

Booking Software & Building Controlling System
Data Exchange Protocol
Nordic Standard

Draft 1
DocumentVersion 10
2016-10-31

Companies that conforms to this standard/document

Name	Country	Date	Contact	Color
Regler Och Webbteteknik Sverige AB	Sweden	2016-09-21	Robin Andersson robin@rows.se	Yellow
Agrando Sweden AB	Sweden	2016-09-22	Rune Hansen rune.hansen@agrand.no	Blue
LJ System AB	Sweden	2016-09-23	Niklas Falemar niklas@ljsystem.se	Orange
Eniac Data AB	Sweden	2016-09-23	David Fröjmark david.frojmark@eni.se	Green
JEFF Electronics AB	Sweden	2016-09-26	Björn Granbom bjorn@jeff.se	Red
Mild Media AB	Sweden	2016-09-27	Albin Nystedt	Purple

Every change a company want to make to this document must be highlighted with corresponding color. When the change is unanimous and democratically decided to stay the color will be removed, and (very important) the fixed change must be noted/listed in the “Document version notes” section. Both of who asks and the one who approved the change must be noted.

Introduction

The goal with this standard is to simplify and unify the data exchanges between booking administration softwares and building controlling systems.

Abbreviations and explanations

Booking administration software: is referred to an administration software where a user has the ability to schedule the use of one or more resources.

In this document Booking Administration Software will be shorted to BAS.

Building controlling systems: is refereed to systems that in one way or another controls a building and/or its surroundings.

In this document Building Controlling Systems will be shorted to BCS.

Data Transfer

The data are to be transferred with HTTP or HTTPS as a POST method.

The URL shall always be the same regardless of what method is invoked.

All transfers are initiated by the BCS.

Thus all requests will be made by a BCS and all responses comes from a BAS.

BAS is holder of the API and the BCS is a user of the API.

HTTP Keep-Alive are preferred.

Request/response payload

The payload data must be formatted as valid **JSON**.

The JSON object must be passed as the body of the POST request.

All variable names are **case sensitive**.

The order of variables is **not predetermined** and can be in any order within the JSON object.

All ID values must be valid GUID/UUID string.

An exception is the instance id for every booking.

The ID must have the format as a GUID/UUID but the generation of the id does not need to follow the UUID standard.

Every request must include a variable called “clientID” which contains a unique identification of the BCS system so that the BAS can filter unwelcome requests.

The JSON object must be passed as the body of the POST request.

Every BCS generates it’s own appID and passes it to every connected BAS.

The same appID can be used for several BAS.

Every request must also include a variable called “clientHash”.

The purpose of the hash is to verify the BCS and it’s appID.

The BCS must generate a hash using HMAC-SHA1 hash.

The hash contains a secret key and the systemTime (no secret).

The secret key is a UUID generated by the BCS and passed to BAS together with the clientID upon the first contact.

The secret key **MUST** be unique between BAS.

Generation of the clientHash

The hash contains a secret clientKey and some none secret data.

The secret clientKey is a generated UUID by BCS and is passed to the BAS administrator.

A BCS must have a unique clientKey for every BAS company.

```
message = [systemTime] + [appID] + [method]
```

```
clientHash = HMAC-SHA1(appKey[16], message)
```

message			
Name	Type	Description	Example
systemTime	String	UInt64 epoch SystemTime in seconds. If the time deviates more than +-10 minutes from BAS clock, the request must be ignored.	1475226019
appID	String	BCS appID expressed as a GUID string.	9818d49a-005d-4a83-93b3-9de04a6a5225
method	String	Current invoked method. This must be case sensitive.	GetCustomerData
Example: 14752260199818d49a-005d-4a83-93b3-9de04a6a5225GetCustomerData			

The calculated clientHash is then expressed as hex in the request, together with systemTime and clientID.

The result should match the following:

systemTime: "1475226019"

clientID: "9818d49a-005d-4a83-93b3-9de04a6a5225"

method: "GetCustomerData"

clientKey: "5878b222-9781-4e1b-936f-ef9ccad60518"

message: "14752260199818d49a-005d-4a83-93b3-9de04a6a5225GetCustomerData"

clientHash: "a43074f5bfa356d3be7dda51ef2e4016292743bd"

Fetching customer data

This method requests information about the customer and their resources.

