Planitec WebServices API — v3.3.1

Introduction

All requests to Planitec Web services are made using the MSTE protocol. MSTE is an open-source JSON based serialization protocol admitting to encode cyclic graphs of objects.

Since MSTE is like a binary in JSON encoding, the description of the API will use a plist-like syntax to describe the objects contained in requests and returned data.

API list

The Planitec Web services API contains the following requests :

Request	Description
`/getCapabilities`	returns global informations about the current web service
`/getUsersList`	returns a list of users like associations, companies, organisms, persons, members or associations containing members
`/getPlacesList`	returns a list of places used for reservations
`/getUsersInfo`	returns complete informations about a given users list
`/getPlacesInfo`	returns complete informations about a given places list
`/getReservationsInfo`	returns standard status informations about a given reservations list
`/getGaps`	returns a list of gaps for a list of places, users, activities... intersecting a range of date
`/getFreeGaps`	returns a list of free gaps in a list of places for a dates range, a list of reservation days and a duration or a given starting and ending time
`/createPerson`	create a new person which could be later used for reservation
`/updatePerson`	modify the basic information of a person
`/createReservation`	create a new reservation tagged as temporary and coming form the net.
`/updateReservation`	change a reservation status from temporary to standard to confirmed or to invalid
`/addReservationDocument`	add a document to an existing reservation
`/getCivilities`	returns a list of known civilities
`/getActivities`	returns the list of standard activities used for reservation
`/getActivityTypes`	returns the list of activity types used in standard Planitec activities
`/getDocumentTypes`	returns a list of known document types
`/getReservationType`	returns a list of known reservation types
`/logEvents`(*)	logging external events in Planitec database

(*) interface in evolution. Keep using V2 for now.

All response to requests are expressed as a MSTE dictionary containing at least :

```
{
  requestName = ”getCapabilities” ;
  requestDate = 2015/08/06-17:22:36 ;
  responseDate = 2015/08/06-17:22:37 ;

  ... other informtions returned (depending on the request)
}
```

Connection

In order to be able to access the following API, you must be connected. The connection is held by a session cookie.

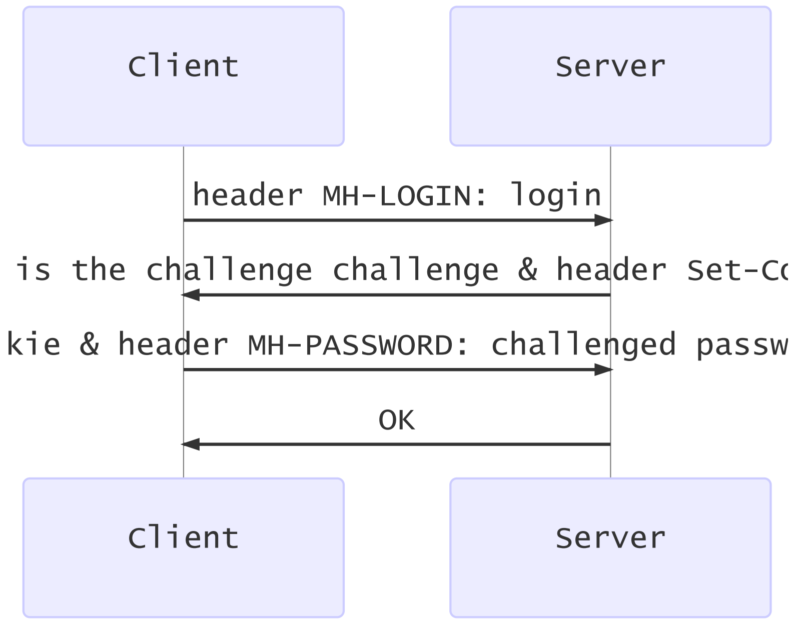
In order to authenticate the connection you first must send the `MH-LOGIN` header with the desired login. You will get a challenge in the body of the response.

With this challenge and the associated password, a challenged password is built.
The challenge is composed of 6 parts:
`algorithm1:hardness1<salt1>algorithm2:hardness2<salt2>`.
There is only one available algorithm at the moment: `1` (SHA512_N).
In order to get the challenged password you must do:

```
s1 = hash(salt1 + password)
do hardness1 times:
    s1 = hash(s1)
t1 = str2hex(s1) # uppercase hex representation of s1

s2 = hash(salt2 + t1)
do hardness2 times:
    s2 = hash(s2)
challenged_password = str2hex(s2) # uppercase hex representation of s2
```

This challenged password is then send by a new request through the `MH-PASSWORD` header. If the server answer is 200 SUCCESS, you are authenticated.


Client
Server
header MH-LOGIN: login
is the challenge challenge & header Set-Co
kie & header MH-PASSWORD: challenged passw
OK
Client
Server

getCapability

Request parameters :

NONE

Request output :

```
informations = {
  database = {
    type = "aString" ;  // database engine ("Oracle", "SQLServer", MySQL"
...)
    modelVersion = "V.S" ; // V = version / S = sub version (ex: "1.0")
  } ;
  service = {
    version = "V.R.C" ; // api version Version.Release.Ccompilation (ex:
"1.0.0")
    name = "sString" ; //Common name of the web service
    availableAPIs = ("getUsersInfo", "getGaps", ...) ; //list of available
requests
  } ;
} ;
```

getUsersList

This requests returns a list of users managed by the connected operator.

Request parameters :

```
usersType = "aString" ; // type of users list we seek (description below)
role = "aString" ; // for now role is optional and if set is
"reservationOperator"
```

The list of `usersType` can be one of the following value :

usersType (string)	Description
`all`	All the managed users
`associations`	Only the managed associations
`persons`	Only the managed persons
`organisms`	Only the managed organisms
`companies`	Only the managed companies

Request output :

```
usersList = (
  {
    identifier = 999 ;  // internal planitec's user unique identifier
    label = "aString" ; // label of the user to be displayed in your list
    type = 9 ; // absent if the asked category is specified in request
               // otherwise 1 = person, 2 = company, 3 = association, 4 =
organism
    externalIdentifier = "aString" ; // user external identifier
    name = "aString" ; // only set for a person
    firstName = "aString" ; // only set for a person
  }, ...
  {
    idenfifier = 999 ;
    label = "xxx" ;
    type = 9 ;
    }
  ) ;
```

If the connected operator has no managed users in the asked category, an empty `usersList` array is returned.

