Inbound SDK

for

Version 4.5

Table of Contents

1 Introduction

1.1 Document Purpose

The document is to provide understanding of the basic Inbound API goals, its methods and relevant SOAP transactions as well as to ensure successful interaction of the Client-developed applications and the ETAdirect application server via the Inbound Interface.

1.2 Scope of the Document

This document primarily describes the API methods used by the ETAdirect Inbound API to transfer information (send requests and accept responses) with external systems. It also gives an overview of how ETAdirect Inbound API works.

1.3 Target Audience

The document is intended for developers and programmers working with the ETAdirect Inbound API in order to integrate ETAdirect with external systems.

1.4 Glossary

Term	Explanation
Activate Queue	Start the work day
Activity	Entity of the ETAdirect system that represents any time consuming activity of the resource
Activity Status	Dynamic value that corresponds to the state of particular activity execution
API	Application Programming Interface – a particular set of rules and specifications that software programs follow to communicate and interact with each other
Bucket	Entity appearing on the resource tree which can contain resources of a defined type and be assigned activities
Company	1) Legal entity, using ETAdirect 2) Entity that represents a Client in ETAdirect system; company is created by TOA technologies during the process of implementation
Customer	End-customer, entity that benefits from the activity
Delivery window	Statistically calculated time period within which a provider is expected to start the activity
ETA	Predicted time at which a resource will arrive at an appointment and start an activity, calculated dynamically for current and historical data
Inventory	Equipment that can be installed or deinstalled during an activity
Group	Feature on the resource tree identifying a particular type of resource
ISO 8601 format	See http://en.wikipedia.org/wiki/ISO_8601
Linked Activities	Two separate activities related so that the completion or start of one is dependent on the completion or start of the other
Mass (activity)	Activity involving 2 or more resources, different from teamwork

Term	Explanation
Not Ordered	Activity with an unspecified order of execution in a queue, so that it can be executed at any time during the working day. Not-ordered activities do not have defined ETAs or delivery windows
Not-scheduled	Activity not assigned to a specific date
Ordered	Activity with a defined place in a queue, which must be performed at a specified time of day. The order of activities can be changed; ordered activities can be changed to not-ordered activities, vice-versa
Property	Field and field value, assigned to an entity in ETAdirect (to user, resource, activity or inventory). There are fields and custom properties
Preferred Provider	Functionality than enables choice of required, preferred, and forbidden resource to be assigned a specific activity
Queue (Route)	List of activities assigned to a resource for a specific date, or a list of non-scheduled activities assigned to a resource
Resource	Element in the resource tree representing a defined company asset
Resource External ID	Company-unique key used to identify a specific resource
Resource Tree	Hierarchy of company resources, showing “parent-child” relationships
Service Window	Time frame expected by the customer for an activity as scheduled by the company
SLA window	Interval of time (that may involve a range of dates) within which certain work has to be performed according to the Service Level Agreement
SOAP	Lightweight protocol for exchange of information in a decentralized, distributed environment
SOAP 1.1	See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
SOAP Interface	Interface used to receive requests and return responses via SOAP
SOAP Client Application	Application running at the Client's site and providing interaction with ETAdirect server via SOAP
Teamwork	Activity type feature which defines which activities can be performed by a Team
Technician (Resource)	Person who performs work at the customer’s premises (the one who provides the service)
Time Slot	1) Fixed service window, defined with a name and label, specifying when certain types of activities can be performed 2) Service Window (if the activity type does not support time slots)
Travel Area	Entity that corresponds to a set of Work Zones
User	1) A person using ETAdirect 2) Entity used for authentication and authorization purpose- the one that is used to grant access to ETAdirect for people or external software
Work Type	Property that defines a company-specific type of activity (customer activity: install, upgrade, etc.; internal activity: lunch, vehicle maintenance, etc.; teamwork)
Work Zone	Defined geographical area in which a resource can perform an activity

2 Inbound Interface Overview

Inbound Interface is implemented as SOAP function for the following purposes:

- set activities for a specific day for all resources or resource groups in ETAdirect
- add new activities to ETAdirect
- update/cancel/reassign/reschedule/ activities in ETAdirect
- change an activity status in ETAdirect
- delete activities from ETAdirect
- set inventory for resources in ETAdirect
- update or delete specific inventory in ETAdirect

The Inbound API uploads data from external system to ETAdirect. The upload can differ in the object uploaded: activities or resource inventories; and in the scale of upload: full or incremental.

2.1 Incremental Upload

The Incremental Upload transaction contains a list of 'command' items that enable processing of specified activities and resource inventory. Each command is processed individually, and failure of one command does not affect other commands in a transaction, except when specifically noted otherwise.

2.1.1 Activity-Related Commands

Activity-related commands process activities in ETAdirect via the Inbound API.
Activities to be processed are defined with the values of key field defined in the request.

The activity related commands:

- 'update_activity' (this command has a deprecated alias: 'update_appointment')
- 'cancel_activity ' (this command has a deprecated alias: 'cancel_appointment')
- 'start_activity'
- 'complete_activity'
- 'notdone_activity'
- 'suspend_activity'
- 'delete_activity'

2.1.2 Inventory-Related Commands

- 'set_inventory'
- 'update_inventory'
- 'delete_inventory'

2.2 Full Upload

There are two types of full upload: Full Activity Upload and Full Inventory Upload. They are described as follows.

Full Upload transaction contains a list of 'provider' items. Each resource has – depending on whether it is an activity or inventory upload – a list of activities or a list of inventories that are to be uploaded. Each activity/inventory is processed individually. Failure of one item does not affect other items, except when specifically noted otherwise.

2.2.1 Full Activity Upload

Full activity Upload replaces/updates all activities in ETAdirect for the given date with the activities sent in a transaction ('update_activity' command is performed for all activities for the given date). All activities for the upload date that have not been processed by the Full Activity Upload transaction will be deleted or cancelled. Thus:

- If a new activity has been inserted, it is not deleted/cancelled
- If an existing activity's properties or inventories have been updated, it is not deleted/cancelled
- If an activity has been moved from resource to resource, it is not deleted/cancelled
- All other activities that were scheduled for the upload date are deleted or cancelled.

2.2.2 Full Inventory Upload

Full Inventory Upload replaces/updates all inventories in ETAdirect with the inventories sent in a transaction ('update_inventory command is performed for all inventory of the specified resource). Since inventory does not depend on the specific date, it is replaced for all dates since the transaction.

All inventories that have not been processed by the Full Inventory Upload transaction will be deleted or cancelled, thus:

- If a new inventory has been inserted, then it is not deleted
- If an existing inventory's properties have been updated, then it is not deleted
- If an inventory has been moved from resource to resource, then it is not deleted.

All other inventories are deleted.

3 Implementation Guidelines

These guidelines are to help developers pass API Certification.

3.1 API Certification

Please note that after initial development and integration phase, your software will undergo certification, to ensure that the API is utilized in an optimal way.
During the certification, TOA may request additional changes to be made to API client, usually concerning performance.
Below are the client implementation guidelines, which should will help the API client software pass certification.

3.2 Send Multiple Commands per Incremental Upload

Inbound API allows sending multiple commands in the same SOAP request.
The SOAP Client application therefore should not send each command individually but should accumulate them for some period of time (e.g. several seconds) and then send the whole batch.

Sending multiple commands per request dramatically increases performance, in part because the network latency overhead can be as high as 0.5 seconds per request, so without batches sending more than 1-2 commands per second, per client thread will not be possible.
As a general rule, a SOAP request containing one activity is processed in approximately the same time as the request containing multiple activities.

Example SOAP request containing 5 commands:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:toatech:InboundInterface:1.0">
 <soapenv:Header/>
 <soapenv:Body>
 <urn:inbound_interface_request>
 <!-- 'user' and 'head' nodes skipped -->

 <data>
 <commands>
 <command> <!-- command payload --> </command>
 <command> <!-- command payload --> </command>
 <command> <!-- command payload --> </command>
 <command> <!-- command payload --> </command>
 <command> <!-- command payload --> </command>
 </commands>
 </data>
 </urn:inbound_interface_request>
 </soapenv:Body>
 </soapenv:Envelope>
```

4 Inbound API Structures

Briefly, the Inbound Interface uploads and processes properties of activities and resource inventory in ETAdirect. This section contains the detailed description of the entities processed by the Inbound API and corresponding structures used in the interface requests and/or responses.

4.1 Properties, Files and Fields

Property is a variable associated with an ETAdirect entity (resource, activity or inventory). This is much like a field of an object in most programming languages. The Inbound API deals with properties of activity, further referred to as 'activity property', properties of resource (actually the only resource property is 'inventory') and properties of inventory, further referred to as 'inventory property'. Every property has a string label by which it is uniquely identified – it can be thought of as a field name and a value.

If there is a property of an entity in ETAdirect, all such entities have this property, and the Inbound Interface can read this property, write to it and make conditional decisions based on the property value.

If a file is assigned to an entity, it is called a '**file property**' or '**file**'. Some properties are initially defined in ETAdirect and others are customer specific and created in the course of implementation. Properties initially defined in ETAdirect are addressed as '**fields**'.

4.1.1 Key Fields

Key fields are specified in the request and used in the Inbound API to indicate a specific activity or specific resource inventory. Key fields for activity are defined in the request 'head/appointments/keys' and key fields for inventory are defined in the request 'head/inventory/keys'.

4.1.1.1 Activity Key Fields

The fields listed in the 'head/appointments/keys' form a composite key that is used to determine, whether an activity exists in ETAdirect and should be processed, or whether it does not exist and a new activity should be created and then processed. Therefore, these fields must be present and non-empty for every activity in the transaction, otherwise the activity will be rejected.

The valid values of 'appointment/keys' are 'appt_number', 'customer_number' and 'name'.

- **'head/appointment/keys' structure example:**

```
<appointment>
        <keys>
                <field>appt_number</field>
                <field>customer_number</field>
        </keys>
</appointment>
```

4.1.2 Inventory Key Fields

The fields listed in the 'head/inventory/keys' form a composite key that is used to determine, whether an inventory exists in ETAdirect and should be processed, or whether it does not exist and a new inventory should be created and then processed. Therefore, these fields must be present and non-empty for every serialized inventory in the transaction, otherwise the inventory will be rejected.

NOTE: 'head/inventory/keys' is only used for serialized inventory, while for non-serialized inventory it is ignored and inventory type + model property value is used as the key.
See section 'Serialized and Non-Serialized Inventory' for details.

The valid values of 'inventory/keys' are 'invsn', 'invtype' and 'invtype_label'.

- **'head/inventory/keys' structure example:**

