History SDK

for

Version 4.5

Part Number E62386-01

Table of Content

1 Introduction

1.1 Document Purpose

The document is intended to provide description of the History API used to retrieve the history of operations performed in ETAdirect. The data retrieved by the History API can be further used by external applications.

1.2 Scope of the Document

This document describes the History API request used to retrieve the history records and responses received for such request.

This document describes the functionality implemented in ETAdirect version 4.5.12.

1.3 Target Audience

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

1.4 Glossary

Term	Explanation
Activate route	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 a 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
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
Delivery window	Statistically calculated time period in which a resource is expected to start an activity
ETA	Predicted time at which a resource will arrive at an appointment and start an activity, calculated dynamically from current and historical data
Field	Property present in the system by default
Inventory	Equipment that can be installed or deinstalled during an activity
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
Property	Field and field value assigned to an entity in ETAdirect (to user, resource, activity or inventory). There are fields and custom properties
Route	List of activities assigned to a resource for a specific date, or a list of non-scheduled activities assigned to a resource
Required Inventory	Inventory necessary for the activity completion
Resource	Element in the resource tree representing a defined company asset
Resource External	Company-unique key used to identify a specific resource

ID	
Resource Preference	Definition of a resource as required, preferred or forbidden for a certain activity used to implement the company-specific business logics
REST	Representational State Transfer, architectural style running over HTTP
Service Request	Message generated as a result of 'send Service Request' operation and assigned to a specific entity in ETAdirect
Service Window	Time frame expected by the customer for an activity as scheduled by the company
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)
Time Zones	Customer-specific list of regions with the same time in which the Company operates. Time zones are defined by their difference to UTC.
Work Type	Property that defines a company-specific type of activity (customer activity: install, upgrade, etc.; internal activity: lunch, vehicle maintenance, etc.; teamwork)

2 History API Introduction

ETAdirect logs changes made to activities, inventory, routes, etc. The history records the performed operation, the time of such operation, the user which performed the operation and the actual changes which were made. The History API serves as an advanced means of retrieving such actions and their details for further use by external applications.

Events are logged in history and become available for retrieval as soon as they occur. This allows real-time data collection and processing which creates a dynamic picture of ETAdirect performance.

3 Benefits of Using History API

The functions of the History API are somewhat similar to those of the Outbound API. However, there are certain aspects in which using the History API can be more beneficial:

Outbound API	History API
Requires a Middleware SOAP server for integration between ETAdirect and the client application.	No Middleware SOAP server is required on the customer's side.
More complex configuration involving message step settings.	No message step configuration.
Only actions changing activity statuses ('start_activity', 'complete_activity', etc.) are returned.	In addition to the actions changing activity statuses, changes to the activity properties ('update_activity') are returned.
Supports complex transactions, including changing data in ETAdirect based on Middleware response.	Focused on retrieving history data, does not change ETAdirect state.
Does not specify which properties of the activity, route, etc. were changed.	Specifies the properties changed by the operation.
Wider and more complex functionality.	Simple and easy to use.

In addition, History API serves as a valuable supplement to the Daily Extract functionality. Daily Extract, as its name suggests, extracts ETAdirect data on the daily basis for the entire day and even, when the Overnight functionality is enabled, for the day before. At the same time, the History API is intended for real-time continuous data retrieval which enables immediate action tracking.

4 History API Workflow

In order to start using the History API, the user needs to send the initial request containing only the authentication parameters. For example, the request with HTTP Basic authentication will have the following format:

```
https: //api.etadirect.com/rest/history/v1/route/?
company=test.instance&request_auth_basic
```

If authentication is successful, a response with no history data is returned:

```
{
   "found":true,
   "next_token":"140108-571,0",
```

```
   "history":[]
}
```

The response contains the "next_token" field the value of which is to be used in the subsequent request. The "next_token" value defines the moment in time starting from which historical data will be returned. The GET request with the "token" value is to be sent in the following format:

```
GET /rest/history/v1/route/?company=test.instance&count=150&token=140115-
808,27
```

The request above will return the first 150 (the "count" parameter value) historical records after the moment referenced by the token. The response will also contain a new "next_token" value (referencing the moment in time when the request was sent) to be used in subsequent requests:

```
{
   "found":true,
   "next_token":"140108-571,0",
   "history":[
      HISTORY RECORDS
   ]
}
```

This procedure allows retrieving a continuous flow of history records starting from the time when the initial request was sent.