Release notes V3.2 :

- `name` and `firstName` fields have been added to the response for human users
- `externalIdentifier` field has been added in order to permit comparison by the operator

Release notes V3 :

- `externalUserID` field is no longer supported in this request
- `usersType` field values memberIn and `associationsWithMemberIn` are no longer supported in this request
- the returned `usersList` objects now contains a `type` field instead of a `name` field. This new `type` field is set only if the requested `usersType` is et to `all`.

getPlacesList

This requests returns a list of places managed by the connected operator.

Request parameters :

```
userID = 999 ; // an optional planitec userID considered as the places
manager
role = "aString" ; // for now role is optional and if set is
"reservationOperator"
```

if no userID is set, the request returns all the managed places

Request output :

```
placesList = (
  {
    identifier = 999 ;  // internal planitec's place unique identifier
    label = "aString" ; // label of the place to be displayed in your list
  },
   ...
) ;
```

If the connected operator has no managed users in the asked category, an empty `placesList` array is returned.

Release notes V3 :

- the returned `placesList` objects don't have a `name` field anymore.
- the request does not user a `peopleID` field anymore. It has been replaced by the field `userID`. The `peopleID` field is still considered as valid but obsolete.

getUsersInfo

This requests returns complete informations about the asked user identifiers.

Request parameters :

```
userIdentifiers = () ; // an array of asked user identifiers (description below)
                       // (you can also use "users" as backward compatibility)
extensionAttributes = {} // a dictionary of User extensions we want to get
                         // with each user (description bellow)
```

The array of `userIdentifiers` (or `users`) should be set for the request to be valid. This `userIdentifiers` array is composed by a list of entries which can be one of this kind :

Entry type	Format	Description
String	a non empty string	The external identifier of a People object you want to fetch.
Number	a non null unsigned int number.	The Planitec internal identifier of a People object you want to fetch.
Dictionary	One `"Identifier"` key which has an unsigned int number value	The Planitec internal identifier of a People object you want to fetch (for backward compatibility).
Dictionary	One `"Identifier"` key which has a non empty string value	The external identifier of a People object you want to fetch (for backward compatibility).

The `extensionAttributes` dictionary should contains a list of key/values pairs. Each key will be the future key used in the `User` object you will get in the request output. Beware not to overwrite an existing key (see below) by a new key you put in `extensionAttributes` dictionary. Each value of the key/value pair in this dictionary define the kind of extensionyou want to get from Planitec database. The definition itself is a dictionary with 2 mandatory entries and an optional third :

```
myExtension = {
    name = "EXTENSION_NAME" ; // mandatory planitec extension name
    type = "aType" ; // mandatory planitec extension type ("string",
"int",
                     // "unsigned", "float", "bool", "date" or "color")
    defaultValue = aValue ; // can be a String, a Date, a MSTE color
object...
} ;
```

System extensions :

Using the extension mechanism, you can retrieve all users extensions including the Planitec's system extensions. Beware, all extensions listed here are not used in all Planitec configurations.

Planitec Extension	Type	Perimeter	Description
BIRTHDAY	date	Only for persons	Birthday of person
CORRES_1	string	Only for moral people	Contact name (deprecated)
CORRESBIS_1	string	idem	Second contact name
STREET_1	string	idem	Contact street number
ADR1_1	string	idem	Contact address line 1
ADR1_2	string	idem	Contact address line 2
ZIP_1	string	idem	Contact zip code
CITY_1	string	idem	Contact city
TEL_1	string	idem	Contact phone
MOBILE_1	string	idem	Contact mobile phone
FAX_1	string	idem	Contact fax
EMAIL_1	string	idem	Contact email
CORRES_2	string	idem	Invoice contact name
CORRESBIS_2	string	idem	Second invoice contact name
STREET_2	string	idem	Invoice contact street number
ADR_2_1	string	idem	Invoice contact address line 1
ADR_2_2	string	idem	Invoice contact address line 2
ZIP_2	string	idem	Invoice contact zip code
CITY_2	string	idem	Invoice contact city
TEL_2	string	idem	Invoice contact phone
MOBILE_2	string	idem	Invoice contact mobile phone
FAX_2	string	idem	Invoice contact fax
EMAIL_2	string	idem	Invoice contact email
			Organism, company or association

WEB_URL	string	idem	web site URL
KAPITAL	int	Only for companies	Company capital
COMMENTARY	string	all users	Commentaries about the user

Request output :

```
requestedUsers = ( ... ) ; // an array of the requested users infos
allUsers = ( ... ) ; // an array of all fetched users (including parents, 
children)
rootUsers = ( ... ) ; // an array of all root users fetched

allUserTypes = ( ... ) ; // all types fetched for fetched users
allFunctions = ( ... ) ; // all functions fetched for fetched users
allActivities = ( ... ) ; // all activities fetched for fetched users
allBadges = ( ... ) ; // all fetched badges for fetched users
allBadgeTypes = ( ...) ; // all fetched badge types for fetched badges
```