```
<inventory>
        <keys>
                <field>invsn</field>
        </keys>
</inventory>
```

4.1.3 Duplicating Properties

Two properties with the same name are called duplicating. They can have different values but in the course of update all will have the value of the properties that belongs to the activity with the highest ID. Sometimes duplicating properties are needed to implement specific logics, but at the same time they can cause confusion. A corresponding warning is issued for an attempt to add duplicating properties.

4.1.4 Property Visibility

The way a property will be processed by the Inbound API depends on its visibility. A property can be set to hidden and will not be seen in any way by the user. On the other hand, visible properties can be mandatory for the request to be processed correctly or optional.

Optional: the user can see the property and can optionally manage it. The 'Required' column contains '**No**' for such property.

Mandatory: – the user can see the property and must define it
– if the transaction contains an invalid mandatory property, the request is rejected with a corresponding error
– if request has no mandatory property, the request is rejected with a corresponding error

The 'Required' column contains '**Yes**' for such property.

4.1.5 Property Type

Every property belongs to a certain type which can be one of the following:

String (referred in tables as 'string')

- any string can be assigned to a property

Integer (referred in tables as 'integer' or 'int')

- any integer numbers can be assigned to a property
- if a request of Inbound API transaction contains a property of 'integer' type but with non-numeric value and the property visibility is not 'mandatory', the property is ignored and a warning is sent in the response
- if a request of Inbound API transaction contains a property of 'integer' type but with non-numeric value and the property visibility is 'mandatory', the request is rejected with the corresponding error

Regular Expression (referred in tables as 'regexp')

- some string and integer properties have a regular expression pattern that restricts possible values
- if a request of Inbound API transaction contains a 'regex' property with a value not matching regular expressions and the property visibility is not 'mandatory', the property is ignored and a warning is sent in the response
- if a request of Inbound API transaction contains a 'regex' property with a value not matching regular expressions and the property visibility is 'mandatory', the request is rejected with the corresponding error

Enumeration (referred in tables as 'enum')

- only values of a fixed subset of integers can be assigned to a property
- the subset is defined for the property and cannot be configured
- the subset values are provided herein

Key (referred in tables as key)

- only values of a fixed subset of integers can be assigned to a property
- the subset is configurable for each specific company

NOTE: In all sections below the property table contains 'name' of the property, its visibility, type and length and description. Length is the value in the brackets by the property type. If the length is not specified then the Inbound API does not check it or it makes no sense (e.g. 'external_id', 'language' are matched against the predefined set of values).

4.1.6 'property' Structure

The 'property' structure is used to represent custom company-specific properties of activities and inventories. It is a simple name-value pair. The 'property' structure consists of the following elements:

Name	Required	Type	Description
label	Yes	string	name of the property, unique for the corresponding property list
value	Yes	string	property value, can be an empty string

- **'property' Structure Example**

```
<property>
        <label>DOOR_COLOR</label>
        <value>ORANGE</value>
</property>
```

4.1.7 'file' Structure

The 'file' structure represents file property of activity and inventory. Its format is more complex than that of a regular property. The 'file' structure consists of the following elements:

Name	Required	Type	Description
property_label	Yes	string	name of the file property, unique for the corresponding property list
filename	Yes	string	original file name with extension
mime_type	Yes	string	MIME type of the file
encoding	Yes	string	encoding of the 'contents' node **valid values:** 'plain-text' or 'base64'
contents	Yes	string	file contents, in either plain text or base64 maximum file size is 512000 bytes but the Base64-encoded file may be larger – it is only checked after being decoded

- **'file' Structure Example**

```
<files>
    <file>
        <property_label>ii_fileprop</property_label>
        <filename>bummer.txt</filename>
        <mime_type>application/octet-stream</mime_type>
        <encoding>base64</encoding>
        <contents>
            Tm90IGEgd29yZCBmcm9tIHRoZWlyIG1vdXRoIGNhbiBiZSB0cnVzdGVkOwp0a
            GVpciBoZWFydCBpcyBmaWxsZWQgd2l0aCBkZXN0cnVjdGlvbi4KVGhlaXIgdG
            hyb2F0IGlzIGFuIG9wZW4gZ3JhdmU7CndpdGggdGhlaXIgdG9uZ3VlIHRoZXk
            gc3BlYWsgZGVjZWl0Lg==
        </contents>
    </file>
</files>
```

4.2 Inventory

Inventory is a piece of hardware that is carried by a technician and can be installed/deinstalled during an activity. The Inbound Interface is used to upload two types of inventories:

Customer Inventory

Inventory that is at the customer's premises before the activity performance belongs to the 'customer' pool in ETAdirect. Inventory in the 'customer' pool is further referred to as 'customer inventory'. Customer inventory is actually an activity property of the activity it is assigned to.

Resource Inventory

Inventory at technician's disposal before the activity performance belongs to 'provider' pool in ETAdirect. Inventory in the 'provider' pool is further referred to as 'provider inventory' or 'resource inventory'. Resource inventory makes a resource property.

Serialized and Non-Serialized Inventory

Serialized inventory is identified with a serial number, while non-serialized inventory has no serial number and is identified by inventory type and model. Non-serialized inventory has the 'quantity' property which defines the number of inventory units in the pool.

4.2.1 'inventory' Structure.

The 'inventory' structure contains customer inventory or resource inventory and its properties. Specific pool is specified every time the structure is mentioned.

The 'inventory' structure contains the following elements:

Name	Required	Type	Description
properties	Yes	struct	array of 'property' structures that contain inventory properties, where 'label' can be any custom inventory property, with read/write visibility for the Inbound API and its corresponding 'value' OR 'label' can be one of the following: 'invsn' — inventory serial number ('value' — string(32)) or 'invtype' — inventory type ('value' – int) or 'invtype_label' – inventory type label (value – string)
userdata	No	string	string returned in SOAP response without change. It is not used by the Inbound Interface, but intended for it (e.g client may use this to store an ID in SOAP request and then quickly find record in SOAP response using this ID) **default value**: none
files	No	struct	array of 'file' structures containing inventory file properties **default value**: none

- **'inventory' Structure Example**

```
<inventory>
        <properties>
                <property>
                        <label>invsn</label>
                        <value>SN34634987</value>
                </property>
                <property>
                        <label>PORT_INFO</label>
                        <value>1:C:1</value>
                </property>
        </properties>
</inventory>
```

4.3 Activities

Activity is the entity of the ETAdirect system that represents any time consuming action of resource. The way activity is processed by the Interface depends on its type and status.

4.3.1 Activity Types

Since ETAdirect 4.1 a single entity — 'activity' corresponds to all time-consuming things done by technicians. Numerous tailored and client-specific types can be specified. Along with a label, name and language, its features are set for each specific type. The features are yes/no flags that define peculiarities of the type processing, e.g. whether activities of the type can be moved, created in a bucket, rescheduled etc. Colors that correspond to a specific activity status now can also be set for each type at time of creation.

The table below provides detailed description of the features:

NOTE: Inbound API obeys activity type restrictions, e.g. an activity will not be moved if the type does not allow it.

Feature	Description
Allow to create from Incoming interface	defines if activities of the type can be created from the Inbound Interface (activities may be originated in TOA and in external systems)
Allow move	defines if activities of the type can be moved between resources
Allow creation in buckets	defines if activities of the type can be created in buckets
Allow reschedule	defines if activities of the type can be moved to another day
Support of inventory	defines if inventory can be used for activities of the type
Support of links	defines if links can be used for activities of the type
Support of not-ordered activities	defines if activities of the type can be not-ordered
Support of not-scheduled activities	defines if activities of the type can be activities without a date cannot be enabled if Teamwork is checked
Support of preferred resources	defines if resource preference (Preferred Resource tab) can be defined for activities of the type
Support of time slots	defines if time slots must be used for activities of the type the logics of activity types and time slots correlation have not changed since 4.0 versions
Teamwork	defines if activities of the type are teamwork activities
Use predefined duration	defines if duration is entered manually for activities of the type If an activity is created with the 'use predefined duration' flag enabled, its duration must be defined (otherwise an error occurs)

4.3.2 Activity Statuses

Activities in ETAdirect can have the following statuses:

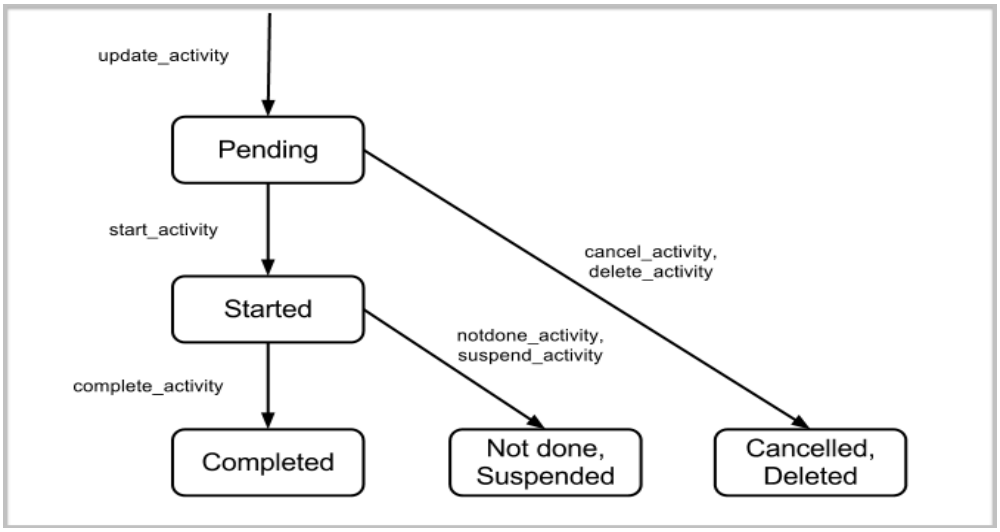

Figure 1: Activity Statuses

Pending

- when activity is created in the resource's queue it gets a 'pending' status

Started

- activity that has been started and is being processed
- only one activity can be 'started' within one queue at the same time

Suspended

- activity that got postponed for some reason
- when activity is postponed in ETAdirect, its 'end_time' is set as the time of suspension, and it gets a 'suspended' status
- along with activity suspension, a new activity of 'pending' status copying the suspended one is created
- the 'suspended' activity is ignored by the Inbound API, and its 'pending' copy is updated instead

Complete

- activity that has been successfully completed

Cancelled

- activity that has not been started and will not be performed

Not done

- activity that has been started but has not been completed

Deleted

- activity that has been deleted from the system

- deleted activities do no appear anywhere in the system and cannot be viewed

4.3.2.1 Ignored and Duplicating Activities

All activity commands ignore any:

- pre-work
- activities with status set to 'cancelled' via the Inbound API
- activities with status set to 'deleted' or 'suspended' (regardless of how the status was set)
- activities scheduled for the past date

Thus even if any of the above-mentioned activities have the key fields, they are not considered among the existing activities and treated by the Inbound Interface as if they do not exist (e.g. if a command is set, and the key fields are present only in the activity cancelled via the Inbound Interface and/or in a deleted activity, they will not be considered and a new activity will be inserted).
If there is more than one activity with the same key fields, the activity with the maximum ID is processed and the rest are treated as 'duplicating' (and cancelled/deleted in the end of the transaction).

4.3.2.2 Activity Status and Incremental Upload

As said above, if an activity is cancelled via the Inbound Interface, or has 'deleted' or 'suspended' status, it is ignored by the Inbound Interface, and if no other activity has key fields, a new activity is inserted and processed.

Pending activities are processed in accordance with the command flow, all its properties, including date, resource and other fields can be processed if set by the command request.

Activities with 'started', 'completed', 'cancelled'(by user) or 'not done' status, that have key field values are processed in accordance with 'action_if_completed' value specified.

4.3.2.2.1 'action_if_completed' Meanings

'action_if_completed' can have any of the following values:

'**ignore**' – not to update the activity and neither to insert a new activity:

- new activity is never created
- command is rejected with 'appointment status is not pending' error

'**update**' – update existing activity

- new activity is never created
- existing activity is updated

NOTE: Regardless of the request data, for these activities only properties can be updated. Therefore, date, resource, fields and inventories cannot be updated (even if 'action_if_complete' = 'update' or 'create')

'**create**' - always create a new activity, unless an existing activity is 'started'

- 'cancelled' by user, 'completed' and 'not done' activities are ignored – new activity is created
- 'started' activity properties are updated

'create_if_reassign_or_reschedule' – create activity with new date or 'provider/provider_group' (= 'create_as_new' and default)

- if date specified for the transaction is different from the one specified for the started activity, a new 'pending' activity is created
- if date OR provider/provider group specified for the transaction is different from the one specified for 'cancelled' by user, 'completed' or 'not done', a new 'pending' activity is created
- in other cases, the existing activity is updated

NOTE: If 'head/appointment/action_if_completed' value of the request contradicts 'command/appointment/appointment/action_if_completed' value, the meaning defined for the command should take precedence.

4.3.3 'appointment' Structure

The 'appointment' structure contains activity properties to be processed by the Inbound API, including activity fields, properties of the customer inventory assigned to the activity, its company-specific properties, its file properties and data on activities linked to it.

NOTE: Please note, that there are 4 fields available in 'appointment' structure through SDK, that are intended for storing and proper processing of activity address information:
- 'address' (should contain street and building reference. Should **not** contain the information for which the dedicated fields are present (city, state, zip), or which is not relevant to activity location).
Not following the rules of fields mapping (like putting all address data into single field, including additional information like "flr. 5, appt. 25, knock 3 times and call Bob") will most likely lead to difficulties with further geocoding and routing processes inside ETAdirect.

Fields visibility is specified separately for each company user. The 'appointment' structure is an array of 'appointment' nodes that can contain the following elements (if no default value is defined in the table, the element by default will be empty):

Name	Type	Description
address	string (100)	customer's address
appt_number	string (40)	ID of the activity in client system
action_if_completed	enum	'action_if_completed' parameter, defines the way non-pending properties are processed. Please see section 'action_if_completed' Meanings for details. **default value:** 'head/appointment/action_if_completed' or its default
cell	regexp	valid customer's cell phone
city	string (40)	customer's city of residence
coordx	float (64bit double)	longitude of customer's location
coordy	float (64bit double)	latitude of customer's location
customer_number	string (40)	customer's account number
email	regexp	customer's valid email address
duration	integer [0 - 65535]	defines duration ('length') of activity in minutes Mandatory for 'insert_appointment' if for activity type 'use predefined duration' feature is enabled (otherwise it will be ignored).
language	string	company-specific language ID or country code top-level domain (ccTLD). If the value is numeric it is treated as integer language ID, else it is treated as CCTLD. **default value:** resource's language ID
name	string	optional customer's name
points	integer [0 - 65535]	optional meaningful for 'update_activity', 'cancel_activity' and 'delete_activity' and is used in Quota Management and Routing **default value**: 0 (please set visibility to mandatory if this default is not acceptable)
phone	regexp	valid customer's regular phone number

Name	Type	Description
reminder_time	integer	customer's reminder notification time: minutes before the start of the activity about which the customer is to be notified. The valid values are defined in the process of implementation.
service_window_start	time	service window start time in HH:MM format **default value:** If the 'Support of Time Slots' feature is enabled for the activity type, then these fields are ignored and 'time_slot' is used instead, otherwise these fields are used to set the service window. Either none or both fields must be present – specifying only one results in error.
service_window_end	time	
sla_window_start	datetime	start of time when the activity can be performed without an SLA loss YYYY-MM-DD HH:MM format
sla_window_end	datetime	end of time when the activity can be performed without an SLA loss YYYY-MM-DD HH:MM format
state	regexp	customer's state, one of valid state values
team_id	string	external identifier of the team-holder Mandatory for 'insert_appointment' if activity type is teamwork, otherwise it is ignored.
time_slot	enum	If the 'Support of Time Slots' feature is enabled for the activity type, then 'time_slot' field is mandatory (command rejected if absent). If the 'Support of Time Slots' feature is disabled for the activity type, then 'time_slot' field must not be present (command rejected if present).
time_zone	enum	time zone ID or time zone name If the value passed in the time zone field is numeric it is treated as integer time zone ID, else it is treated as time zone name **default value:** resource's time zone ID
worktype	enum	activity type id and label of the worktype only one of the two must be present (mandatory)
worktype_label	enum	
zip	regexp	optional; valid zip code customer's zip/post code; depends on company's rules for zip codes
inventories	struct	array of 'inventory' structures containing activity inventories
properties	struct	array of 'property' structures containing activity properties
provider_preferences	struct	array of 'preference' structures the define resources preferred, required or forbidden for the activity When empty, the existing preferences are deleted.
userdata	string	string that is returned in SOAP response without change, not used by Inbound Interface, intended for client use (e.g. to store an ID in SOAP request and quickly find record in SOAP response using this ID)
files	struct	array of 'file' structures containing activity file properties
links	struct	array of 'link' structures each describing a link between the current activity and another activity
required_inventories	struct	array of 'required_inventory' structures that define the inventories required for the activity

- **'appointment' Structure Example**

