLUS+1® sensors
- PLUS+1® software
- PLUS+1® software services, support and training
- Position controls and sensors
- PVG proportional valves
- Steering components and systems
- Telematics

Danfoss can accept no responsibility for possible errors in catalogues, brochures and other printed material. Danfoss reserves the right to alter its products without notice. This also applies to products already on order provided that such alterations can be made without subsequent changes being necessary in specifications already agreed.

All trademarks in this material are property of the respective companies. Danfoss and the Danfoss logotype are trademarks of Danfoss A/S. All rights reserved.