4.1 History Data Retrieval

As the History API is intended to provide continuous real-time data flow, the recommended history data retrieval routine is as follows:

- the GET request is to be sent with the "token" parameter set to the "next_token" obtained from the initial request
- the response will contain the history data available in the system from the moment the token was generated, and a new "next_token" value
- if the request returns less records than requested (the "count" parameter value), it means that the response contains all data available for retrieval at the moment. The next request with the "next_token" is to be sent after a certain pause to allow new data to accumulate (the recommended interval is 10 seconds)
- if the request returns the same number of records as was requested, it means, with a high degree of probability, that there are more records available. Therefore, a new GET request is to be sent immediately with the "next_token" received in the response to retrieve the next batch of the history data. The process is to be repeated until all data is retrieved.

Note: The History API uses the database storing history records for 3 months. Any records accumulated in the database over the period of 3 months can be retrieved. The maximum number of records retrieved by a single request is 1,000 (the "count" parameter of the request).

5 Authentication

5.1 History API Authentication Methods

In order to start using the History API, the user has to perform authentication, that is, to verify their identity and authorization to use the API.

The History API supports the following methods of authentication:

- Authentication using authentication token
- HTTP Basic (authentication fields are sent as part of a standard "Authorization" HTTP header). This method can be used to access APIs directly from WEB browsers.

5.1.1 Authentication Using Authentication Token

The History API can authenticate by sending two parameters:

- **company** (query parameter containing the instance name)
- **toa-token** (query parameter or HTTP header, containing a base64-encoded JSON-object with the following fields):

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

Both parameters can be added to the query string as follows:

```
https: //api.etadirect.com/rest/history/v1/route/?company=test.instance&toa-token=eyAibm93IiA6ICIyMDE0LTA1LTI3VDEyOjU2OjA4KzAzMDAiLCAiY29tcGFueSIgOiAieWFtYXRvIiwgImxvZ2luIiA6ICJzb2FwIiwgImF1dGhfc3RyaW5nIiA6ICJiNTA1NTNlYTFiNWZlOGFjNGRjYWEyMWZkNTk4YTljNCIgfQ==
```

Or, the 'company' parameter can be added to the query with the 'toa-token' sent as a custom HTTP header as follows:

```
GET /rest/history/v1/route/?company=test.instance HTTP/1.0
User-Agent: curl/7.31.0
Host: api.etadirect.com
Accept: */*
toa-token:
eyAibm93IiA6ICIyMDE0LTA1LTI3VDEyOjU2OjA4KzAzMDAiLCAiY29tcGFueSIgOiAieWFtYXRvIiwgImxvZ2luIiA6ICJzb2FwIiwgImF1dGhfc3RyaW5nIiA6ICJiNTA1NTNlYTFiNWZlOGFjNGRjYWEyMWZkNTk4YTljNCIgfQ==
```

Note: This authentication method is the most recommended for use as the most reliable and secure.

5.1.1.1 "toa-token" Construction

Below is the example of construction of 'toa-token' used in the History API authentication for the following credentials:

- instance name: "test.instance"
- login: "api.user"
- password: "1"

The "now" parameter is generated from the client's current date and time in ISO 8601 format, for example: "2014-02-05T15:45:02Z". The "auth_string" is a password hash in MD5 form (CONCATENATE(now , MD5(password))):

```
MD5("2014-02-05T15:45:02Z",MD5("1")) => "4b7337af5af07bd66157bf1d06dc7a3c"
```

The "company" field should be set to instance name ("test.instance" in the example). The following fields should be encoded in a JSON object:

```
{
  "now"         : "2014-02-05T15:45:02Z",
  "company"     : "test.instance",
  "login"       : "api.user",
  "auth_string" : "4b7337af5af07bd66157bf1d06dc7a3c"
}
```