```
<appointment>
        <appt_number>0001</appt_number>
        <customer_number>Customer_number N85</customer_number>
        <worktype>11</worktype>
        <worktype_label>DEI</worktype_label>
        <service_window_start>8:00</service_window_start>
        <service_window_end>19:00</service_window_end>
        <name>name</name>
        <phone>11111111</phone>
        <email>email@email.com</email>
        <cell>11111111</cell>
        <address>address</address>
        <city>city</city>
        <zip>11111</zip>
        <state>CA</state>
        <points>234</points>
        <language>1</language>
        <daybefore_flag>0</daybefore_flag>
        <reminder_time>0</reminder_time>
        <time_zone>2</time_zone>
        <properties>
                <property>
                        <label>MAP_GRID</label>
                        <value>AA11</value>
                </property>
        </properties>
        <userdata>12345 </userdata>
    </appointment>
```

4.3.3.1 'links' Structure

The 'links' structure is a list of activities linked to the current activity.

Links may point to the activities existing prior to the current transaction or added in the current transaction.

- If a link cannot be added, a warning is issued and the activity is still handled normally
- If a link to the specified activity already exists, nothing happens
- If a link to the specified activity of another type exists then the existing link is erased first (e.g there is a Start After link and a Start Together link is added)
- For 'cancel_appointment' commands, links will be ignored

The 'links' structure consists of an array of the following optional structures (at least one must be present):

Name	Type	Description
erase_links	struct	array of 'link_type' elements defining the link type to be deleted for the activity If the 'erase_links' structure contains no 'link_type' elements, all existing links of the activity are deleted.
link	struct	element containing the data of the link to be created between two activities **NOTE**: this element is used instead of the obsolete 'link_start_after' and 'link_start_together' elements
link_start_together	struct	element that contains details of another activity that will start simultaneously with the current activity **NOTE**: obsolete element replaced with the 'link' element, however, still used for backward compatibility
link_start_after	struct	element that contains details of another activity that will start before the current activity **NOTE**: obsolete element replaced with the 'link' element, however, still used for backward compatibility

- **'erase_links' Structure**

Name	Required	Type	Description
link_type	No	string	label of the link type to be deleted. All links of the same type will be deleted. **NOTE**: each directed link has two labels. If the 'erase_links' structure contains only one label, only one direction of the link will be deleted.

- **'link' Structure**

Name	Required	Type	Description
appt_number	Yes	string	ID of the activity to be linked to the current activity in the client system
link_type	Yes	string	label of the link type. The 'from' and 'to' type labels are acceptable.
min_interval, max_interval	No	int	minimal and maximal time interval between two linked activities. This parameter should be present or absent depending on the selected link type.

- **'links' structure example**

```
<appointment>
  <appt_number>A</appt_number>
  <links>
    <erase_links/>
    <link_start_after>
      <appt_number>B</appt_number>
    </link_start_after>
    <link_start_together>
      <appt_number>C</appt_number>
    </link_start_together>
    <link>
      <appt_number>D</appt_number>
      <link_type>related</link_type>
    </link>
    <link>
      <appt_number>E</appt_number>
      <link_type>clone of</link_type>
    </link>
    <link>
      <appt_number>F</appt_number>
      <link_type>start-after</link_type>
      <min_interval>10</min_interval>
      <max_interval>60</max_interval>
    </link>
    <link>
      <appt_number>F</appt_number>
      <link_type>start-together</link_type>
    </link>
  </links>
</appointment>
```

4.3.3.2 'preference' Structure

'provider_preferences' structure is an array of 'preference' structures to define resources preferred, required or forbidden for the activity to implement company-specific business logics. 'preferences' structure contains the following elements:

Name	Type	Description
'external_id'	string	ID of the resource for whom the preference is set (is mandatory for the structure)
'type'	enum	type of preference (is mandatory for the structure): **valid values**: **required**: if any of the resources in the 'provider_preferences' node have preference/type='required', only one of these resources can be assigned the activity **preferred**: if no resources the 'provider_preferences' node have preference/type='required', the resources with preference/type='preferred' will have the priority when the activity is assigned **forbidden:** resources with preference/type='forbidden', cannot be assigned the activity

NOTE: When 'provider_preferences' structure is present, but empty, the existing preferences are deleted.

- **'provider_preferences' Example**

```
<provider_preferences>
   <preference>
      <external_id>11103</external_id>
      <type>forbidden</type>
   </preference>
   <preference>
      <external_id>11104</external_id>
      <type>preferred</type>
   </preference>
</provider_preferences>
```

4.3.3.3 'required_inventories' Structure

The 'required_inventories' structure is an array of 'required_inventory' structures that define inventories required for the activity performance.

If an 'update_activity' command contains a 'required_inventories' element, then:

- the existing required inventories of the activity are deleted
- new required inventories are added when specified in the request
- an empty 'required_inventories' element deletes all existing required inventories of the activity

The required inventory will not be added if:

- Command type is not 'update_activity'
- Activity status is not 'pending'
- Activity does not have the 'required inventory support' feature enabled
- Inventory Type specified in the request is invalid
- Model specified in the request does not match the model property rules
- Required Inventory with this type ID and Model already exists for this activity

In all of the above cases, a warning is returned in response and the rest of the command is executed without modifying the required inventory.

'required_inventory' structure contains the following elements:

Name	Type	Description
type	string	Inventory type label
model	string	Inventory model. Valid values depend on the inventory type. 'model' is validated against the rules of the 'Model' property of 'Inventory Type'.
quantity	int	Number of inventories required for the activity.

- **'required_inventories' Structure Example**