Each `User` object described in `requestedUsers`, `allUsers` and `rootUsers` arrays and `employees`, `members`, `electeds`, `parentUser` and `children` relationships contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal user identifier
`externalIdentifier`	String	64	YES	External user identifier
`label`	String	151	NO	Complete user name (can be computed)
`name`	String	100	NO	User name (2 character mini)
`firstName`	String	50	YES	User first name
`originalName`	String	100	YES	User maiden name
`administrativeCode`	String	50	YES	Person social security number or association, organism or company registration number.
`code`	String	20	YES	User code for internal or external import purpose (was previously `internalCode`)
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`parentsPath`	String	512	YES	Path of the parents composed by the id of each parent people separated by '/' (eg: `"1/2/3"`).
`streetNumber`	String	10	YES	
`address1`	String	100	YES	first line of address
`address2`	String	100	YES	second line of address
`zipCode`	String	10	YES	
`city`	String	50	YES	
`tel1`	String	20	YES	phone number 1
`tel2`	String	20	YES	phone number 2
`fax`	String	20	YES	fax number
`email`	String	128	YES	email
`geox`	Float		YES	x projected localization
`geoy`	Float		YES	y projected localization
`parentUser`	Object		YES	parent User object
`children`	Array		YES	array of children objects (array of User objects)
`type`	Object		YES	type object of the user (a UserType object)
`juridicType`	Object		YES	juridic type object of the user (a UserType object). This attribute is the user's civility.
`badges`	Array		YES	array of Badge objects possessed by the user
`practicedActivities`	Array		YES	array of Activity objects describing for moral person the activities they offer
`employees`	Array		YES	array of User objects which are employees
`members`	Array		YES	array of User objects which are members
`electeds`	Array		YES	array of User objects which are elected officials

Each `UserType` object described in `type` and `juridicType` relationships contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal user type identifier
`label`	String	100	NO	User type name (2 characters min)
`code`	String	20	YES	User type code
`category`	Uint32	1	YES	absent or 0 = standard, 1 = juridic type
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`users`	Array		NO	array of Users objects which have that type

Each Activity Object described in practicedActivities Array contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal activity identifier
`activityTypeID`	UInt32		YES	Planitec internal activity type identifier
`label`	String	100	NO	Activity name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp

Each `Badge` object described in `badges` relationship contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal badge identifier
`number`	UInt32		NO	Badge number.
`serialNumber`	String	100	NO	badge serial number
`label`	String	30	NO	Generaly a less than 30 characters string composed by the technology code (20 characters max) and the badge number (10 decimal digits max)
`ownerName`	String	151	NO	Owner complete name. Obsolete as *redundant information with user.label. Should disappear in future versions*.
`isMasterBadge`	Boolean		YES	absence is equivalent to the NO value
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`user`	Object		NO	the User object this badge belongs to
`type`	Object		YES	The badge's type as a BadgeType object.

Each `BadgeType` object described in the previous `type` relationship in `Badge` object contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal badge type identifier
`label`	String	100	NO	Badge type (technology) name (2 characters mini)
`code`	String	20	YES	Badge technology code name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`badges`	Array		NO	array of Badges objects which have that type

Release notes V3 :

- `users` request field has been replaced by `userIdentifiers`. The previous `users` field can still be used has backward compatibility but is considered as obsolete.
- `User` objects fields évolution :

    - added `originalName`, `externalIdentifier`, `fax`, `juridicType` and `geox` and `geoy` fields
    - field `code` replaces old `internalCode` field
- `Badge` objects `ownerName` field is now tagged as obsolete
- Request now accepts to fetch Planitec extensions on `User` objects.
- `UserType` objects have now a `category` field, 0 = standard, 1 = juridic type

getPlacesInfo

This requests returns complete informations about the asked places identifiers.

Request parameters :

```
placeIdentifiers = () ; // an array of asked user identifiers (description
below)
                       // (you can also use "places" as backward
compatibility)
extensionAttributes = {} // a dictionary of User extensions we want to get
                         // with each place (description bellow)
```

The `placeIdenfiers` array is a list a Planitec’s `Place` identifiers. All these identifiers are positive numbers.

The `extensionAttributes` dictionary follows the same rules as described in getUsersInfo request.

System extensions :

Using the extension mechanism, you can retrieve places extensions including the Planitec's system extensions.

Planitec Extension	type	Description
COUNTY	string	County (Canton)
SECTOR	string	Sector (Arrondissement)
CITY_CODE	string	CityCode (Commune)
COMMENTARY	string	Commentaries about the place
TOTCAP	int	Place total capacity
SEATCAP	int	Seating place capacity
STANDCAP	int	Standing place capacity
SURFACE	int	Place surface
DIMENSIONS	string	Place dimension description

Request output :

```
requestedPlaces = (...) ; // an array of the requested users infos
allPlaces = (...) ; // an array of all fetched places (including parents,
children)
rootPlaces = (...) ; // an array of all root places fetched
allPlaceTypes = ( ... ) ; // all types fetched for fetched places
allActivities = ( ... ) ; // all activities fetched for fetched places
allTriggers = ( ... ) ; // all fetched triggers for fetched places
```

Each `Place` object described in `requestedPlaces`, `allPlaces` and `rootPlaces` arrays and `parentPlace`, `substitutions` and `children` relationships contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal place identifier
`resourceIdentifier`	UInt32		NO	Planitec internal place's twin resource identifier
`label`	String	100	NO	User name (2 character mini)
`code`	String	20	YES	Place code for internal or external import purpose (was previously `internalCode`)
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`parentsPath`	String	512	YES	Path of the parents composed by the id of each parent place separated by '/' (eg: `"1/2/3"`).
`streetNumber`	String	10	YES	
`address1`	String	100	YES	first line of address
`address2`	String	100	YES	second line of address
`zipCode`	String	10	YES	
`city`	String	50	YES	
`tel1`	String	20	YES	phone number 1
`tel2`	String	20	YES	phone number 2
`fax`	String	20	YES	fax number
`email`	String	128	YES	email
`geox`	Float		YES	x projected localization
`geoy`	Float		YES	y projected localization
`provitionTime`	Int32		YES	default time necessary for setting up the place
`returnTime`	Int32		YES	default time necessary to clean the place
`parentPlace`	Object		YES	parent Place object
`children`	Array		YES	array of children objects (array of Place objects)
`type`	Object		YES	type object of the user (a PlaceType object)
`triggers`	Array		YES	array of Trigger objects attached to the Place object
`activities`	Array		YES	array of Activity objects which could be practices by users this place
`substitutions`	Array		YES	array of Places objects which can be used as substitutions places for this place if it is occupied

Each `PlaceType` object described in `type` relationship contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal place type identifier
`label`	String	100	NO	Place type name (2 characters min)
`code`	String	20	YES	Place type code
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`places`	Array		NO	array of Place objects which have that type