REQUEST

Name	Description	Note
method	Requested method. Every request must have this. This must be case sensitive.	
client	Information about the BCS client.	
client.api	<p>Request version number as signed integer. This is to ensure that future changes that break backward compatibility does not destroy existing connections.</p> <p>The “version” value is divided into three parts. Part 1: API Level implementation. (see description) Part 2: Method version. Is only to be changed if backward compatibility is broken. Part 3: Documentation standard version.</p> <p>The BAS must return status.code = 460 or 461 if the version of part 1 or 2 is unknown. (See status.code section below) Part 3 should not create errors.</p>	
client.id	A unique identification of the BCS system. Must be GUID/UUID compatible. Every single BCSs should be able to have the same clientID between several BASs.	
client.time	Epoch time used for clientHash.	
client.hash	The resulting hash expressed in hex.	
payload	Object containing data for the method.	
payload.customers	A GUID/UUID list of requested customers. The customer id is determined in BAS and handed to BCS prior to installation.	

Example request:

```
{
  "method": "GetCustomerData",
  "client": {
    "api": "1.1.10",
    "id": "9818d49a-005d-4a83-93b3-9de04a6a5225",
    "time": 1475226019,
    "hash": "a43074f5bfa356d3be7dda51ef2e4016292743bd"
  },
  "payload" : {
    "customers": ["e3941203-37c8-4aaf-a10c-a46100ccb787"]
  }
}
```

RESPONSE

Name	Description	Note
status	Must exist in all responses.	
status.code	<p>Signed integer. Based on the HTTP standard codes (https://en.wikipedia.org/wiki/List_of_HTTP_status_codes). Future additions to the codes should if possible be based on it's HTTP counterpart or selected so that it doesn't conflict with any known HTTP code.</p> <p>Success 200 = OK (<i>All is good and data returned as requested.</i>) 201 = Created (<i>Data received and processed.</i>) 202 = Accepted (<i>Data received and are still being processed.</i>) 204 = No Content (<i>Request OK but no data returned.</i>) 299 = OK Deprecated (<i>Same as 200 except the method will be removed in next level. Alternative method might be suggested in status.msg.</i>)</p> <p>Client Error 400 = Bad Request (<i>Invalid JSON in request.</i>) 401 = Unauthorized (<i>Unknown clientID or invalid token.</i>) 405 = Method Not Allowed (<i>Requested method does not exist.</i>) 409 = Conflict (<i>Sent data already exist.</i>) 410 = Gone (<i>Method does not exist anymore. Must have returned 299 in previous level.</i>) 418 = I'm a teapot 460 = Level Version Out Of Range (<i>Requested level version not yet implemented.</i>) 461 = Method Version Out Of Range (<i>Requested method version not yet implemented.</i>)</p> <p>Server Error 500 = Internal Server Error (<i>Fatal error in BAS. Further debug error code may be added to status.msg.</i>)</p> <p>OBS: These codes, even though based on the HTTP response codes, should not be set in the response header as the HTTP code. The HTTP code should always be 200.</p>	Added new status codes based on HTTP codes.
status.msg	Description of the warning or error. This should only be used by humans for debugging and not by the parser. If status.code 500 is returned, the message may include a code used by the BAS for	

	debugging.	
server	Object containing information about BAS server.	
server.api	The highest version supported by BAS. The BCS should note this and conform to older version if BCS is more updated than the BAS.	
server.time	Server epoch time.	
payload	Object containing data for the method.	
payload.customers	A list of requested customers.	
payload.customers[x].id	Customer id. The ID must be formatted as a GUID/UUID. The generation of the ID is not specified.	
payload.customers[x].name	The name of the customer within BAS.	
payload.customers[x].resources	A list of resources this customer has created.	
payload.customers[x].resources.id	A unique id for the resource. The ID must be formatted as a GUID/UUID. The generation of the ID is not specified.	
payload.customers[x].resources.name	<p>The name of the resource in BAS. A rename of a resource in BAS should not alter the id.</p> <p>Optional: if BCS wants to rename its resource/zone name automatically.</p> <p>Optional: if the BAS has hierarchic architecture and wants to display the full path. It should in that case be separated with a /</p>	