And the resulting JSON object should be base64-encoded:

```
ICAgIHsKICAgICAgIm5vdyIgICAgICAgICA6ICIyMDE0LTAyLTA1VDE1OjQ1OjAyWiIsCiAgICAg
ICJjb21wYW55IiAgICAgOiAidGVzdC5pbnN0YW5jZSIsCiAgICAgICJsb2dpbiIgICAgICAgOiAi
YXBpLnVzZXIiLAogICAgICAiYXV0aF9zdHJpbmciIDogIjRiNzMzN2FmNWFmMDdiZDY2MTU3YmYx
ZDA2ZGM3YTNjIgogICAgfQ==
```

Example of the 'toa-token' construction in PHP language using the above-mentioned parameters:

```
// input
$company  = 'test.instance';
$login    = 'api.user';
$password = '1';
// algorithm
$now = gmdate('c');
$toa_token = base64_encode(
   json_encode(
       array(
          'now'         => $now,
          'company'     => 'test.instance',
          'login'       => 'api.user',
          'auth_string' => md5($now . md5($password)),
       )
   )
);
// output
print($toa_token);
```

5.1.2 Authentication Using HTTP Basic

Note: According to its standard, HTTP Basic authentication method involves sending the password without any protection, therefore, it is not recommended over non-encrypted channels. HTTP Basic authentication can be used for testing the API from the browser.

HTTP Basic authentication involves one parameter added to the History API URL query string:

- **company** (parameter containing the instance name)

Also, a standard HTTP header is sent in the request:

- **Authorization** (header containing a base64-encoded string consisting of the login and a plain-text password)

The Authorization header value is constructed by concatenating the login and password with the special field delimiter and the ":" symbol. Below is the example of the Authorization header construction for the following credentials:

- instance name: "test.instance"
- login: "api.user"
- password: "SECRET"

These fields are concatenated as follows:

```
CONCATENATE("api.user", ":", "SECRET") = "api.user:SECRET"
```

The resulting string is then base64-encoded as follows:

```
BASE64("api.user:SECRET") = "YXBpLnVzZXI6U0VDUkVU"
```

The string obtained as the result is used as the Authorization in the HTTP Basic header.

Example of HTTP Basic header containing the Authorization string and company query parameter:

```
GET /rest/history/v1/route/?company=test.instance HTTP/1.0
User-Agent: curl/7.31.0
Host: api.etadirect.com
Accept: */*
Authorization: YXBpLnVzZXI6U0VDUkVU
```

Authenticating using HTTP Basic can be done directly from the WEB browser. For that purpose, the user has to enter the History API URL in the browser address bar and append the special parameter 'request_auth_basic':

```
https: //api.etadirect.com/rest/history/v1/route/?
company=test.instance&request_auth_basic
```

The user is then requested to enter their ETAdirect login and password in the dialog window. If authentication is successful, the user is able to use the API.

More information on HTTP Basic authentication method can be found at http://en.wikipedia.org/wiki/Basic_access_authentication.

5.1.3 Authentication Failure

Authentication fails if:

'now'	is different from the current time on the server and this difference exceeds the predefined time-window (30 minutes by default)
'company'	cannot be found in the ETAdirect
'login'	cannot be found for this company
user with this 'login' is	not authorized to use the method
'auth_string'	is not equal to md5(now+md5(password))
	For example: *'now' = "2005-07-07T09:25:02+00:00" and password = "Pa$$w0rD"* *t h e n* *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 History API Permission

Access to the History API is controlled by the permission defined in the Manage Application → Permissions → API.