Each `Activity` Object described in `activities` relationship contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal activity identifier
`activityTypeID`	UInt32		YES	Planitec internal activity type identifier
`label`	String	100	NO	Activity name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`places`	Array		NO	List of places where this activity is done

Each `Trigger` object described in `triggers` relationship contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal trigger identifier
`label`	String	100	NO	Trigger name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`places`	Array		NO	All places where this trigger appears

`Release notes` V3 :

- `places` request field has been replaced by `placeIdentifiers`. The previous `places` field can still be used has backward compatibility but is considered as obsolete.
- `Place` objects fields évolution :
  - added `resourceIdentifier`, `fax`, `provisionTime`, `returnTime`, `geox` and `geoy` fields
  - field `code` replaces old `internalCode` field
- Request now accepts to fetch Planitec extensions on `Place` objects.

getReservationsInfo [new request in V3]

This requests returns complete informations about the asked reservations identifiers. Warning, this request does not fetch any gaps, it's only information status about reservations

Request parameters :

```
reservationsIdentifiers = () ; // an array of asked reservation
identifiers
                               // (description below)
extensionAttributes = {} // a dictionary of User extensions we want to get
                         // with each place (description bellow)
```

The `reservationsIdentifiers` array is a list a Planitec's `Reservations` UInt32 identifiers.

The `extensionAttributes` dictionary follows the same rules as described in getUsersInfo request.

Common extensions :

Using the extension mechanism, you can retrieve all reservations extensions including these common ones :

Common Extension	Type	Description
COMMENTARY	string	Commentary of the reservation (system extension)
MINUT_ENTREE	string	Only for BODET access systems. Returns a string containing an unsigned value in minutes
MINUT_SORTIE	string	Only for BODET access systems. Returns a string containing an unsigned value in minutes
TYPE_CRENEAU	string	Only for BODET access systems. Returns a string containing an unsigned code : 2 = `"Badgeage"`, 11 = `"Libre"`, 12 = "Libre après premier badgeage ».
DEL_PRE	string	Delay in minutes for heating. Returns a string containing an unsigned value
DEL_FERM	string	Delay in minutes before closing. Returns a string containing an unsigned value
DEL_OUV	string	Delay in minutes before opening. Returns a string containing an unsigned value

Request output :

```
requestedReservations = (...) ; // an array of the requested users infos
allReservationTypes = ( ... ) ; // all types fetched for fetched
reservations
allActivities = ( ... ) ; // all activities fetched for fetched
reservations
```

Each `Reservation` object and objects described in `requestedReservations` array contains :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal reservation identifier
`label`	String	100	NO	Reservation object
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`isSingle`	Boolean		NO	Does the reservation contains one gap or multiple gaps
`isLocked`	Boolean		NO	Is the reservation user locked
`code`	String	20	YES	Internal reservation code
`start`	DateTime		NO	Reservation starting date & time
`end`	DateTime		NO	Reservation ending date & time
`previsionalRegistereds`	UInt32		NO	Previsional people coming (can be 0)
`registereds`	UInt32		NO	People who did come (can be 0)
`situation`	Char		NO	0 = invalid, 1 = pre-reservation, 2 = standard reservation, 3 = confirmed
`activity`	Object		NO	The reservation activity
`type`	Object		YES	The reservation type
`contractor`	Object		NO	The reservation contractor
`price`	Int32		NO	The reservation price (without VAT) in 1/1000 of the current money unit
`vat`	Int32		NO	The reservation VAT) in 1/1000 of the current money unit

Each `ReservationType` object described in `type` relationship contains following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal reservation type identifier
`label`	String	100	NO	Reservation type name (2 characters min)
`code`	String	20	YES	Reservation type code
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`reservations`	Array		NO	array of Reservation objects with that type

Each Activity Object described in `activity` relationship contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal activity identifier
`activityTypeID`	UInt32		YES	Planitec internal activity type identifier
`label`	String	100	NO	Activity name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`reservations`	Array		NO	List of reservations with this activity

Each People Object described in `contractor` relationship contains the following attributes :

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal user identifier
`externalIdentifier`	String	64	YES	External user identifier
`label`	String	151	NO	Complete user name (can be computed)
`name`	String	100	NO	User name (2 character mini)
`firstName`	String	50	YES	User first name
`originalName`	String	100	YES	User maiden name
`administrativeCode`	String	50	YES	Person social security number or organization registration number.
`code`	String	20	YES	User code for internal or external import purpose (was previously `internalCode`)
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`parentsPath`	String	512	YES	Path of the parents composed by the id of each parent people separated by '/'.
`streetNumber`	String	10	YES	
`address1`	String	100	YES	first line of address
`address2`	String	100	YES	second line of address
`zipCode`	String	10	YES	
`city`	String	50	YES	
`tel1`	String	20	YES	phone number 1
`tel2`	String	20	YES	phone number 2
`fax`	String	20	YES	fax number
`email`	String	128	YES	email
`geox`	Float		YES	x projected localization
`geoy`	Float		YES	y projected localization

Release notes V3.1 :

- the returned attribute `situation` did replace the `reservationStatus` field described in V3 and is now a full unsigned value with 0 = invalid, 1 = pre-reservation, 2 = standard reservation and 3 = confirmed reservation.

createPerson [new request in V3]

This requests allow the current operator to create new person who can be reservation contractors.

Request parameters :

The requested parameters for this request are described in the following table :

Parameter	Type	Mandatory	Size max	Description
`externaUserIdentifier`	String	YES	64	The operator's external unique identifier. This identifier cannot change after the person creation.
`name`	String	YES	100	Person name. Should contains at least 2 characters
`firstName`	String	NO	50	Person first name
`code`	String	NO	20	Person administrative code
`email`	String	NO	128	Person email
`civilityCode` OR `civilityID`	String OR UInt32	NO	20 (string)	Person civility code (juridic type internal code) or person civility ID.
`streetNumber`	String	NO	10	address street number
`address1`	String	NO	100	address line 1
`address2`	String	NO	100	address line 2
`zipCode`	String	NO	10	address zip code
`city`	String	NO	50	address city
`tel1`	String	NO	20	phone 1
`tel2`	String	NO	20	phone 2
`fax`	String	NO	20	fax

Request output :

```
externalUserIdentifier = "aString" ; // the given external identifier
userIdentifier = aNumber ; // Planitech internal UInt32 people identifier
creationStatus = "aString" ; // "OK" or "DUPLICATE" if the user did
already
                             // exists and cannot be created twice OR
"BADCIVILITY"
                             // if the civility code is wrong
```

A `BADCIVILITY` value in `creationStatus` means that the person is created but it’s civility is not set.