Example response:

```
{
  "status": {
    "code": 200,
    "msg": "ok"
  },
  "server": {
    "api": "1.1.10",
    "time": 1475226019
  },
  "payload": {
    "customers": [{
      "id": "e3941203-37c8-4aaf-a10c-a46100ccb787",
      "name": "Customer name",
      "resources": [{
        "id": "5817c100-d599-4f2e-9c25-07e7a64075a0",
        "name": "Location/resource one"
      }, {
        "id": "fe77c299-980e-49a2-82a5-4f42a4cadf34",
        "name": "Location/resource two"
      }
    ]
  }
}
```

Fetching resource data (string time and dates)

REQUEST

Name	Description	Note
method	See method: “GetCustomerData” for more info.	
client	See method: “GetCustomerData” for more info.	
client.api	See method: “GetCustomerData” for more info.	
client.id	See method: “GetCustomerData” for more info.	
client.time	See method: “GetCustomerData” for more info.	
client.hash	See method: “GetCustomerData” for more info.	
payload	Object containing data for the method.	
payload.dateFormat	<p>Specifies what type of format the date and times must have in this request. Both the request and response must conform to selected format.</p> <p>“string” = Dates and times is to be represented as string in the format “yyyy-mm-dd hh:MM:ss” in GMT timezone.</p> <p>“epoch” = Dates and times is to be represented as Unix epoch times. The number of seconds that have elapsed since January 1, 1970 (midnight UTC/GMT) It must interpreted as UInt64</p>	
payload.start	<p>Specifies from what date and time (GMT time) the bookings should start.</p> <p>Every booking that has it’s start and/or end date on or after this time.</p> <p>If dateFormat is set to “epoch” this value must also conform to epoch standard and vise versa.</p>	
payload.end	<p>Specifies from what date and time (GMT time) the bookings should end.</p> <p>Every booking that has it’s start and/or end date before this time.</p> <p>If dateFormat is set to “epoch” this value must also conform to epoch standard and vise versa.</p>	
payload.resources	<p>A list of requested resources.</p> <p>The resources id is requested with method: “GetCustomerData” in beforehand.</p>	

Example request:

```
{
  "method": " GetResourceData",
  "client": {
    "api": "1.1.10",
    "id": "9818d49a-005d-4a83-93b3-9de04a6a5225",
    "time": 1475226019,
    "hash": "a43074f5bfa356d3be7dda51ef2e4016292743bd"
  },
  "payload": {
    "dateFormat": "string"
    "start": "2015-05-01",
    "end": "2016-05-01",
    "resources": [
      "5817c100-d599-4f2e-9c25-07e7a64075a0",
      "fe77c299-980e-49a2-82a5-4f42a4cadf34"
    ]
  }
}
```

RESPONSE

Name	Description	Note
status	See method: "GetCustomerData" for more info.	
status.code	See method: "GetCustomerData" for more info.	
status.msg	See method: "GetCustomerData" for more info.	
server	Object containing information about BAS server.	
server.api	The highest version supported by BAS. The BCS should note this and conform to older version if BCS is more updated than the BAS.	
server.time	Server epoch time.	
payload	Object containing data for the method.	
payload.list	The list of requested bookings. Booked start time < Requested end time and Booked end time > Requested start time	Formatted.
payload.list.id	Unique id for this specific booking instance. No duplicates! This is an "instance id", not booking id. If the booking is a repeated booking "I.e every Sunday" every Sunday must produce its own unique id. This is to accommodate the future function to let the BCS report back information on past bookings to the BAS. The ID must be formatted as a GUID/UUID. The generation of the ID is not specified.	
payload.list.start	The start (GMT time) of the booking. Expressed as selected dateFormat.	
payload.list.end	The end (GMT time) of the booking. Expressed as selected dateFormat.	
payload.list.created	The date and time (GMT time) current booking was made in the BAS. Expressed as selected dateFormat.	
payload.list.signature	A signature/name on the person who made the current booking.	
payload.list.heat	Indicates what type of heat/temperature the booker wants on this particular booking. In the BAS this should be a dropdown or similar presented to the booker when the booking is made or the the template is defined. All values smaller than zero is predefined temperatures in BCS by the facility manager. -2 = Cleaning temperature. Predefined temperature for cleaning staff. -1 = No heat. The temperature should be the same	