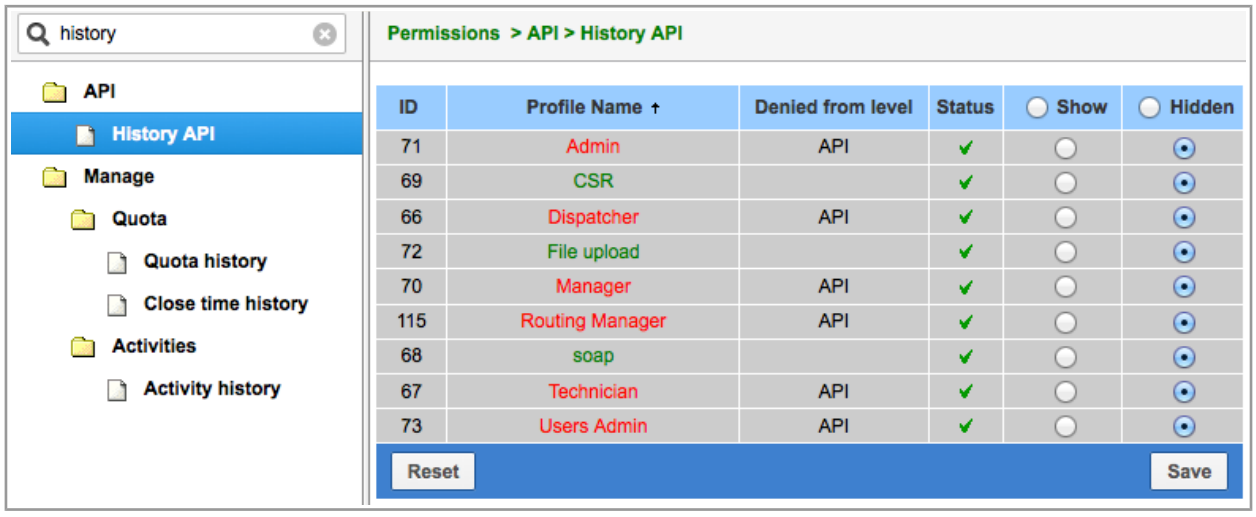

Figure 1: History API permission

Each user is able to retrieve the history records available to the resource assigned to such user.

7 History API Operation Description

7.1 Objects Monitored by History

The History API returns history records of changes to the following types of objects of ETAdirect:

- route (one calendar day of one resource with a list of scheduled or non-scheduled activities assigned to the resource)
- activity (time-consuming action performed by a resource)
- activity link (correlation between start and end of two activities)
- resource preference (rules defining required, preferred or forbidden resources for a certain activity. Resource preferences determine activity assignment)
- required inventory (inventory necessary to complete a certain activity)
- inventory (equipment installed or deinstalled during an activity performance)
- request (message generated as a result of 'send Service Request' operation and assigned to a specific entity in ETAdirect)

7.2 History API Request

The History API uses a single method, namely, the GET method, to retrieve all history for all objects in ETAdirect which were updated within the time elapsed since the previous token issuance. The GET request has the following format:

```
GET /rest/history/v1/route/?company=test.instance&count=150&token=140115-808,27
```

where:

- company – company name as used in authentication
- count – the maximum number of records to be returned in the response (the maximum value: 1000, the default value: 100)
- token – string defining the point from which history is to be returned. The token is created at time of the initial valid request and remains valid for the next 3 months. Each subsequent response contains a new token. If a request is sent with the same token as the previous request, the response contains the history returned for the previous request plus all changes logged after that. However, if a request is sent with a token for which the 3-months validity period has expired, the request is not processed. If a request is sent with the new token, the response contains changes logged after such token creation.

Optionally, the request can contain the 'debug' parameter allowing data to be returned in human-friendly format. Otherwise, the data is returned as a single string.

The 'debug' parameter use is recommended for testing the functionality. In this case the optimum scenario includes authentication using the HTTP Basic method which requires no complex configuration

and a request sent with the 'debug' parameter which will allow the user to view the response in the easily readable format. The user is then able to check whether the returned data is complete and correct.

Note: a response in such format will not be processed by the external application.

7.3 History API Response

A valid GET request returns a response containing the records of actions performed on ETAdirect objects, up to the "count" number defined in the request. A single response may contain records on different objects update. The response structure is described below.

Each History API response always contains a package header.

The package header contains the request result ('true' for a valid request and 'false' for an invalid one) and the next token which can be used in subsequent history requests. The package header is followed by the history records.

Note: for the sake of clarity, all examples below are shown in the human-friendly format.

```
{
   "found":true,
   "next_token":"140108-571,0",
   "history":[
      HISTORY RECORDS
   ]
}
```

The history data returned in a response is organized in records, each containing one change to an ETAdirect object. Each record has a record header consisting of the following fields:

- "operation" – name of the operation logged in the history
- "action_time" – time of the action
- "user" – ETAdirect user performing the action. If the action was performed by the system, the "user" field is omitted

```
{
    "operation": "create_route",
    "action_time": "2014-01-15 13:36:54",
    "user": "admin",
    "route": {
        "date": "2014-01-15",
        "resource_id": "33036",
        "changes": {
            "activated": "2014-01-15 13:36:00",
            "time_zone": "Eastern",
            "calendar_time_from": "2014-01-15 12:00:00",
            "calendar_time_to": "2014-01-15 21:00:00",
            "calendar_points": "100"
        }
```

```
    }
}
```

The header is followed by the name of the object, the fields identifying the particular object and the description of changes performed to the object.