Release notes V3.1 :

- the request field `civilityID` has beed added to the request. It could be used instead of `civilityCode`. Both keys cannot be used at the same time

updatePerson [new request in V3]

This requests allow the current operator to update a person informations.

Request parameters :

The request parameters for this request are nearly the same as the createPerson request. You just can tell which person you want to modify by using its Planitec internal Uint32 `userIdentifier` instead of the `externalUserIdentifier`. Both request attributes cannot be used at the same time.

With this request you never can remove the civility of a person. You can change it though by using a new `civilityCode`.

Request output :

```
modificationStatus = "aString" ; // "OK" or "BADCIVILITY" or "NOTFOUND" or
NOP
```

`BADCIVILITY` means that the given civility code is wrong. All fields but this one are modified.

`NOTFOUND` means we have not found the person meant to be changed.

`NOP` means nothing was changed (you did’t change the content of the given person by your call)

Release notes V3.1 :

- the dictionary `modifications` does not exist any more in the request. Instead, all modified fields are directly set in the request.

createReservation [new request in V3]

This requests allow the current operator to create new reservations. The reservation is created with a INET status and as pre-reservations (situation code 1)

Request parameters :

The requested parameters for this request are described in the following table :

Parameter	Type	Mandatory	Size max	Description
`object`	String	NO	100	Short description of the reservation. If not set, the contractor's name is used.
`contractorID` or `contractorExternalIdentifier`	UInt32 or string	YES	64 (string)	planitec ID or external identifier of the reservation contractor
`requesterID` or requesterExternalIdentifier	UInt32 or string	NO	64 (string)	planitec ID or external identifier of the reservation requester
`activityID` or `activityCode`	UInt32 or string	YES	64 (string)	Planitec ID or internalCode of the reservation activity
`typeID` or `typeCode`	UInt32 or string	NO	64 (string)	Planitec ID or internalCode of the reservation type
`code`	String	NO	20	Reservation internal code
`price`	UInt32	NO. Absent if weekly is set to YES.	60% of MaxInt	Reservation price in 1/1000 of unit. If set, the reservation price is set with no calculation by Planitec.
`vatRate`	UInt32	NO. Absent if weekly is set to YES.	4000	VAT rate in 1/10000.
`isWeekly`	Boolean	YES		The reservation has a single gap or multiple gaps based on a weekly period type.
`start`	Date	YES		Reservation starting date
`end`	Date	YES		Reservation ending date
`places`	Array	YES		Array of Planitec's place identifiers
`gapStart`	UInt32	YES if weekly. Absent otherwise	1435	starting gaps time (minutes)
`gapEnd`	UInt32	YES if weekly. Absent otherwise	1440	ending gaps time (minutes)
`days`	Natural Array	YES if weekly. Absent otherwise	7 numbers	List of days we enforce the reservations (0 to 6).
`noConflicts`	Boolean	NO (considered as NO if absent)		If set to YES, the reservation cannot be created if there's any conflict with any other reservations

Request output :

```
reservationIdentifier = aNumber ; // Planitech internal UInt32 reservation
creationStatus = "aString" ; "OK" or "INVALID" or "BADTYPE" or "KO" or
"CONFLICTS"
```

The returned `creationStatus` are :

creationStatus	Description
`OK`	The reservation was correctly created and all the informations you put in were accepted.
`BADTYPE`	The reservation was correctly created and the information but the reservation type were accepted.
`INVALID`	This code is returned only if you put `noConflicts` parameter to `YES` and the reservation you tried to create appeared to be in conflict. In that case the created `reservationIdentifier` is returned but the created reservation is saved in Planitec Database with an `invalid` status.
`CONFLICTS`	This code is returned only if you put `noConflicts` parameter to `YES` and the reservation you tried to create appeared to be in conflict and for some internal reason the web service could not register the new created reservation as invalid. In that case the created `reservationIdentifier` is returned and it’s up to the web service to later change that reservation status to invalid.
`KO`	The reservation could not be created at all.

Release notes V3.3 :

- A request field `noConflicts` has been added.

Release notes V3 :

- the request field `contractorExternalIdentifier` has beed added to the request. It could be used instead of `contractorID`. Both keys cannot be used at the same time.
- the request field `requesterExternalIdentifier` has beed added to the request. It could be used instead of `requesterID`. Both keys cannot be used at the same time.

- the request field `activityCode` has beed added to the request. It could be used instead of `activityID`. Both keys cannot be used at the same time.
- the request fields `typeID` and `typeCode` have beed added to the request. Their role is to set the future reservation type. Both keys cannot be used at the same time.
- the request field `isWeeklyReservation` has been replaced by the field `isWeekly`.
- the request field `startingDate` has been replaced by the field `start`.
- the request field `endingDate` has been replaced by the field `end`.
- the request field `startingGapsHour` has been replaced by the field `gapStart`.
- the request field `endingGapsHour` has been replaced by the field `gapEnd`.
- the request field `reservationDays` has been replaced by the field `days`.

updateReservation [new request in V3.1]

This requests allow the current operator to update a reservation status. It replaces the `updateReservationStatus` request which appeared in V3.

Request parameters :

```
reservationIdentifier = aNumber ; // Planitech internal UInt32 reservation
ID
situation = aNumber ; // 0 to 3 (description bellow)
typeID = aNumber or typeCode = "aString" ; // you can change the
reservation type
object = aString ; // you can change the reservation object
code = aSring ; // you can change the reservation internal code
```

Each status cannot be set at all times. Here is the status evolution table

Original situation	Authorized situation change
`1` = pre-reservation	0 = invalid, `2` = standard, `3` = confirmed,
`2` = standard	0 = invalid, `3` = confirmed
`3` = confirmed	`0` = invalid
`0` = invalid	NONE

Following this rules, this web service cannot revalidate any invalidated reservation.