	<p>as if there was no booking at this time. The humidity protection must also be active. This value is meant to be used when the resource is booked but no people will be there.</p> <p>0 = Standard heat. No change. The predefined booked temperature is to be used.</p> <p>> 0 = Selected temperature. I.e value of 21 means that the desired temperature for this booking is 21°C</p>	
payload.list.title	The title of this booking.	
payload.list.resources	<p>A list of resources that is booked in the same booking.</p> <p>The requested resourceID should be in the list. And all resources booked together with the booking. If for example the lights resource is booked together with the booking (exists in this list) the BCS knows that the lights inside also should be lit up. Or if the door lock resources is booked the door should also be unlocked for this booking.</p>	

Example response:

```
{
  "status": {
    "code": 0,
    "msg": "ok"
  },
  "server": {
    "api": "1.1.10",
    "time": 1475226019
  },
  "payload": {
    "list": [{
      "id": "afff431b-2835-45a1-9c5e-a100d746ea0c",
      "start": "2015-05-05 11:30:00",
      "end": "2015-05-05 12:00:00",
      "created": "2015-05-01 11:00:00",
      "signature": "Eva Andersson",
      "heat": 0,
      "title": "Booking with standard heat temp",
      "resources": [
        "fe77c299-980e-49a2-82a5-4f42a4cadf34"
      ]
    }, {
      "id": "99c42508-0e9c-4eef-af19-d7dbb48f9b27",
      "start": "2015-04-30 18:30:00",
      "end": "2015-05-02 19:30:00",
      "created": "2015-04-01 11:00:00",
      "signature": "Eva Andersson",
      "heat": -1,
      "title": "Booking with no heat",
      "resources": [
        "5817c100-d599-4f2e-9c25-07e7a64075a0",
        "0f01ce5b-0c9b-4534-b1f1-f4891511ecb1"
      ]
    }
  ]
}
```

Document version notes

Notes 0.0.3:

1. ROWS

After a discussion between ROWS and Jeff a decision has been made to remove timezones.

The goal is to always express dates and times in GMT (Greenwich Mean Time).

It will be up to the receiver of the data to convert the time to appropriate timezone and apply daylight savings if necessary. This is not democratically voted since the number of introduced companies is too low.

2. ROWS

More information about each parameter is requested from Jeff. This is now applied.

Document version notes

Notes 0.0.4:

1. Eniac Data

Eniac Data AB added to list.

Document version notes

Notes 0.0.5:

1. ROWS

Added LJ System AB and JEFF Electronics in the list.

2. ROWS

Changed ROWS color to Yellow and gave LJ System the orange.

3. ROWS

Approved minor clarifying texts from Eniac.

4. ROWS

Adopted Eniacs suggestion on the extra information for “The list of requested bookings.”

Document version notes

Notes 0.0.6:

1. ROWS

Added Mildmedia in the list.

2. ROWS

Moved Document version notes to the end of the document.

Document version notes

Notes 0.0.8:

1. ROWS

Specified more details about ID variables.

2. ROWS

Changed responseStatus names.

3. ROWS

Changed version value implementation based on discussions.

4. ROWS

Added server.supportedVersion in the response

5. ROWS

Changed status.code implementation and added some standard responses.

Document version notes

Notes 0.0.9:

1. LJ System

Change the term “appId” to “clientId”.

2. LJ System

Added status.code's based on the HTTP standard codes.

3. LJ System

Changed the term “hash” to “token”.

Document version notes

Notes 10:

1. ROWS

Removed dots from document version to conform to the API version specifications.

2. ROWS

Format changes

3. ROWS

Added “payload” object.

4. ROWS

Added clientHash example

5. ROWS

Moved clientID to a separate JSON object.

6. ROWS

Added “server” object in the response.

7. ROWS

Added server.time