7.3.1 History API Response Description

Below is the description of history records logged for each ETAdirect object monitored by history.

Note: the History API logs the same operations which are used in respect of the corresponding objects by the Mobile Client API.

7.3.1.1 Route Update

The following route operations are logged in the history and returned by the History API:

- create_route (route formation in ETAdirect occurring when activities are assigned to a resource or when the resource having no assigned activities activates their route to start their working day)
- update_route (change of the resource's working calendar)
- start_route (start of the resource's working day by activating the route)
- end_route (end of the resource's working day by deactivating the route)
- restart_route (reopening of the resource's route by reactivating the previously deactivated route)

Each route-related record contains the route identifier fields defining the route to which the history record is related:

Field	Description
date	route date in the YYYY-MM-DD format
resource_id	external ID of the resource to which the route is assigned

The records may optionally contain the "changes" field containing the fields and their values changed for the route. The "changes" field logs updates of the following fields:

Field	Description
activated	date and time of the route activation in the YYYY-MM-DD HH:MM:SS format
deactivated	date and time of the route deactivation in the YYYY-MM-DD HH:MM:SS format
calendar_time_from	date and time of start of the resource's calendar assigned to the route in the YYYY-MM-DD HH:MM:SS format
calendar_time_to	date and time of end of the resource's calendar assigned to the route in the YYYY-MM-DD HH:MM:SS format
calendar_points	number of points assigned to the resource for the route (if the company uses points)
time_zone	the resource's time zone **Note**: all times are returned in GMT.
traveling_time	time from the end of the last activity in the route till the arrival to the final location (contains the actual traveling time value if the 'Working time includes travel from last activity' option is enabled for the resource, otherwise contains '0')

- **Route Update Response Example**

The example below shows the history record of an activated route creation:

```
{
    "operation": "create_route",
```

```
    "action_time": "2014-01-15 13:36:54",
    "user": "admin",
    "route": {
        "date": "2014-01-15",
        "resource_id": "33036",
        "changes": {
            "activated": "2014-01-15 13:36:00",
            "time_zone": "Eastern",
            "calendar_time_from": "2014-01-15 12:00:00",
            "calendar_time_to": "2014-01-15 21:00:00",
            "calendar_points": "100"
        }
    }
}
```

7.3.1.2 Activity Update

The following activity operations are logged in the history and returned by the History API:

- create_activity (formation of a new activity)
- update_activity (change of the activity properties)
- start_activity (change of the activity status to 'started' meaning that the resource has begun performing the activity)
- suspend_activity (change of the activity status to 'pending' and a simultaneous creation of a new activity with the 'suspended' status is created duplicating the original activity)
- complete_activity (change of the activity status to 'complete' meaning a successful performance of the activity)
- notdone_activity (change of the activity status to 'notdone' meaning that a started activity cannot be successfully completed for some reason)
- cancel_activity (change of the activity status to 'cancelled' meaning that the activity has not been started and will not be performed)
- delete_activity (removal of a cancelled activity from the system. An activity can be deleted only from an inactive route)
- delay_activity (change of the end time of a started activity to extend the activity duration)
- reopen_activity (creation of a 'pending' activity duplicating a previously completed, cancelled or not-done activity)
- prework_activity (creation of an additional activity necessary for performance of a pending activity. Preworks are always created in the 'started' status)

For the following operations the identifier fields contain old values and the "changes" field contains new values:

- move_activity (activity reassignment to a different resource and/or date)
- reschedule_activity (activity move to a different date and/or time)

Each activity-related record contains the activity identifier fields:

Field	Description
date	date of the route to which the activity is assigned in the YYYY-MM-DD format
resource_id	external ID of the resource to which the activity is assigned
activity_id	internal activity ID
appt_number	activity number
customer_number	customer's account number

The records may optionally contain the "changes" field containing the fields and their values changed

for the activity. The "changes" field logs updates of activity fields and custom properties. In addition to the fields listed in the table below, changes to other activity fields existing in the system and custom activity properties defined in the specific company can also be returned.