Request output :

```
modificationStatus = "aString" ; // "OK", "BADTYPE", "NOTFOUND" or "NOP"
```

A `BADTYPE` value in `modificationStatus` means that the new reservation type was not found and so the actual reservation type remains valid.

A `NOTFOUND` value in `modificationStatus` means the system didn’t found the searched reservation.

A `NOP` value in `modificationStatus` means that the modifications you attempted to make didn’t change the actual reservation. Beware, if you request only for a type change and if the new type is not found, the final error would be `NOP` and not `BADTYPE` because nothing did change.

addReservationDocument [new request in V3]

This requests allow the current operator to attach a new document to an existing reservation.

Request parameters :

```
reservationIdentifier = aNumber ; // Planitech internal UInt32 reservation
ID

.../... // document description fields (see below)

documentContent = aDataStram ; // an MSTE Data with the document content
```

The `document` descriptions fields are :

Attribute	Type	Mandatory	Size max	Description
label	String	NO	100	Document label (if not set uses part of fileName)
fileName	String	YES	100	Document file name (not a path, only the file name with its extension, e.g. `"MyPassport.pdf"`). Having an extension is mandatory.
code	String	NO	20	Document internal code
typeID or typeCode	UInt32 or String	NO	20 (String)	Document type Planitec identifier (number) or document type code (should match a real document type internal code).
validityStart	Date	NO		Date object for starting validity
validityEnd	Date	NO		Date object for ending validity
issueDate	Date	NO		Date object. The date of issue of the document (e.g. the issue date of you driver licence)

Request output :

```
reservationDocumentIdentifier = aNumber ; // planitec ID of the created
document
creationStatus = "OK" or "NOTFOUND" ; // NOTFOUND if the reservation is
not found
```

Release notes V3.1 :

- added request field `issueDate`.
- the request field `name` has been replaced by `label`.
- the `document` dictionary has been replaced by its fields in the request.
- the request field `typeID` has beed added to the request. It has the same role as `typeCode` field. Their role is to set the future document reservation type. Both keys cannot be used at the same time.
- Less creationStatus as request output. Everything should be OK for the document to be created and when it’s created, the `reservationDocumentIdentifier` is returned.
- The maximal transmissionn size is **10MB** while Base64 encoded with request extra informations which means your document should not be heavier than **7MB**.

getGaps

This requests allow the current operator to retrieve planning gaps with all associated informations.

-

Request parameters :

The requested parameters for this request are described in the following table :

Parameter	Type	Mandatory	Description
`start`	DateTime	YES	Precise date from when we want to fetch gaps
`end`	DateTime	YES	Precise date to when we want to fetch gaps
`placeIdentifiers`	Array of UInt32	NO	List of Planitec's place IDs from where we want to fetch gaps
`userIdentfiers`	Array of UInt32 or String Identifiers	NO	List of Planitec’s user IDs or externalIdentifiers (you can mix) to define the contractors from whom we want to fetch gaps
`activityIdentifiers`	Array or UInt32	NO	List of Planitec activity IDs to define the activities from which we want to fetch gaps

Request output :

```
gaps = (...) ; // an array of gaps matching the request
closureGaps = ( ... ) ; // an array of closure gaps matching the request
allPlaces = ( ... ) ; // all places fetched for fetched gaps
allUsers = ( ... ) ; // all contractor users fetched for fetched gaps
allTriggers = ( ... ) ; // all triggers fetched for fetched gaps
allBadges = ( ... ) ; // all badges fetched for fetched gaps
allBadgeTypes = ( ... ) ; // all badges types fetched for fetched gaps
allActivities = ( ... ) ; // all activities fetched for fetched gaps
requestedPlaces = ( ... ) ; // all requested places
requestedUsers = ( ... ) ; // all requested users
requestedActivities = ( ... ) ; // all requested activities
rootPlaces = ( ... ) ; // extract root places from all fetched places
rootUsers = ( ... ) ; // root users from all fetched users
```

All gaps returned by the request have the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec's gap unique identifier
`label`	String	100	NO	Reservation label
`reservationID`	UInt32		NO	Planitec's reservation unique identifier
`modificationDate`	DateTime		NO	Last modification date
`startingDate`	DateTime		NO	gap start
`endingDate`	DateTime		NO	gap end
`isClosure`	Boolean		YES	if set and the value is YES, the gap is a closure reservation gap
`openingMode`	UInt32	12	YES	value can be 2 = « Badgeage » , 11 = « Free », 12 = « Libre après premier badgeage ». DEPRECATED. SHOULD SOON BE REPLACED BY FETCHING EXTENSIONS
`openingTimer`	UInt32		YES	DEPRECATED. SHOULD SOON BE REPLACED BY FETCHING EXTENSIONS
`closingTimer`	UInt32		YES	DEPRECATED. SHOULD SOON BE REPLACED BY FETCHING EXTENSIONS
`activity`	Activity object		NO	reservation Activity Object
`contractor`	User object		NO	reservation contractor User Object
`users`	Array of User objects		YES	array of users who have badges for this gap
`places`	Array of Place objects		NO	array of places reserved with this gap
`badges`	Array of Badge objects		YES	array of badges available for this gap
`triggers`	Array of Trigger objects		YES	array of triggers available for this gap