```
<required_inventories>
   <required_inventory>
      <type>CONNECTOR</type>
      <model>CX1135</model>
      <quantity>5</quantity>
   </required_inventory>
</required_inventories>
```

5 Detailed Commands Description

5.1 Command List

Activity-related commands can be used to manage an activity throughout its lifecycle.

The following activity-related commands are available:

- start_activity
- complete_activity
- notdone_activity
- suspend_activity
- update_activity
- cancel_activity
- delete_activity

The following inventory-related commands are available:

- set_inventory
- update_inventory
- delete_inventory

5.2 Activity-Related Command Actions

Basically, the commands perform the following activity-related actions (not all commands perform all actions, so please see the command description for details):

- finding an existing activity
- determining if the activity is to be moved or re-scheduled
- ordering activities in the resource route
- updating the properties of exiting activity
- updating inventories of existing activity
- cancelling and deleting activity

5.2.1 Finding an Existing Activity

The logic of activity-related commands depends on the presence or absence of an existing activity, where 'activity does not exist' means there are no activities that match all the key values and 'activity exists' means at least one activity matches all key values:
To determine if the activity already exists in the system, activities with the same key values are found and filtered as follows:

5.2.1.1) **IF**:

- existing activity type is 'break' OR 'prework'
- existing activity status is one of {'deleted', 'suspended'}
- existing activity is scheduled for a past date
- existing activity status is 'cancelled' and was cancelled by Inbound API (not by a Manage Application user)

THEN: activity is ignored (not considered existing)

IF: there still is more than one activity with the same key fields

THEN: the activity with the maximum ID is treated as existing and the others are treated as duplicated and are cancelled/deleted after a successful command execution.

5.2.2 Determining if the Activity Is to Be Moved or Rescheduled

NOTE: Please note that activity move or reschedule is the second step of the command execution, that is performed before the activity is updated, cancelled, or deleted. If the move or reschedule defined in the request cannot be performed, the command will be rejected.

5.2.2.1) **IF**: activity does not exist AND command type is 'cancel_appointment'

THEN: command is rejected with message 'appointment is not found'

5.2.2.2) **IF**: command date field is not specified AND activity does not exist

THEN: command is rejected with message 'date is empty'

5.2.2.3) **IF**: command date field is not specified AND activity already exists

THEN: command date is set to existing activity date

NOTE: If the date field is empty – the activity is set to non-scheduled

5.2.2.4) **IF**: command date field value is a date in the past

THEN: command is rejected with a message 'action on the past is not allowed'

5.2.2.5) **IF**: command 'external_id' field is not specified AND activity does not exist

AND default appointment pool is not specified or invalid

THEN: command is rejected with message 'external_id not specified'

5.2.2.6) **IF**: command 'external_id' field is not specified AND activity does not exist

AND default appointment pool is specified and valid

THEN: command provider is set to default appointment pool value AND 'external_id not specified – inserting into default pool' warning is issued

5.2.2.7) **IF**: command 'external_id' field is not specified AND activity already exists

AND existing activity provider is invalid

AND default appointment pool is not specified or is invalid

THEN: existing activity is updated but not moved/rescheduled or cancelled

5.2.2.8) **IF**: command 'external_id' field is not specified AND activity already exists

AND existing activity provider is invalid

AND default appointment pool is specified and valid

THEN: command provider is set to default appointment pool value and 'external_id not specified and existing queue is invalid – moving to default pool' warning is issued

5.2.2.9) **IF**: command external_id field is not specified AND activity already exists

AND existing activity provider is valid

THEN: command provider is set to existing activity provider

5.2.2.10) **IF**: command 'external_id' field is specified and valid AND activity already exists

AND command 'external_id' is not equal to existing activity 'external_id'

AND command date is equal to existing activity date

AND existing activity provider is in the same provider group as command provider

THEN: command provider is set to existing activity provider AND 'provider is in the same group as original provider – will not move' warning is issued

5.2.2.11) **IF**: command 'external_id' field is specified AND activity already exists

AND command provider is invalid

AND default activity pool is not specified or invalid

THEN: existing activity is updated but not moved/rescheduled and not cancelled

5.2.2.12) **IF**: command 'external_id' field is specified AND activity does not exist

AND command provider is invalid

AND default appointment pool is not specified or invalid

THEN: command is rejected with 'queue is invalid' message

5.2.2.13) **IF**: command 'external_id' field is specified AND activity already exists

AND command provider is invalid

AND default appointment pool is specified and valid

THEN command provider is set to default appointment pool value AND 'queue is invalid – falling back to default pool' warning is issued

5.2.2.14) **IF**: neither service window start or end are specified

THEN: new activity is created as 'unordered'

5.2.2.15) **IF**: activity already exists AND command date is different from existing activity date

AND activity has been set started, completed, cancelled, or notdone via the Manage Application

THEN: existing activity is ignored and new pending activity is added for the new date

5.2.2.16) **IF**: activity already exists AND command provider is different from existing activity provider

AND activity has been set completed, cancelled or notdone via the Manage Application

THEN existing activity is ignored and new pending activity is added to the new provider

5.2.2.17) **IF**: activity already exists AND command provider is different from existing activity provider

AND activity has been set started via the Manage Application

THEN activity is not moved nor it is created – the properties of existing activity are updated.

5.2.3 Ordering Activities in the Route

Activity-related commands deal with ordered and not-ordered activities.

Not-ordered Activities can be completed at any time of the day and appear at the top of the list in the GUIs, and can be started at any time. Relative order of not-ordered activities is not significant.

(If activity is uploaded through the Inbound API and its service window time is not specified in the transaction, it becomes unordered. The 'unordered' flag can be set by the user through the Manage Application when either creating a new activity or updating an existing one. Then even if a service window is set in the Inbound Interface, the activity will stay unordered).

Ordered Activities should be started within the period defined with the service window start and service window end values. In the GUIs, earlier activities are higher in the list.

Activities are ordered in the resource's route (activity ordering is not significant for buckets):

- by service window end values
- if they are the same, activities are ordered by the service window start values
- if they are the same, activities are ordered by their SLA window end values
- if they are the same, activities are ordered by their SLA window start values
- if they are the same, activities are ordered by the activity ID

When a route contains ordered finish-to-start linked activities, such links have higher priority in the activities ordering than any other criteria. The activities are ordered according to their sequence in the link and afterwards the service window, SLA and ID are checked.

Here is an example of correct ordering by the activity service window:

#	service window start	service window end
1	8:00	10:00
2	10:00	11:00
3	8:00	12:00
4	10:00	12:00:00

When activities are updated, they are again reordered by the same criteria, if necessary.

5.2.4 Updating/Replacing Properties

In the course of activity-related commands execution, activity properties can be updated or replaced, subject to 'head/properties_mode' setting of the transaction.
'replace': when activity is updated, all existing properties are erased and properties from the request are added instead
'update': when activity is updated, properties from the request are added to the existing properties, the existing properties are not deleted.
The service window is always updated to the service window value specified in the command:
IF: date AND provider of existing activity equal date AND provider of the command
AND service window of existing activity is different from service window of the command
THEN: service window of the command is set for the activity
If the service window value of the command is empty, it is set to empty for the activity and unordered flag is set to true.

NOTE: If a field/property is not mentioned in the request and the 'head/properties mode' is 'update', the values of the field/property are not changed.

If in a request with 'head/properties mode' = 'update', a field/property is sent with an empty value, the values of most of the fields/properties are set to empty value.

The exceptions are 'files' and 'links' fields – an empty field in the request with 'head/properties mode' = 'update' will not erase them but will leave them unchanged.

5.2.4.1 Updating Fields Changed in Manage Application

Some fields of the existing activities cannot be updated by the Inbound Interface, if they have been previously changed in ETAdirect Manage Application. This is done in order to protect the changes made by the user from being erased by the update.

These fields are:

- 'reminder_time'
- 'language'
- 'time_zone'
- 'phone'
- 'email'

- 'cell'
- 'name'
- 'address'
- 'city'
- 'zip'
- 'state'
- time_slot_id'
- 'sla_window_start'
- 'sla_window_end'
- 'service_window_start'
- 'service_window_end'

5.2.5 Updating Activity Inventory

In the course of activity-related incremental commands execution, activity inventory's properties can be updated. All activity inventory specified in the request is validated and the inventory that has been validated is updated.

5.2.5.1 Activity Inventory Validation

Inventory is validated or rejected if conditions are not met, and the successfully validated activity inventory items are updated. The inventory is validated as follows:

5.2.5.1.1) **IF:** inventory keys are not set in the transaction head ('head/inventory/keys')

THEN: inventory is rejected with 'inventory key fields are not defined' message

5.2.5.1.2) **IF**: inventory has no properties at all

THEN: inventory is rejected with 'inventory properties are absent' or 'inventory key is absent' message

5.2.5.1.3) **IF:** any of inventory key properties is not specified

THEN: inventory is rejected with 'inventory key field <FIELD> is empty' or 'inventory key field <FIELD> is absent' message

5.2.5.1.4) **IF** any of inv_pid; inv_aid; invtype; invid; inv_change_invid values are non-numeric

THEN: inventory is rejected with 'inventory key field <FIELD> has numeric type but non-numeric value <VALUE>' message

5.2.5.1.5) **IF**: any inventory property has no name

THEN: inventory is rejected with 'property has no name' message

NOTE: If an inventory fails any of the following checks, then only this inventory action is not performed.

5.2.5.2 Updating Validated Inventory

No existing inventory is deleted. For each inventory piece, the following logics are realized:

5.2.5.2.1) **IF:** existing activity has inventory with the same key values as this inventory piece

THEN: all non-key fields of existing inventory are updated with values from the command AND fields of the existing inventory absent in the inbound command are deleted (if any)

5.2.5.2.2) **IF:** existing activity does not have this inventory piece

THEN: new inventory is inserted

5.2.6 Canceling/Deleting Activities

If several activities meet all key-fields, only the activity with the maximum ID is processed. Other such activities are deleted if the resource's queue has not yet been started, or cancelled if the queue has been started.

5.3 Peculiarities of Activity Processing

5.3.1 Canceling a Non-Scheduled Activity

Before 4.1.6 it was only possible to cancel a non-scheduled activity by explicitly specifying empty 'date' in the 'cancel_activity' command.

Since ETAdirect 4.1.6 when canceling a non-scheduled activity, if the date is not specified, it will be rescheduled to the current day and then cancelled.

IF: The resource has a non-working day today or is inactive

AND fallback resource ('command/fallback_external_id' or 'head/default_appointment_pool') is not specified or is invalid

THEN: an error will appear and the activity will not be cancelled

IF: The resource has a non-working day today or is inactive

AND a fallback resource ('command/fallback_external_id' or 'head/default_appointment_pool') is specified and valid

THEN: activity will be moved to the fallback resource's queue for the current day and cancelled.

5.3.2 Deleting a Non-Scheduled Activity

Since ETAdirect 4.1.6, 'delete_activity' command can delete non-scheduled activities without moving them, so it will never fail.

5.3.3 Updating Activity Assigned to a Non-Working Resource

Before 4.1.6 if an existing activity was assigned to a resource with non-working calendar or inactive resource or had to be moved to such a resource with the update, upon an update such activity was cancelled with the 'Queue is invalid' error message.

Since ETAdirect 4.1.6, such activity will be updated as requested, but it will be moved or cancelled:

IF: 'update_activity' command is performed for an existing activity

AND activity is currently assigned to inactive queue (non-working day, inactive provider, etc.)

AND command does not have to move/reschedule activity ('external_id'/date not specified or are the same as existing)

THEN:

IF the fallback/default pool is specified

activity is updated, and moved to fallback/ default pool for that date

ELSE activity is updated, but not moved/rescheduled and not cancelled

IF: 'update_activity' command is performed for an existing activity

AND command is to move activity to inactive queue (non-working day, inactive provider, etc.)

THEN:

IF the fallback/default pool is specified

activity is updated, and moved to fallback/ default pool for that date

ELSE activity is updated, but not moved/rescheduled and not cancelled

5.4 Activity Related Command Details

5.4.1 'start_activity' Command

The command affects only the activity specified by key fields of 'command/appointment' element that exists in the system within an active queue for the current day and sets the activity status to 'started'. The command will:

1) If the activity processed is not the first in the queue:

1a) move the activity into the first position in the queue

1b) issue 'the appointment starting order is invalid' warning

2) If/when the activity processed is the first in the queue:

2a) start the activity specified by key fields of 'command/appointment' element

2b) update activity properties if 'command/appointment/properties' element is present

2c) record travel time (time from the end of the previous activity to the start of the processed activity) in the statistics table unless the activity was not initially the first in the queue

The command will fail if:

- activity does not exist in ETAdirect
- activity is in an inactive queue
- 'command/time' is not current day (except overnight work)
- another activity in queue has status 'started'
- 'command/time' is less than queue activation time
- 'command/time' is less than the time the previous activity was finished

5.4.2 'complete_activity' Command

The command affects only the started activity specified by key fields of 'command/appointment' element that exists in the system and sets the activity status to 'completed'. The command will:

a) complete the existing started activity, specified by key fields of 'command/appointment'

b) update activity properties if 'command/appointment/properties' element is present

The command will fail if:

- activity does not exist in ETAdirect
- activity status is other than started
- 'command/time' is less than activity start time

5.4.3 'notdone_activity' Command

The command affects only the started activity specified by key fields of 'command/appointment' element that exists in the system and sets the activity status to 'not done'. The command will:

a) set existing started activity, specified by key fields of 'command/appointment' to 'not done'

b) update activity properties if 'command/appointment/properties' element is present

The command will fail if:

- activity does not exist in ETAdirect
- activity status is other than started
- 'command/time' is less than activity start time

5.4.4 'suspend_activity' Command

The command affects only the started activity specified by key fields of 'command/appointment' element that exists in the system. It sets the activity status to 'suspended' and creates a new pending activity. The command will:

a) set existing started activity specified by key fields of 'command/appointment' to 'suspended'

b) create new activity with 'pending' status that duplicates the 'suspended' activity

c) set the 'end_time' of the suspended activity to command/time

d) update pending activity properties if 'command/appointment/properties' element is present

The command will fail if:

- activity does not exist in ETAdirect
- activity status is other than started
- 'command/time' is less than activity start time

5.4.5 'update_activity' Command

The command affects only the specified activity. The command is alias for 'update_appointment' command. It works as follows:

1) If no activities with specified key field values exist in the system:

1a) new activity is inserted

1b) fields, properties, and customer inventory is set for it, as specified in the request.

2) If a pending activity with specified key field values exists in the system:

2a) its date, resource, fields, properties and inventory is updated/replaced as set in request

2b) all duplicating activities are deleted/cancelled.

3) if a 'started', 'cancelled' by user, 'completed', and 'notdone' activity with specified key field values exists in the system, it is processed in accordance with its 'action_if_completed' meaning

NOTE: Please see Updating/Replacing Properties section for more details on the 'update_activity' work.

5.4.6 'cancel_activity' command

1) If no activities with specified key field values exist in the system the command is rejected.

2) If an activity with specified key field values exists in the system:

2a) activity properties and fields (not inventory) are updated by the 'update_appointment' command

2b) activity is cancelled

2c) all duplicating activities are deleted

3) If existing activity is not-scheduled and 'date' field is not defined:

3a) if the resource for the current day is valid, the activity is rescheduled to the current date and then cancelled

3b) if the resource is not working or not valid for the current day and fallback resource is specified and valid – the activity is moved to the fallback resource, rescheduled to the current date and then cancelled

3c) if the resource is not working or not valid for the current day and fallback resource is not specified or is invalid, the command is rejected and the activity is not cancelled.

5.4.7 'delete_activity' Command

The command affects the pending activity specified by key fields of 'command/appointment' that exists in the system and deletes the activity from the system if the queue has not been started yet or cancels the activity otherwise. The command will:

1) move the activity to the specified resource, if the 'external_id' is specified in the command and is different from the existing activity 'external_id'

2) if the resource's queue is active:

2a) update the activity properties and fields (not inventory) by 'update_appointment' command

2b) cancel the activity

3) if the resource's route is inactive, delete the activity (the 'on move' message scenario will not be triggered)

The command will fail if the activity does not exist in ETAdirect or is not of 'pending' status.

5.5 Inventory-Related Commands

Inventory-related commands process resource inventory in ETAdirect. The inventory validation logic is the same as of activity inventory validation
If any inventory set in the command is invalid, it is rejected, the corresponding message is issued, but the command is performed for the rest of the inventory.

5.5.1 'set_inventory' Command

This command affects only the specified technician's inventory. It works as follows:

1) all inventory in the specified technician's pool is deleted

2) inventory specified in the command is added to the technician's pool

5.5.2 'update_inventory' Command

This command affects only the specified technician's inventory. The main difference from the 'set_inventory' command is that no inventories are deleted. It works as follows:

1) If no inventory with specified key field values exists in the system for the technician specified:

1a) new inventory is inserted

1b) fields and properties are set for it, as specified in the request.

2) If inventory with specified key field values exists in the system for the technician specified:

2a) fields and properties specified in the request are updated

2b) fields not specified in the command are deleted.

5.5.3 'delete_inventory' Command

This command affects only the specified technician's inventory. Basically, it deletes all inventories present in the command from the technician's pool.

6 'inbound_interface_request' Method Description

The only method used by the Inbound API for all transactions is the 'inbound_interface_request' method.

6.1 Method Workflow

6.1.1. The request size is checked. If it exceeds 20 MB the transaction is rejected and HTTP code 400 is returned.

6.1.2. Authentication check is performed.

6.1.3. Upload type is determined as either Incremental Upload or Full Upload.

6.1.4. SOAP request is checked for validity.

6.1.5. Upload-specific processing is performed (see sections below).

6.1.6. All actions added to the action queue are executed, modifying the database.

6.1.7. Response is returned to the agent containing either a single error response, if the transaction validation failed, or detailed reports on individual commands.

6.1.1 Incremental Upload Workflow

For each command in the transaction following actions are performed:

6.1.1.1 command is checked for presence of mandatory fields and rejected if any check fails

6.1.1.2 command's individual items (activities/inventories) are validated

6.1.1.2.1 if the command contains several customer/resource inventories, they are validated or rejected separately

6.1.1.2.2 Inbound API determines actions to be performed for this command and adds them to the action queue.

6.1.2 Full Upload Workflow

For each resource in the transaction the following actions are performed:

6.1.2.1 resource is validated, e.g. if the resource is not found in the ETAdirect database, all activity/inventory commands for this resource fail

6.1.2.2 each activity/inventory of this resource is checked individually

6.1.2.2.1 if any activity/inventory fails these checks, this item is skipped with an error message and other items are processed further

6.1.2.3 Inbound Interface determines actions to be performed for each activity/inventory and adds them to the action queue

6.1.2.3.1 for Full activity Upload all activities for the upload date not updated by the transaction are cancelled ('cancel_activity' commands for them are added to the action queue)

6.1.2.3.2 for Full Inventory Upload all inventories not updated by the transaction are deleted ('delete_inventory' commands are added to the action queue). As inventories are not associated with date, all inventories are removed regardless of the transaction upload date.

6.2 'inbound_interface_request' Request

The request node contains three mandatory child nodes:

- '**user**': authentication structure
- '**head**': contains settings that apply to the entire transaction and are necessary to interpret the actions in the 'data' node
- '**data**': contains actions to be performed, i.e. actual activities and/or inventories to be uploaded.

6.2.1 'user' Authentication Structure

The 'user' structure is used for authentication and contains the following string elements:

Name	Required	Type	Description
now	Yes	string	current time in ISO 8601 format
company	Yes	string	case-insensitive identifier of the Client, for whom data is to be retrieved (the instance name) provided by TOA technologies during integration
login	Yes	string	case-insensitive identifier of a specific user within the Company provided by TOA technologies during integration
auth_string	Yes	string	authentication hash auth_string = md5(now + md5(password)); where 'password' is case-sensitive set of characters used for user authentication provided by TOA technologies during integration

For example:

```
<user>
   <now>2008-10-01T17:52:48+0300</now>
   <login>engineer</login>
   <company>sunrise</company>
   <auth_string>a730db71f33be0ee0d17418c771adfd4</auth_string>
</user>
```

6.2.1.1 Authentication

The node is used for the authentication request. If any of the situations below occur, authentication fails and the relevant error is returned. Authentication fails if:

1	now	is different from the current time on the server and this difference exceeds the predefined time-window (30 minutes by default)
2	company	cannot be found in the ETAdirect
3	login	cannot be found for this company
4	user with this 'login' is not authorized to use the current method	
5	auth_string	is not equal to md5(now+md5(password)) ***For example:*** *'now' = "2005-07-07T09:25:02+00:00" and password = "Pa$$w0rD"* *then* *md5 (password) = "06395148c998f3388e87f222bfd5c84b"* *concatenated string =* *= "2005-0707T09:25:02+00:0006395148c998f3388e87f222bfd5c84b"* ***auth_string should be:*** *auth_string = "62469089f554d7a38bacd9be3f29a989"*

Otherwise authentication is successful and the request is processed further.

6.2.2 'head' Node

The 'head' node defines the settings that apply to the entire transaction, and define the transaction flow details. As the elements visibility differs for different upload **types,**the table contains the visibility of each element in the first line of 'Description' cell.

NOTE: For activity related uploads, properties mode can also be set defining if all activity properties not being updated should be erased.

The 'head' node contains the following elements:

Name	Type	Description
upload_type	enum	**mandatory for all upload types** type of the transaction **valid values:** 'full'; 'incremental' 'full': full activity upload or full inventory upload 'incremental': incremental upload
id	string	**optional for all upload types** Can be set if the client wants to identify the transaction, otherwise it can be omitted. unique transaction identifier 'id' is not used by the application itself but has proved to be useful in particular transaction analysis recommended to generate it by some unique-string generator such as a Universally Unique Identifier algorithm
date	date	**mandatory and meaningful only for Full activity Upload** date of transaction in YYYY-MM-DD format specifies the date for which the activities should be uploaded
provider_group	string	**optional, meaningful only for Incremental Upload** label of a resource property used to divide resources within one company into "groups" and used to determine whether the activity can be moved Whenever the Inbound API tries to move an existing activity from one resource to another, it first checks if they belong to the same resource group and if they do, the activity is not moved; i.e. **IF**: inbound activity's 'external_id' field is specified and valid AND activity already exists in the database AND inbound activity 'external_id' is not equal to the existing activity 'external_id' AND inbound activity date is equal to the existing activity date AND existing activity resource is in the same resource group as the inbound activity resource **THEN**: activity is not moved. Existing activity is updated **valid values:** any valid resource property label **default value:** empty string
processing_mode'	enum	**optional, meaningful for Full activity Upload and Full Inventory Upload** defines if activities or inventory will be uploaded: **valid values:** 'appointment_only'; 'inventory_only' **default value:** 'appointment_only'

Name	Type	Description
default_appointment_pool	string	**meaningful only for Incremental Upload (activity commands)** a fallback resource to whom an activity will be assigned, if it cannot be assigned to the resource specified for the activity **valid values:** any valid 'external_id' of a resource or a bucket **default value:** empty string
allow_change_date	enum	**optional, meaningful only for Incremental Upload (with activity-related commands)** specifies, if activities can be re-scheduled to a different date • 'yes' then the activity is re-scheduled to a new date • 'no' then the activity is ignored and a new activity is inserted in the new date **valid values:** 'yes', 'no'; **default value:**'yes'
appointment	struct	**mandatory for Full Activity Upload and for Incremental Upload (with activity-related commands)** activity-processing settings and list of fields to be used to uniquely identify activities when processed
appointment/keys	struct	**mandatory for Full Activity Upload and for Incremental Upload (activity-related commands)** 'keys' structure with activity key fields used to identify processed activities by **valid values** for 'keys/field' are 'appt_number', 'customer_number' and 'name'
appointment/action_if_completed	enum	**optional, meaningful for Full activity Upload and for Incremental Upload (activity-related commands)** Please see section 'action_if_completed' Meanings for details. specifies processing flow for 'started', 'cancelled' by user, 'completed' and 'notdone' activities **valid values:** 'ignore', 'update' 'create_if_reassign_or_reschedule'; 'create' **default value:** 'create_if_reassign_or_reschedule' If the command value contradicts the value set in the 'appointment' structure for specific command, the meaning defined for the command should take precedence.
inventory	struct	**mandatory for all upload types** For Full Inventory Upload and for Incremental Upload (with inventory-related commands) – relates to resource inventory, and for Full activity Upload and for Incremental Upload (with activity-related commands) – relates to customer inventory of the activity. Contains inventory-specific settings including a list of key fields to be used to uniquely identify resource inventories when processing.

Name	Type	Description
inventory/keys	struct	**mandatory for all upload types** For Full Inventory Upload and for Incremental Upload (with inventory-related commands) – relates to resource inventory, and for Full activity Upload and for Incremental Upload (with activity-related commands) – relates to customer inventory of the activity. 'keys' structure with inventory key fields used to identify processed resource inventory by **valid values** for 'keys/field' are: 'invsn'– inventory serial number 'invtype'– inventory type invtype_label – inventory type label, same meaning as 'invtype'
inventory/upload_type	enum	**optional and meaningful for all upload types** For Full Inventory Upload and for Incremental Upload (with inventory-related commands) – relates to resource inventory, and for Full activity Upload and for Incremental Upload (with activity-related commands) – relates to customer inventory of the activity. **valid values:** 'full':when inventory is updated, all existing inventory is deleted and inventory specified in the command is set 'incremental': inventory specified in the request are added to the existing inventory; existing inventory is not deleted **default value:** 'full'
properties_mode		**optional and meaningful only for Full activity Upload and Incremental Upload (with activity-related commands)** **valid values:** 'replace': when activity is updated, all existing properties are erased and properties from request are added instead 'update': when activity is updated, properties from request are added to existing properties; the existing properties are not deleted **default value**: for 'update_activity' command type the default value is 'replace', for other types it is 'update'

NOTE: Please note, that if the user has no rights to change the property, it will not be deleted or changed, regardless of the request setting.

6.2.3 'data' Node

The format of the 'data' section differs depending on the 'head/processing_mode' and 'head/upload_type'. Depending on these values, there can be three different structures:

For 'head/upload_type' = 'incremental' – Incremental Upload 'data' Node

For 'head/upload_type' = 'full' – Full activity/Inventory Upload 'data' Node

6.2.3.1 Incremental Upload 'data' Node

The incremental upload 'data' contains a single child 'commands' which is an array of individual 'command' structures, as follows:

```
<data>
        <commands>
                <command></command>
                <command></command>
                <command></command>
        </commands>
</data>
```

- **'command' structure**

The 'command' structure consists of the following elements:

Name	Required	Type	Description
type	Yes	enum	type of command to be performed **valid values**: 'start_activity'; 'complete_activity'; 'notdone_activity'; 'suspend_activity'; 'update_activity'; 'cancel_activity'; 'delete_activity'; 'set_inventory'; 'update_inventory'; 'delete_inventory'
date	-	date	date of command in YYYY-MM-DD format; **mandatory to create activity** and defines the date to which the activity is to be assigned the field is processed in accordance with reschedule rules If command date field is defined AND not empty, the activity is rescheduled to the date. If command date field is not defined AND activity does not exist, command is rejected with message 'date is empty'. If command date field is not specified AND activity already exists, command date is set to the existing activity date. If command date field is defined but empty, the activity is non-scheduled. **NOTE:** If a date is specified in the command request (whether it is empty or not) the activity will be rescheduled accordingly prior to its update / cancellation / deletion and if such a reschedule is not valid, the command will be rejected).
external_id	No	string key	indicates the resource, for which the command will be performed, if the resource is different from initial activity resource if not specified, assigned to the existing activity resource

Name	Required	Type	Description
fallback_external_id	No	string key	indicates resource, for which the commands will be performed, if the resource specified by 'external_id' is not available (e.g. on holiday etc.). It must not be present when 'external_id' is absent. It functions in the same way as default activity pool: e.g. activity arrives with external_id=A, fallback_external_id=B, default_activity_pool=C - A is checked and if A is available, activity is added to A - if B is checked and if B is available, activity is added to B - if C is checked and if C is available, activity is added to C - if activity is not added it is empty by default
time	-	time	Operation time in YYYY-MM-DD HH:MM:SS format **optional for 'start_activity', 'complete_activity', 'notdone_activity' and 'suspend_activity'** and meaningless for the rest. If not set for these operations, then default is used – current time in time zone of resource. start_time for activity started with 'start_activity', and end_time for activities processed with 'complete_activity', 'notdone_activity' and 'suspend_activity' (for suspended)
appointment	-	struct	**mandatory for activity-related commands** and meaningless for the rest array of 'appointment' structures that contain activity fields for activity update and cancel commands
inventories	-	struct	**mandatory for inventory-related commands** and meaningless for the rest array of 'inventory' structures that contain list of inventories for this resource
userdata	No	string	string, returned in SOAP response without change; not used by the Inbound API, but intended for clients (e.g to store an ID in SOAP request and then quickly find record in SOAP response using this ID) **default value: none**

- **Incremental Upload 'data' Node Example**

```
<data>
    <commands>
       <command>
          <date>2007-06-08</date>
          <type>start_activity</type>
          <time>2011-12-31 23:59:59</time>
          <external_id>53305</external_id>
          <fallback_external_id>AREA01</fallback_external_id>
          <appointment>
          </appointment>
       </command>
       <command>
          <date>2007-06-08</date>
          <type>set_inventory</type>
          <external_id>53305</external_id>
          <inventories>
                <inventory/>
                <inventory/>
                <inventory/>
          </inventories>
       </command>
   </commands>
</data>
```

6.2.3.2 Full Upload 'data' Node

Full Activity Upload and Full Inventory Upload 'data' node have similar structures, and will therefore be described together. For 'head/upload_type' = 'full' and 'head/processing_mode'='appointments_only' - Full Activity Upload 'data' Node. For 'head/upload_type' = 'full' and 'head/processing_mode'='inventory_only' – Full activity Upload 'data' Node.
Both 'data' nodes contain a single child 'providers', which is an array of resources for which the data will be uploaded, as follows:

```
<data>
        <providers>
                <provider></provider>
                <provider></provider>
                <provider></provider>
        </providers>
</data>
```

- **'provider' structure**

The 'provider' structure consists of the following elements:

Name	Required	Type	Description
external_id	Yes	string key	resource's 'external_id' indicates the resource for which the action will be performed
fallback_external_id	No	string key	used if the resource specified by 'external_id' field is not available for some reason (e.g. on holiday etc) **default value:** none
appointments	-	struct	**mandatory for Full Activity Upload** array of 'appointment' structures for activities that are to be uploaded for this resource
inventories	-	struct	**mandatory for Full Inventory Upload** array of 'inventory' structures for inventory that is to be uploaded for this resource
userdata	No	string	string returned in SOAP response without change Not used by the Inbound Interface, but is intended for use by clients (for example a client may use this to store an ID in SOAP request and then quickly find record in SOAP response using such ID). **default value:** none

• **Full Activity Upload 'data' Example**	• **Full Inventory Upload 'data' Example**
<data> <providers> <provider> <external_id>53302</external_id> <appointments> <appointment></appointment> <appointment></appointment> <appointment></appointment> </appointments> </provider> </providers> </data>	<data> <providers> <provider> <external_id>53302</external_id> <inventories> <inventory></inventory> <inventory></inventory> <inventory></inventory> </inventories> </provider> </providers> </data>

6.2.4 'inbound_interface_request' Request Examples

6.2.4.1 Incremental Upload Request Example

```
<Envelope>
  <Body>
    <inbound_interface_request>
      <user>
        <now>2007-06-08T14:46:08+02:00</now>
        <company>bhn10</company>
        <login>root</login>
        <auth_string>f33b5c67b17060091c2a588663b9e99e</auth_string>
      </user>
      <head>
        <upload_type>incremental</upload_type>
        <id>test_incremental_fileupload.xml</id>
        <date>2007-06-08</date>
        <allow_change_date>yes</allow_change_date>
        <appointment>
          <keys>
            <field>appt_number</field>
            <field>customer_number</field>
          </keys>
        </appointment>
        <inventory>
          <keys>
            <field>invsn</field>
          </keys>
        </inventory>
        <provider_group>bhn10_resource_group</provider_group>
      </head>
      <data>
```

```
        <commands>
          <command>
            <date>2007-06-08</date>
            <type>update_appointment</type>
            <external_id>53305</external_id>
            <appointment>
              <appt_number>0001</appt_number>
              <customer_number>Customer_number N85</customer_number>
              <worktype>11</worktype>
              <service_window_start>8:00</service_window_start>
              <service_window_end>19:00</service_window_end>
              <name>name</name>
              <phone>11111111</phone>
              <email>email@email.com</email>
              <cell>11111111</cell>
              <address>address</address>
              <city>city</city>
              <zip>11111</zip>
              <state>CA</state>
              <language>1</language>
              <daybefore_flag>0</daybefore_flag>
              <reminder_time>0</reminder_time>
              <time_zone>2</time_zone>
              <properties>
                <property>
                  <label>MAP_GRID</label>
                  <value>AA11</value>
                </property>
              </properties>
            </appointment>
          </command>
        </commands>
      </data>
    </inbound_interface_request>
  </Body>
</Envelope>
```

6.2.4.2 Full activity Upload Request Example

```
<Envelope>
  <Body>
    <inbound_interface_request>
      <user>
        <now>2007-05-11T18:08:18+02:00</now>
        <company>bhn10</company>
        <login>root</login>
        <auth_string>1dfd3c49964320f0b8de655d0680a7fe</auth_string>
      </user>
      <head>
        <upload_type>full</upload_type>
        <id>full_fileupload.xml</id>
        <date>2007-05-25</date>
        <appointment>
          <keys>
            <field>appt_number</field>
            <field>customer_number</field>
          </keys>
        </appointment>
        <inventory>
          <keys>
            <field>invsn</field>
          </keys>
        </inventory>
        <provider_group>group</provider_group>
        <processing_mode>inventory_only</processing_mode>
      </head>
      <data>
        <providers>
          <provider>
            <external_id>53302</external_id>
            <appointments>
             <appointment>
              <appt_number>0001</appt_number>
              <customer_number>Customer_number N85</customer_number>
              <worktype_label>DEI</worktype_label>
              <service_window_start>8:00</service_window_start>
              <service_window_end>19:00</service_window_end>
              <name>name</name>
              <phone>11111111</phone>
              <email>email@email.com</email>
              <cell>11111111</cell>
              <address>address</address>
```

```
              <city>city</city>
              <zip>11111</zip>
              <state>CA</state>
              <language>1</language>
              <reminder_time>0</reminder_time>
              <time_zone>2</time_zone>
              <properties>
                <property>
                  <label>MAP_GRID</label>
                  <value>AA11</value>
                </property>
              </properties>
            </appointment>
          </appointments>
        </provider>
      </providers>
    </data>
  </inbound_interface_request>
 </Body>
</Envelope>
```

6.2.4.3 Full Inventory Upload Request Example

```
<Envelope>
  <Body>
    <inbound_interface_request>
      <user>
        <now>2007-05-11T18:08:18+02:00</now>
        <company>bhn10</company>
        <login>root</login>
        <auth_string>1dfd3c49964320f0b8de655d0680a7fe</auth_string>
      </user>
      <head>
        <upload_type>full</upload_type>
        <id>full_fileupload.xml</id>
        <date>2007-05-25</date>
        <appointment>
          <keys>
            <field>appt_number</field>
            <field>customer_number</field>
          </keys>
        </appointment>
        <inventory>
          <keys>
            <field>invsn</field>
          </keys>
        </inventory>
        <provider_group>group</provider_group>
        <processing_mode>inventory_only</processing_mode>
      </head>
      <data>
        <providers>
          <provider>
            <external_id>53302</external_id>
            <inventories>
              <inventory>
                <properties>
                  <property>
                    <label>ITEM_NUMBER</label>
                    <value>11111</value>
                  </property>
                  <property>
                    <label>invsn</label>
                    <value>LALALA-23</value>
                  </property>
                </properties>
```

```
            </inventory>
          </inventories>
        </provider>
      </providers>
    </data>
  </inbound_interface_request>
 </Body>
</Envelope>
```

6.3 'inbound_interface_request' Response

The response node contains three mandatory child nodes:

- **'user'**: same information that was sent in the request's 'user' node. The Inbound Interface returns this node without modification.
- **'head'**: contains the same information that was sent in the request 'head' node. The Inbound Interface returns this node without modification.
- **'data'**: results of individual operations such as appointment updates and inventory updates

```
<Envelope>
        <Body>
                <inbound_interface_response>
                        <user>
                        </user>
                        <head>
                        </head>
                        <data>
                        </data>
                </inbound_interface_response>
        </Body>
</Envelope>
```

6.3.1 'inbound_interface_request' Response 'data' Node

The structure of the response 'data' node copies 'data' node of the request with the following differences:

- every element of the response 'data' node must contain its key fields for identifying any 'userdata' nodes present in corresponding items of the request. Other fields may be absent.
- every element of the response may contain a single 'report' node with one or more report messages, that contain an error, a warning, and success descriptions.
- order of the elements in the response may be different from the order of the same elements in request. Elements must be identified by key values or by passing a key to a 'userdata' node.

6.3.1.1 'report' Node

The 'report' node is an array of 'message' structures, each of which contains data on the transaction entry execution result and its specific meaning depends on the structure it appears in (parent node).

- **'message' Structure**

The 'report' node is an array of 'message' structures. At least one 'message' structure must be present. The 'message' structure contains the following elements:

Name	Required	Type	Description
result	Yes	string	brief result of the entry processing; **valid values:** 'error' – the entry processing has failed 'warning' – the entry processing has been completed with result, slightly different from the expected. There may be more than one warning in a report. 'success' – the entry has been successfully processed with the expected result
type	No	string	the field is deprecated and may be empty for some messages. It should not be used for report processing.
code	No	enum	the code of the error or warning in the system
description	Yes/No	string	mandatory description of the error or warning For success messages it can contain some optional information such as record ID in the database.

- **'report' Node Example 1**

The report below contains the success message (shown in green) for an 'update_appointment' command. Note that the 'appointment' node contains only activity key fields (as defined in the 'head/activity/keys' structure) and the 'report' structure:

```
<appointment>
 <appt_number>xx03</appt_number>
 <customer_number>Customer_number N61</customer_number>
 <report>
   <message>
     <description>Appointment id = 6488526 </description>
     <result>success</result>
     <type>update</type>
   </message>
 </report>
</appointment>
```

- **'report' Node Example 2**

The report below contains the success message (shown in green) and warning message (shown in blue) for an 'update_inventory' command. Note that the 'inventory' node contains only inventory key fields (as defined in the 'head/inventory/keys' structure) and the 'report' structure:

```
<inventory>
 <properties>
   <property>
     <label>invsn</label>
     <value>insert_8888</value>
   </property>
 </properties>
 <report>
   <message>
     <description>Inventory serial number = insert_8888 </description>
     <result>success</result>
     <type>update</type>
   </message>
   <message>
     <result>warning</result>
     <type/>
     <code>69117</code>
     <description>Cannot update existing inventory: inserting new- activity
type does not support this</description>
   </message>
 </report>
</inventory>
```

- **'report' Node Example 3**

The report below contains the error message (shown in red) for an 'update_appointment' command.