Field	Description
type	activity type
status	activity status
worktype	activity work type
workzone	activity work zone
duration	activity duration in minutes
time_slot	label of activity time slot
service_window_start	customer service window start date and time in the YYYY-MM-DD HH:MM:SS format
service_window_end	customer service window end date and time in the YYYY-MM-DD HH:MM:SS format
delivery_window_start	activity delivery window start time in the HH:MM:SS format
delivery_window_end	activity delivery window end time in the HH:MM:SS format
name	customer's name
phone	customer's regular (land) phone number
email	customer's email address
address	customer's address
city	customer's city of residence
zip	customer's zip/post code
state	customer's state of residence
language	notification language company specific language label (en, es, etc.)
reminder_time	reminder notification time: how many minutes before the activity start time the customer should be notified
time_zone	customer's time zone **Note**: all times are returned in GMT.
coord_status	whether or not activity coordinates were found
coordx	latitude of the activity (of the customer's location)
coordy	longitude of the activity (of the customer's location)
start_time	ETA time (for started and ended activities – time when the activity was started) in YYYY-MM-DD HH:MM:SS format
end_time	predicted or actual end time of activity in YYYY-MM-DD HH:MM:SS format
team_id	external ID of the team-holder – the head resource within a team
unordered	parameter defining that there is no specific time within the resource's route when the activity has to be performed
position_in_route	number of the activity in the route

- **Activity Update Response Example**

The example below shows the history record of a started activity creation:

```
{
    "operation": "create_activity",
    "action_time": "2014-01-15 16:34:29",
    "user": "admin",
    "activity": {
        "date": "2014-01-15",
        "resource_id": "33011",
        "activity_id": 3998009,
        "changes": {
            "ACTIVITY_NOTES": "just lunch",
            "type": "regular",
            "status": "started",
            "worktype": "LU",
            "duration": "60",
            "start_time": "2014-01-15 16:34:00",
            "time_slot": "lunch",
            "service_window_start": "2014-01-15 20:00:00",
            "service_window_end": "2014-01-15 20:30:00",
            "language": "en",
            "time_zone": "Pacific",
            "position_in_route": "1",
            "time_of_booking": "2014-01-15 16:34:29",
            "time_of_assignment": "2014-01-15 16:34:29"
        }
    }
}
```

The example below shows the history record of an activity move between resources. In this case, the identifier field "resource_id" contains the old value, while the "resource_id" field in "changes" contains the new one:

```
{
    "operation": "move_activity",
    "action_time": "2014-01-15 17:34:28",
    "user_id": "admin",
    "activity": {
        "date": "2014-01-15",
        "resource_id": "33015",  // old resource id
        "activity_id": 3956550,
        "appt_number": "#137163458",
        "customer_number": "019942164",
        "changes": {
            "resource_id": "33011"  // new resource id
        }
```

```
    }
},
{
    "operation": "update_activity",
    "action_time": "2014-01-15 17:34:28",
    "user": "admin",
    "activity": {
        "date": "2014-01-15",
        "resource_id": "33011",  // new resource id
        "activity_id": 3956550,
        "appt_number": "#137163458",
        "customer_number": "019942164",
        "changes": {
            "duration": "30",
            "start_time": "2014-01-16 00:13:52",
            "traveling_time": "11",
            "delivery_window_start": "2014-01-15 23:30:00",
            "delivery_window_end": "2014-01-16 00:45:00",
            "position_in_route": "12",
            "time_of_assignment": "2014-01-15 17:34:28"
        }
    }
}
```

7.3.1.3 Activity Link Updates

The following activity link operations are logged in the history and returned by the History API:

- link_activities (creation of a link between two activities)
- unlink_activities (removal of a link between two activities)

Each activity link-related record contains the link identifier fields:

For the first activity:

Field	Description
link_type	label of the link type
from_activity	structure containing identifier fields of the activity from which the link is established
to_activity_id	internal ID of the activity to which the link is established
to_appt_number	number of the activity to which the link is established

For the second activity:

Field	Description
link_type	label of the link type
to_activity	structure containing identifier fields of the activity to which the link is established
from_activity_id	internal ID of the activity from which the link is established
from_appt_number	number of the activity from which the link is established

Note: for each activity link created or removed, the History API response contains two records – for the first activity in the link and for the second activity.