Each `User` Object described in returned request relatationships contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal user identifier
`externalIdentifier`	String	64	YES	External user identifier
`label`	String	151	NO	Complete user name (can be computed)
`name`	String	100	NO	User name (2 character mini)
`firstName`	String	50	YES	User first name
`originalName`	String	100	YES	User maiden name
`administrativeCode`	String	50	YES	Person social security number or association, organism or company registration number.
`code`	String	20	YES	User code for internal or external import purpose (was previously `internalCode`)
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`parentsPath`	String	512	YES	Path of the parents composed by the id of each parent people separated by '/' (eg: `"1/2/3"`).
`streetNumber`	String	10	YES	
`address1`	String	100	YES	first line of address
`address2`	String	100	YES	second line of address
`zipCode`	String	10	YES	
`city`	String	50	YES	
`tel1`	String	20	YES	phone number 1
`tel2`	String	20	YES	phone number 2
`fax`	String	20	YES	fax number
`email`	String	128	YES	email
`geox`	Float		YES	x projected localization
`geoy`	Float		YES	y projected localization
`parentUser`	Object		YES	parent User object
`children`	Array		YES	array of children objects (array of User objects)
`type`	Object		YES	type object of the user (a UserType object)
`juridicType`	Object		YES	juridic type object of the user (a UserType object). The juridic type of a person is its civility.
`badges`	Array		YES	array of Badge objects possessed by the user
`gaps`	Array		NO	Gaps where the user appears as contractor or employee or possessor of a badge

Each `Place` Object described in returned request relatationships contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal place identifier
`resourceIdentifier`	UInt32		NO	Planitec internal place's twin resource identifier
`label`	String	100	NO	User name (2 character mini)
`code`	String	20	YES	Place code for internal or external import purpose (was previously `internalCode`)
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`parentsPath`	String	512	YES	Path of the parents composed by the id of each parent place separated by '/' (eg: `"1/2/3"`).
`streetNumber`	String	10	YES	
`address1`	String	100	YES	first line of address
`address2`	String	100	YES	second line of address
`zipCode`	String	10	YES	
`city`	String	50	YES	
`tel1`	String	20	YES	phone number 1
`tel2`	String	20	YES	phone number 2
`fax`	String	20	YES	fax number
`email`	String	128	YES	email
`geox`	Float		YES	x projected localization
`geoy`	Float		YES	y projected localization
`provitionTime`	Int32		YES	default time necessary for setting up the place
`returnTime`	Int32		YES	default time necessary to clean the place
`parentPlace`	Object		YES	parent Place object
`children`	Array		YES	array of children objects (array of Place objects)
`triggers`	Array		YES	array of Trigger objects attached to the Place object
`gaps`	Array		YES	array of Gaps objects which contain the place as reserved resource
`closureGaps`	Array		YES	array of Gaps objects which contain the place as closure

One of the array `gaps` or `closureGaps` `should be set.

Each `Activity` Object described in returned request relatationships contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal activity identifier
`activityTypeID`	UInt32		YES	Planitec internal activity type identifier
`label`	String	100	NO	Activity name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`gaps`	Array		NO	Gaps where the activity is praticed

Each Badge `Object` described in returned request relatationships contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal badge identifier
`number`	UInt32		NO	Badge number.
`serialNumber`	String	100	NO	badge serial number
`label`	String	30	NO	Generaly a less than 30 characters string composed by the technology code (20 characters max) and the badge number (10 decimal digits max)
`ownerName`	String	151	NO	Owner complete name. Obsolete as *redundant information with user.label. Should disappear in future versions*.
`isMasterBadge`	Boolean		YES	absence is equivalent to the NO value
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`user`	Object		NO	the User object this badge belongs to
`type`	Object		YES	The badge’s type as a BadgeType object.
`gaps`	Array		NO	Gaps where the current badge is used.

Each `BadgeType` Object described in returned request relatationships contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal badge type identifier
`label`	String	100	NO	Badge type (technology) name (2 characters mini)
`code`	String	20	YES	Badge technology code name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`badges`	Array		NO	array of Badges objects which have that type

Each `Trigger` Object described in returned request relatationships contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`identifier`	UInt32		NO	Planitec internal trigger identifier
`label`	String	100	NO	Trigger name
`modificationDate`	DateTime		NO	Last object modification DateTime stamp
`places`	Array		NO	All places where this trigger appears
`gaps`	Array		NO	All gaps where this trigger appears

Release notes V3 :

- the request field `startingDate` has been replaced by `start`.
- the request field `endingDate` has been replaced by `end`.
- the request field `places` has been replaced by `placeIdentifiers`.
- the request field `users` has been replaced by `userIdentfiers`.
- the request field `activities` has been replaced by `activityIdentifiers`.
- The array `closureGaps` in Place objects is now working. Before, `gaps` relationship was wrongly used
- A relationship `contractor` did appear in Gap objects. Before there was only the `users` relationship.

getFreeGaps

This requests allow the current operator to verify the availability of a given list of places.

-

Request parameters :

This request must have 3 mandatory parameters :

```
placeIdentifiers = ( ... ) ; // an array of targeted place identifiers
startingDate = aDate ; // starting date for searching
endingDate = aDate ; // ending date for searching
```

Then, we can specify a searched duration within a range of time :

```
requestedDuration = aNumber ; // searched duration in minutes
startingTime = aNumber ; // starting time in minutes for searching
endingTime = aNumber ; // ending time in minutes for searching
```

Or you can directly specify a searched gap in the given dates :

```
requestedStartingTime = aNumber ; // gap starting time in minutes for
searching
requestedEndingTime = aNumber ; // gap ending time in minutes for
searching
```

In both case, you can optionally specify the searched days :

```
reservationDays = (...) ; // a array of searched days (0 = Sun, ... 6 =
Sat)
```

Request output :

```
availablePlaces = (...) ; // an array of places with free gaps
(description below)
```

Each `Place` in `availablePlaces` contains the following attributes:

Attribute	Type	Size max	allowsNull	Description
`placeIdentifier`	UInt32		NO	The place internal identifier
`resourceIdentifier`	UInt32		NO	The place twin resource internal identifier
`label`	String	100	NO	The place name
`freeGaps`	Array		NO	All gaps available for the given place in the requested range

Each free `Gap` in `freeGaps` is a `MSTE Couple` object with the starting availability as a `Date` object in its `firstMember` and the ending availability as a `Date` object in its `secondMember`.

Release notes V3 :

- the attribute `placeID` in returned Place objects as been replaced by `placeIdentifier`.
- the attribute `resourceID` in returned Place objects as been replaced by `resourceIdentifier`.
- the attribute `resourceName` in returned Place objects as been replaced by `label`.

getCivilities [new request in V3.2]

This requests allow the current operator to all civilities registered in the database.