```
<appointment>
 <appt_number>0013</appt_number>
 <customer_number>Customer_number N54</customer_number>
 <report>
   <message>
     <result>error</result>
     <type/>
     <code>69051</code>
     <description>Property name is empty</description>
   </message>
 </report>
</appointment>
```

- **'report' Node Meaning**

Meaning of a specific 'report' node depends on the structure of its parent node. For example when 'report' appears inside of an 'appointment' structure, then this report contains the result of the activity transaction identified by the keys of the 'appointment' structure. The following structures in the response can contain report nodes:

In	Appears if	Meaning
inventory	always	'report' that appears inside the 'inventory' structure and describes the result of the inventory update command – whether it succeeded or failed and if any warning conditions were encountered
appointment	always	'report' that appears inside of the 'appointment' structure and describes the result of the activity update command – whether it succeeded or failed, and if any warning conditions were encountered
'response' node (root node)	only when an error occurs	when 'report' appears inside the 'inbound_interface_response' structure, it usually contains an error message means that the entire transaction failed and no individual commands were processed this can be a result of: • authentication failure • invalid transaction format • invalid or incompatible head section parameters
'provider'	only when an error occurs	when 'report' appears inside of the 'provider' structure it usually contains an error message means that all activity/inventory commands for the resource failed this can be a result of: • invalid resource 'external_id' • resource has a non-working day or any other conditions that prevent activities from being added to this resource exist

In	Appears if	Meaning
'command'	only when an error occurs	when 'report' appears inside the 'command' structure, it usually contains an error message this means that all activity/inventory commands for this command failed this can be result of: • incorrect command name • invalid command fields formatting • a field listed in transaction 'head/activity/keys' is not specified for activity-related command • a field listed in transaction 'head/inventory/keys' section is not specified for inventory-related command • activity field or property that has its visibility for file-upload the profile set to mandatory is not specified for 'update_activity' • inventory field or property that has its visibility for file-upload the profile set to mandatory is not specified for inventory-related command • command 'external_id' field is not specified for inventory-related command • resource with the 'external_id' specified in the command is not found for inventory-related command.

7 Previous Versions

Backward compatibility between Inbound API in ETAdirect 4.5 has been retained as in versions 4.2, 4.3 and 4.4. The only change is that the 'device' field has gone obsolete. If sent in the request, the field will be ignored.

- **Changed compared to version 4.4:**

– New inventory field 'invtype_label' has been introduced which must contain the inventory type label, when specified
– Support of non-serialized inventory has been implemented
– New 'required_inventories' structure has been introduced which allows defining the required inventories for an activity
– The 'links' structure has been changed to correspond to the new activity links concept

Appendix A: List of Error and Warning Messages

Non-zero error codes for the Inbound Interface are described below, where error is a problem in the transaction that causes the command rejection, and warning is a problem in the transaction with which the command can be executed.

If the error message is not descriptive, additional description is provided in the table below. The errors are listed in the order of their codes.

In case of encountering any error codes not mentioned in the table below, contact support.

No.	E/W	Text	Description and Solution
60080	error	You don't have permission for this action.	Authentication failed. This could be because of one the following: – invalid credentials (user does not exist or password is incorrect) – user does not have permissions set up to access the Inbound API – the <now> date differs from the server time by more than 30 minutes – the <auth_string> is generated incorrectly
69001	error	Wrong version of SOAP request. Expected start node '%s', got '%s'.	Check the input.
69002	error	Mandatory element is empty or absent: '%s/%s'	Check the input for the element referred in the message.
69003	error	'head/upload_type' element is absent or invalid	Check the input.
69005	error	'data/providers' element is absent or empty	Check the input.
69006	error	'data/commands' must not be present for full upload	Check the input.
69007	error	'data/commands' element is absent or empty	Check the input.
69008	error	'data/providers' must not be present for incremental upload	Check the input.
69009	error	'head/processing_mode' has invalid value '%s'. (valid values are '%s','%s')	Check the input.
69010	error	'head/date' is not a YYYY-MM-DD date '%s'	Check the input.
69011	error	'head/date' is in past	Check the input.
69012	error	'head/date' parameter is mandatory for full upload type	Check the input.
69013	error	'head/provider_group' parameter-property not found '%s'	**head/provider_group** if specified, must contain a label of a valid provider property visible to inbound interface. Ensure provider group is a valid property label. See Company Settings → Properties page for the list of properties.

No.	E/W	Text	Description and Solution
69014	error	'head/provider_group' parameter- is not a provider property '%s'	**head/provider_group** if specified, must contain a label of a valid provider property visible to inbound interface. Ensure provider group is a valid property label. See Company Settings → Properties page for the list of properties.
69015	error	'head/appointment/action_if_completed' has invalid value: '%s'	Check the input.
69016	error	'head/appointment/keys' parameter is absent or empty	Valid keys are: appt_number, customer_number, name.
69017	error	'head/appointment/keys' invalid appointment key: '%s'	Valid keys are: appt_number, customer_number, name.
69018	error	'head/inventory/upload_type' has invalid value '%s'. (valid values are '%s','%s')	Set **head/inventory/upload_type** to either full or incremental.
69019	error	'head/inventory/keys' parameter is absent or empty	Valid keys are: invsn, invtype, invtype_label.
69020	error	'head/inventory/keys' invalid inventory key: '%s'	Valid keys are: invsn, invtype, invtype_label.
69021	error	'head/properties_mode' has invalid value: '%s'. (valid values are '%s','%s','%s')	Valid keys are: update, replace.
69022	error	Invalid date format in field '%s': expected '%s' got '%s'	Check the input.
69023	error	Invalid integer format in field '%s': got '%s'	Check the input.
69024	error	Invalid floating point number format in field '%s': got '%s'	Check the input.
69025	error	Internal error in '%s' - PHP interface not implemented	Contact the support.
69026	error	Internal error in '%s' %d	Contact the support.
69027	error	Unexpected end of document	The xml request is invalid. Check the input.
69028	error	Error parsing XML	The xml request is invalid. Check the input.
69029	error	NULL name of element node	Internal error. Contact the support.
69030	error	Node not found: '%s'	Check the input.
69031	error	Child node not found: '%s/%s'	Check the input.
69032	error	Error decoding base64	A file property is not properly encoded in request. Ensure it's encoded in base64.
69033	error	Invalid appointment key name in link: '%s'	Valid keys are: appt_number, customer_number, name
69034	error	Error creating xml-reader	Internal error. Contact the support.
69035	error	Time slot not found: '%s'	Appointment time_slot field must contain valid time slot label. Check the input.
69037	error	Will not set unordered- activity type does not support this	Activity type specified in 'worktype' field does not allow appointment without service window. Change activity type.

No.	E/W	Text	Description and Solution
69038	error	Key field is absent: '%s'	Activity or inventory key is absent. Check the input.
69039	error	Key field is empty: '%s'	Activity or inventory key is empty. Check the input.
69040	error	No key fields specified	Activity or inventory key is absent. Check the input.
69041	error	Inventory key fields are not defined	Internal error. Contact the support.
69042	error	Inventory properties are absent	Inventory has no properties – nothing can be added. Check the input.
69043	error	Inventory key field is empty: '%s'	Activity or inventory key is empty. Check the input.
69044	error	Inventory key field has numeric type but non-numeric value: label='%s' value='%s'	'invtype' field must be a number. Check the input.
69045	error	Inventory key field is absent: '%s'	Activity or inventory key is absent. Check the input.
69046	error	Inventory key is empty	Inventory key is empty. Check the input.
69049	warning	Cannot add link. Appointment not found: '%s'	Link refers to an activity that cannot be found (e.g. it may be in the past). Do not send this link.
69050	error	Provider not found: '%s'	Check the input.
69051	warning	Property name is empty	Check the input.
69052	warning	Invalid property name: '%s'	Property not found in ETAdirect. Check the input or add this property.
69053	warning	Duplicate property: '%s'	There are multiple properties in the same appointment/inventory with the same label. All but one of them (the last one) will be ignored. Remove duplicates from the input.
69054	warning	Invalid file encoding: '%s'. (valid values are '%s', '%s')	'files/file/encoding' has invalid value. Check the input.
69055	warning	File too big: '%s'	The file passed into a file property is bigger than allowed limit of 512000 bytes. File will not be added. Ensure that the files to be sent are not too large.
69056	warning	Could not insert file '%s'	Internal error. Contact the support.
69057	warning	Property is not a file property but sent as file: '%s'	Non-file property is sent in 'files' array. Check the property description.
69058	warning	Invalid MIME type for property: '%s' mime_type='%s'	File property has MIME type that is not in the list of allowed types. Check the property description.
69059	warning	No filename for property: '%s'	File property has no filename. Check the input.
69060	warning	File is empty: '%s'	File property has no file data. Check the input.
69061	warning	Duplicate file property: '%s'	There are multiple file properties in the same appointment/inventory with the same label. All but the last one will be ignored. Remove duplicates from the input.
69062	warning	Property is read-only: '%s'	You have no permission to modify this property. Check permissions in the display profile.

No.	E/W	Text	Description and Solution
69063	warning	Property is not visible: '%s'	You have no permission to modify this property. Check permissions in the display profile.
69064	warning	Duplicate field %s	Non-array field is encountered twice. Check the input.
69065	error	Mandatory field missing: worktype	Check the input.
69066	error	Unknown worktype ID: '%s'	Check the input.
69067	error	Unknown worktype label: '%s'	Check the input.
69068	error	Activity type cannot be created from inbound interface: '%s'	The activity type does not the support 'Allow to create from Incoming interface' option. Check the input / activity type.
69069	error	Invalid phone number format: '%s'	Check the input. Contact the support if you think the phone format is correct.
69070	error	Invalid cell phone number format: '%s'	Check the input. Contact the support if you think the cell format is correct.
69071	error	Unknown time zone id: %d	Check the input.
69072	error	Unknown time zone name: '%s'	Check the input.
69073	error	Unknown language id: %d	Check the input.
69074	error	Unknown language name: '%s'	Check the input.
69075	error	Either 'service_window_start' or 'service_window_end' is absent	Either none or both of fields must be present. Check the input.
69076	error	Field contents is not a time 'service_window_start': '%s'	Check the input.
69077	error	Field contents is not a time 'service_window_end': '%s'	Check the input.
69078	error	Service window start > service window end: '%s' > '%s'	Check the input.
69079	error	Field contents is not a date-time: SLA window'	Check the input.
69080	error	SLA window start > SLA window end: '%s' > '%s'	Check the input.
69081	error	Either 'coordX' or 'coordY' is absent	Either none or both of fields must be present. Check the input.
69082	error	'coordX' is not a real number: '%s'	Check the input.
69083	error	'coordY' is not a real number: '%s'	Check the input.
69084	error	'coordX' is not in range [-180; 180]: '%g'	Check the input.
69085	error	'coordY' is not in range [-90; 90]: '%g'	Check the input.
69086	error	Invalid 'zip' format: '%s'	Check the input. Contact the support if you think zip format is correct.
69087	error	Invalid 'state' format: '%s'	Check the input. Contact the support if you think state format is correct.
69088	error	Invalid 'email' format: '%s'	Check the input. Contact the support if you think email format is correct.