- **Activity Link Update Response Example**

The example below shows the history record of a link creation.

```
{
    "operation": "link_activities",
    "action_time": "2014-01-15 16:54:48",
    "user": "admin",
    "activity_link": {
        "link_type": "start-before",
        "from_activity": {
            "date": "2014-01-15",
            "resource_id": "33003",
            "activity_id": 3956464,
            "appt_number": "#137163544",
            "customer_number": "019922286"
        },
```

```
        "to_activity_id": 3954821,
        "to_appt_number": "#137165187"
    }
},
{
    "operation": "link_activities",
    "action_time": "2014-01-15 16:54:48",
    "user": "admin",
    "activity_link": {
        "link_type": "start-after",
        "to_activity": {
            "date": "2014-01-15",
            "resource_id": "routing",
            "activity_id": 3954821,
            "appt_number": "#137165187",
            "customer_number": "019911355"
        },
        "from_activity_id": 3956464,
        "from_appt_number": "#137163544"
    }
}
```

7.3.1.4 Resource Preference Updates

The following resource preference operations are logged in the history and returned by the History API:

- set_resource_preference (definition of resources preferred, required or forbidden for an activity)
- del_resource_preference (deletion of any previously set resource preferences)

Each preference-related record contains the resource preference identifier fields:

Field	Description
activity	structure containing identifier fields of the activity for which resource preferences are set or deleted
type	resource preference type (preferred, forbidden, or required
resource_id	external ID of the resource for which preferences are set or deleted

- **Resource Preference Update Response Example**

The example below shows the history record of a preferred resource adding:

```
{
    "operation": "set_resource_preferences",
    "action_time": "2014-01-15 17:05:16",
    "user": "admin",
    "preference": {
        "activity": {
            "date": "2014-01-15",
            "resource_id": "33003",
            "activity_id": 3956464,
            "appt_number": "#137163544",
            "customer_number": "019922286"
        },
        "resource_id": "11129",
        "type": "preferred"
    }
}
```

7.3.1.5 Required Inventory Updates

The following required inventory operations are logged in the history and returned by the History API:

- create_required_inventory (adding required inventory to an activity)
- update_required_inventory (changing properties of required inventory)
- delete_required_inventory (removing required inventory from an activity)

Each required inventory-related record contains the required inventory identifier fields:

Field	Description
activity	structure containing identifier fields of the activity for which required inventory is set, updated or removed
type	required inventory type
resource_id	required inventory model

The records may optionally contain the "changes" field containing the "quantity" field, if the inventory quantity was changed in the operation (for non-serialized inventory).

- **Required Inventory Update Response Example**

The example below shows the history record of required inventory adding to an activity:

```
{
    "operation": "create_required_inventory",
    "action_time": "2014-01-28 12:14:02",
    "user": "admin",
    "required_inventory": {
        "type": "Wire",
        "model": "RG-45",
        "activity": {
            "date": "2014-01-28",
            "resource_id": "33035",
            "activity_id": 3954885,
            "appt_number": "137165123",
            "customer_number": "019921925"
        },
        "changes": {
            "model": "RG-45",
            "quantity": "5"
        }
    }
}
```

7.3.1.6 Inventory Updates

The following inventory operations are logged in the history and returned by the History API with the activity identifier:

- install_inventory (moving the inventory from the 'resource' pool to the 'install' pool)
- deinstall_inventory (moving the inventory from the 'customer' pool to the 'deinstall' pool)
- create_customer_inventory (adding inventory to the 'customer' pool)
- delete_inventory (removing inventory from the 'customer' pool)
- undo_install_inventory (moving the inventory from the 'install' pool to the 'resource' pool)
- undo_deinstall_inventory (moving the inventory from the 'deinstall' pool to the 'customer' pool)
- exchange_inventory_install (moving the inventory from the 'resource' pool to the 'install' pool in an 'exchange_inventory' operation)
- exchange_inventory_deinstall (moving the inventory from the 'customer' pool to the 'deinstall' pool in an 'exchange_inventory' operation)
- update_customer_inventory (change of customer inventory properties)

> **Note**: 'exchange_inventory_install' and 'exchange_inventory_deinstall' are two records of the same 'exchange_inventory' operation but logged for each of the two inventories involved in the exchange.