69094	error	Invalid 'reminder_time' format: '%s'	The 'reminder_time' value must be numeric. Check the input.

No.	E/W	Text	Description and Solution
69095	error	'reminder_time' out of range %d [%d; %d]	The 'reminder_time' value must be a number in specified range. Check the input.
69096	error	Time slot is not supported by this activity type	The activity type does not the support 'Support of time slots' option. Check the input / activity type.
69097	error	Time slot is required for this activity type	The 'Support of time slots' option is enabled for the activity type, but no time slot is defined. Check the input / activity type.
69098	error	Will not add inventory- activity type does not support this	The activity type does not the support 'Support of inventory' option. Check the input / activity type.
69099	warning	Will not add/remove links- activity type does not support this	The activity type does not the support 'Support of links' option. Check the input / activity type.
69100	error	Duration field is required for this activity type	The 'Use predefined duration' option is enabled for the activity type, but no duration is defined (duration must be defined in the request). Check the input / activity type.
69101	error	Duplicate inventory key: '%s'	There are two or more inventories for this activity with the same key, OR There are two or more provider inventories in request with the same key. Remove duplicates from the request.
69102	error	Duplicate appointment in transaction: '%s'	There are two or more activities with the same key. When the request contains several commands with the same key, the last command is accepted while others are rejected. Remove duplicates from the request.
69103	warning	Link keys not specified	Check the input.
69104	warning	Duplicate link: '%s'	There are two or more links for this activity with the same key. Remove duplicates from the request.
69105	error	'command/type' is invalid: '%s'	Valid values: 'start_activity'; 'complete_activity'; 'notdone_activity'; 'suspend_activity'; 'update_activity'; 'cancel_activity'; 'delete_activity'; 'set_inventory'; 'update_inventory'; 'delete_inventory'; Check the input.
69106	error	'command/date' is not a 'YYYY-MM-DD' date: '%s'	Check the input.
69107	error	'command/time' is not a 'YYYY-MM-DD HH:MM:SS' datetime: '%s'	Check the input.
69108	error	'command/appointment' cannot be absent for command type: '%s'	Check the input.
69109	error	'command/appointment' cannot be present for command type: '%s'	Check the input.
69110	error	'command/external_id' cannot be absent for command type: '%s'	Check the input.
69111	error	'external_id' is absent	Check the input.
69112	error	Duplicate 'external_id': '%s'	Check the input.
69113	error	Provider does not have any appointments: '%s'	Check the input.

No.	E/W	Text	Description and Solution
69114	warning	Cannot delete link: %s	Internal error (not critical for the transaction success). Contact support.
69115	warning	Cannot add link: %s	Internal error (not critical for the transaction success). Contact support.
69116	warning	Providers are in the same group- will not move	Check if it was the intention.
69117	warning	Cannot update existing inventory: inserting new	Internal error. Contact the support.
69118	error	Cannot delete - inventory does not exist	The inventory does not exist (e.g. has already been deleted) – error can be ignored as there is no more need to delete the inventory.
69119	error	Cannot delete - error getting existing inventory	Internal error. Contact the support.
69120	error	Cannot delete - inventory is assigned to another provider	Specify the correct provider.
69121	warning	Provider is in the same group as original provider - will not move: external_id='%s'	Cannot move inventory. Check if it was the intention.
69122	warning	Falling back to fallback_external_id: %s	Provider specified in 'command/fallback_external_id' cannot be assigned the activity.
69123	warning	Falling back to default pool: %s	Provider specified in 'head/default_appointment_pool' cannot be assigned the activity.
69124	error/ warning	Queue is invalid: %s	See description after the colon and check the input accordingly. When updating an activity and defining an ended queue, activity will be updated but not rescheduled; else the command will fail.
69125	warning	Coordinates are out of bounds - [%g, %g]	Coordinates do not fit into any of bounding rectangles defined for company at Company settings → Business rules → Company boundaries. Check the input / the boundaries.
69126	error	Appointment not found. cannot cancel	The activity cannot be found (e.g. has already been cancelled or deleted) – error can be ignored as there is no more need to cancel the activity.
69127	error	Appointment has been changed during upload	Internal error. Contact the support.
69128	error	'date' is empty	There is no date in command and there is no existing appointment (date does not have to be specified if the activity already exists). Add date field to the command.
69129	error	Appointment status is not 'pending'. cannot update	Activities with status other than 'pending' cannot be updated. Check if it was the intention. Check head/action_if_completed flag value.
69130	error	Unexpected appointment status. cannot update	Internal error. Contact the support.
69131	error	Unexpected value of action_if_completed setting	Internal error. Contact the support.

No.	E/W	Text	Description and Solution
69132	error	No properties to update	Appointment is not pending and there are no properties to update. Can be ignored.
69133	error	Unable to update properties	Internal error. Contact the support.
69134	error	Will not add unscheduled appointment- activity type does not support this	The activity type does not the support the 'Support of not-scheduled activities' option. Check the input / activity type.
69135	error	Date is too far in future	Dates until the end of the next year can be used (e.g. in 2011 the latest valid date is 2012-12-31). Check the input.
69136	error	Will not assign to bucket- activity type does not support this	The activity type does not the support the 'Allow creation in buckets' option. Check the input / activity type.
69137	error	Appointment is not found	Cannot process (cancel, start etc.) activity because it does not exist.
69138	error	Command type is unknown	Type of the command is invalid. Check the input.
69139	warning	Will not move- activity type does not support this	The activity type does not support the 'Allow creation in buckets' option. Check the input / activity type.
69140	warning	Will not reschedule- activity type does not support this	The activity type does not support the 'Allow creation in buckets' option. Check the input / activity type.
69141	error	Appointment status is not 'pending'. cannot start	Activities with status other than 'pending' cannot be started. Check the input.
69142	error	Appointment status is not 'started'	Activities with status other than 'started' cannot be set complete/suspend/notdone. Start the activity or check the input.
69143	error	String property value does not match pattern: label='%s' value='%s' pattern='%s'	Check the input. Check property description (Company settings → Properties → Modify → Pattern).
69144	error	Int property value is non-numeric: label='%s' value='%s'	Check the input. Check property description (Company settings → Properties → Modify → Pattern).
69145	error	Enum property value is reserved: label='%s' value='%s'	Reserved values '0' and '-1' cannot be set for 'enum' property. Check the input. Check property description.
69146	error	Enum property value is inactive: label='%s' value='%s'	Check the input. Check property description (Company settings → Properties → Modify → Pattern).
69147	error	Enum property value is out of range: label='%s' value='%s'	Check the input. Check property description (Company settings → Properties → Modify → Pattern).
69148	error	Property type not supported for property: '%s'	Internal error. Contact support.
69149	error	Mandatory property is absent: '%s'	Property defined as mandatory is missing. Check the input. Check the display profile.
69150	error	Mandatory property is empty: '%s'	Property defined as mandatory is empty. Check the input. Check the display profile.
69151	error	Appointment key has no property description '%s'	Internal error. Contact support.

No.	E/W	Text	Description and Solution
69152	error	The appointment cannot be started at the specified time: min_time='%s' time='%s'	Check the input.
69153	error	User is not active: '%s'	Authentication failed. Check the user record.
69154	error	User is blocked: '%s'	Authentication failed. Check the user record.
69155	warning	Invalid provider preference type: '%s'	Valid values are 'preferred', 'required', 'forbidden'. Check the input.
69156	warning	Provider preference external_id not found: '%s'	Check the input.
69157	warning	Cant have required and preferred providers	The 'required' preference type always prevails over 'preferred', so it is impossible to have both defined for one activity. Check the input.
69158	warning	Cannot delete provider preferences: '%s'	Internal error. Contact support.
69159	warning	Cannot add provider preferences: '%s'	Internal error. Contact support.
69160	error	Error executing commands: '%s'	Internal error. Contact support.
69161	error, warning	Error executing command: '%s'	Internal error (may be not critical for the transaction success). Contact support.
69162	error	Error trying to validate properties: '%s'	Internal error. Contact support.
69163	warning	Company has duplicate time zones with name: '%s'	ETAdirect configuration problem. Contact support. Workaround: use time zone's numeric ID instead of literal name.
69164	warning	Activity has link alerts: flags=%u, desc='%s'	Depends on alert flags value.
69165	error	Will not add teamwork - activity type does not support this	Check activity type.
69166	error	'team_id' field is required for this activity type	Check the input.
69167	error	Invalid provider in field team_id: '%s'	Check the input.
69168	error	Will not change team_id: '%s'	The team_id can only be set when creating new activity and cannot be changed. Ensure team_id in updates is the same as initially defined for the activity or do not send team_id in updates.
69169	warning	Will not move/reschedule teamwork	Activities of a teamwork type cannot be moved or rescheduled. Check the input / activity type.
69170	error	Appointment status is not 'pending'. cannot delete	Only activities of 'pending' status can be deleted. Check the input / activity type.
69171	warning	Will not add/remove provider_preferences- activity type does not support this	The activity type does not the support the 'Support of preferred resources' option. Check the input / activity type.
69172	warning	Some fields could not be updated-changed by user: %s	The fields were changed by ETAdirect user and thus cannot be updated by the Inbound Interface.
69173	warning	Activity is started out of order-moved to the beginning of the queue	Can be ignored if that is the desired behavior.

No.	E/W	Text	Description and Solution
69174	error	Appointment status is not 'pending'. cannot cancel	Activities of status other that 'pending' cannot be cancelled.
69175	error	Both worktype and worktype_label are present	Only one of the two fields should be present in the request.
69176	error	Activity type does not support time slot: '%s'	The activity type does not the support the 'Support of time slots' option. Check the input / activity type.
69177	error	Cannot change status of unscheduled activity	Status of unscheduled activity cannot be changed if the 'date' of the command is not defined.
69178	error	Cannot start unscheduled activity on date='%s' reason='%s'	Trying to start unscheduled activity but cannot move it to a specified date because of provider non-working day or calendar error. Check provider calendar.
69179	error	Will not move/reschedule to invalid queue: %s	Activity cannot be moved to the queue, e.g. because of the resource preferences set for the activity.
69180	error	No inventories to add or delete	The set_inventory command is empty and performs no actions.
69181	warning	time_slot overrides service_window_start/service_window_end	If both service window and time slot have been sent for an activity, the Inbound Interface will ignore the service window values.
69184	warning	Cannot add required inventories	Required inventories cannot be added. The warning message will contain the reason which is one of the following: – Command type is not 'update_activity' – Activity status is not 'pending' – Activity does not have the "required inventory support" feature enabled – Inventory Type specified in the request is invalid – Model specified in the request does not match the model property rules – Required Inventory with this type ID and Model already exists for this activity
69185	error	Invalid inventory type label	Internal error
69186	error	Invalid inventory type ID	Internal error
69187	error	Both invtype and invtype_label are present	Internal error
69188	error	Invalid link type label	Internal error
69189	error	'head/allow_change_date' has invalid value	Internal error
69190	error	Cannot update file property	Internal error occurs at an attempt to update the file property, due to service unavailability or misconfiguration. Contact support.