The following inventory operations are logged in the history and returned by the History API without the activity identifier:

- create_resource_inventory (adding inventory to the 'resource' pool)
- update_resource_inventory (change of resource inventory properties)
- delete_resource_inventory (removing inventory from the 'resource' pool)

Each inventory-related record contains the inventory identifier fields:

Field	Description
activity	structure containing identifier fields of the activity to which the inventory is assigned **Note**: the 'activity' structure' is returned only for certain inventory operations (see above)
inventory_id	inventory ID
type	inventory type
serial_number	inventory serial number (for serialized inventory)

The records may optionally contain the "changes" field containing the fields and their values changed for the inventory. The "changes" field logs updates of inventory fields and custom properties. In

addition to the fields listed in the table below, changes to other inventory fields existing in the system and custom inventory properties defined in the specific company can also be returned.

Field	Description
status	inventory pool
quantity	number of inventories (for non-serialized inventory only)

- **Inventory Update Response Example**

The example below shows the history record of inventory installation:

```
{
    "operation": "install_inventory",
    "action_time": "2014-01-15 17:11:57",
    "user": "admin",
    "inventory": {
        "activity": {
            "date": "2014-01-15",
            "resource_id": "33003",
            "activity_id": 3956439,
            "appt_number": "#137163569",
            "customer_number": "019911470"
        },
        "inventory_id": 20998758,
        "type": "NT",
        "status": "install",
        "serial_number": "0001757132",
        "changes": {
            "status": "install",
            "EQUIPMENT_ROOM_CODE": "2354235488"
        }
    }
}
```

The following example shows the history record of customer inventory creation:

```
{
    "operation": "create_customer_inventory",
    "action_time": "2014-01-28 11:24:32",
    "user": "admin",
    "inventory": {
        "serial_number": "HRSC636029",
        "type": "NT",
        "status": "customer",
        "inventory_id": 21034416,
        "activity": {
            "date": "2014-01-28",
            "resource_id": "33003",
```

```
            "activity_id": 3954809,
            "appt_number": "#137165199",
            "customer_number": "019901104"
        },
        "changes": {
            "status": "customer",
            "type": "NT",
            "serial_number": "HRSC636029",
            "quantity": "1",
            "EQUIPMENT_ROOM_CODE": "test"
        }
    }
}
```

7.3.1.7 Request Creation

The following request operations are logged in the history and returned by the History API:

- create_customer_request (creation of a service request assigned to an activity)
- create_inventory_request (creation of a service request assigned to an inventory)
- create_resource_request (creation of a service request assigned to a resource)

Each request-related record contains the request identifier fields:

Field	Description
type	request type
date	request date
resource_id	external ID of the resource associated with the request

Note: depending on the request type, the identifier fields may also contain activity (for customer requests) and inventory (for inventory requests) identifier structures.

The records may optionally contain the "changes" field containing the fields and their values changed for the service request. The "changes" field logs updates of any service request custom properties existing in the system.

- **Request Creation Response Example**

The example below shows the history record of a customer request creation:

```
{
    "operation": "create_customer_request",
    "action_time": "2014-02-05 12:04:24",
    "user": "admin",
    "request": {
        "type": "SR",
        "date": "2014-02-06",
        "resource_id": "routing",
        "activity": {
            "activity_id": 3952162,
            "appt_number": "#137167846",
            "customer_number": "019892755"
        },
        "changes": {
            "sr_body": "asfd"
        }
    }
}
```

The following example shows the history record of an inventory request creation:


```
{
    "operation": "create_inventory_request",
    "action_time": "2014-02-05 13:42:57",
    "user": "admin",
    "request": {
        "type": "SR",
        "date": "2014-02-05",
        "resource_id": "33015",
        "activity": {
            "activity_id": 3954828,
            "appt_number": "137165180",
            "customer_number": "019946338"
        },
        "inventory": {
            "serial_number": "7213125210",
            "type": "TV",
            "inventory_id": 20994113
        },
        "changes": {
            "sr_subject": "asdfd",
            "sr_body": "asdf"
        }
    }
}
```