Request parameters :

NONE

Request output :

```
civilities = ( ... ) ; // an array of Civility objects
```

Each `Civility` object contains the following attributes

`Attribute`	`Type`	`Size Max`	`allowsNull`	`Description`
label	String	100	NO	Civility name
identifier	UInt32		NO	Planitec internal identifier
code	String	20	YES	Internal code
modificationDate	DateTime		NO	Last modification date

getActivities [new request in V3.2, modified in 3.3.1]

This requests allow the current operator to retrieve all standard activities registered in the database.

Request parameters :

NONE

Request output :

```
activities = ( ... ) ; // an array of Activity objects
```

Each `Activity` object contains the following attributes

`Attribute`	`Type`	`Size Max`	`allowsNull`	`Description`
label	String	100	NO	Activity name
identifier	UInt32		NO	Planitec internal identifier
activityTypeID	UInt32		YES	Planitec internal activity type identifier
code	String	20	YES	Internal code
modificationDate	DateTime		NO	Last modification date

getActivityTypes [new request in V3.3.1]

This requests allow the current operator to all standard activity types registered in the database.

Request parameters :

NONE

Request output :

```
types = ( ... ) ; // an array of ActivityType objects
```

Each `ActivityType` object contains the following attributes

`Attribute`	`Type`	`Size Max`	`allowsNull`	`Description`
label	String	100	NO	Activity type name
identifier	UInt32		NO	Planitec internal identifier
code	String	20	YES	Internal code
modificationDate	DateTime		NO	Last modification date

getDocumentTypes [new request in V3.2]

This requests allow the current operator to all document types registered in the database.

Request parameters :

NONE

Request output :

```
types = ( ... ) ; // an array of DocumentType objects
```

Each `DocumentType` object contains the following attributes

`Attribute`	`Type`	`Size Max`	`allowsNull`	`Description`
label	String	100	NO	Document type name
identifier	UInt32		NO	Planitec internal identifier
code	String	20	YES	Internal code
modificationDate	DateTime		NO	Last modification date

getReservationTypes [new request in V3.2]

This requests allow the current operator to all reservation types registered in the database.

Request parameters :

NONE

Request output :

```
types = ( ... ) ; // an array of ReservationType objects
```

Each `ReservationType` object contains the following attributes

`Attribute`	`Type`	`Size Max`	`allowsNull`	`Description`
label	String	100	NO	Reservation type name
identifier	UInt32		NO	Planitec internal identifier
code	String	20	YES	Internal code
modificationDate	DateTime		NO	Last modification date

logEvents

This requests allow the current operator to log events coming from its own system to Planitec database.

-

Request parameters :

This request only contains an array of events :

```
events = ( ... ) ; // an array of events to be inserted in log events
database
```

The event array is filled with objects containing the following attributes :

Attribute	Type	Mandatory	Size max	Description
`externalIdentifier`	String	NO	40	Event external identifier
`eventDate`	DateTime	YES		Event date
`eventDescription`	String	YES	1000	Event description
`badgeIdentifier` or `badgeLabel`	UInt32 or String	NO	30 (string)	Badge used during the event. The badge is found either with its Planitec's unique identifier or with its Label
`level`	UInt32	NO	0, 1 or 2	0 = standard, 1 = alarm, 2 = error
`type`	String	YES	255	Name of the event
`placeIdentifier`	UInt32	NO		Planitec Place internal identifier. Place where the event occured
`triggerIdentifier`	UInt32	NO		Planitec Trigger internal identifier. Trigger which was fired during the event
`userIdentifier` or `userExternalIdentifier`	UInt32 or String	NO	64 (string)	User connected to this event. The user is described with its Planitec internal identifier (an unsigned) or with its external identifier (a string)

Request output :

This request has no output.

Release notes V3.1 :

- the request field `externalIdentifier` did replace the field `eventID`.
- the request field `eventDescription` did replace the field `desc`.
- the request field `badgeIdentifier` did replace the field `badgeID`.
- the request field `placeIdentifier` did replace the field `placeID`.
- the request field `userIdentifier` did replace the field `userID`.
- the request field `triggerIdentifier` did replace the field `triggerID`.

- the request field `userExternalIdentifier` could be used instead of `userIdentifier`. Both fields cannot be used at the same time.

$$WARNING : THISREQUESTISINEVOLUTIONPROCESS. KEEPUSINGV2.$$

Annexe Spec MSTE

Objectifs:

- facilité de transport:
  - chaine de caractères avec possibilité d'escaper tout caractère Unicode tout en permettant l’utilisation native de l’UTF8 ;
  - sous-ensemble de la norme JSON: tableau de valeurs (chaines et nombres)
- support des classes personalisés
- support des cycles

Version 1.*

Les versions 1.0.* ont pour différences avec la version 2:

- non streamable à l'encodage, il est nécéssaire d'avoir toute la chaîne pour finir l'encodage (CRC, classes et clés)
- utilisation d'un CRC (qui dans la pratique est bien souvent ignoré)
- encodage des références uniquements

La version 1.0.2 réorganise les codes des tokens et simplifie les références.

Format de la chaîne MSTE:

```
+--------------+------------------+-----+---------+------+--------------+
| Version MSTE | Nombre de tokens | CRC | Classes | Clés | Objet racine |
+--------------+------------------+-----+---------+------+--------------+
```

- Version MSTE: MSTE0101 / MSTE0102
- Nombre de tokens: le nombre de tokens total (cela comprends Version MSTE et Nombre de tokens)
- CRC: Le CRC de la chaîne final calculé en définissant le token CRC à CRC00000000
- Classes: le nombre de classes suivi des classes
- Clés: le nombre clés suivi des clés

Tokens MSTE0102

Type	Code	Suite attendue	Ref
Objet null	0	Aucune	Non
Valeur "vraie"	1	Aucune	Non

Valeur "fausse"	2	Aucune	Non
Chaîne vide	3	Aucune	Non
Data vide	4	Aucune	Non
inutilisées	5 - 8		
Référence	9	1 token contenant l'index dans le tableau des objets déjà décodés de l'objet à utiliser comme référence.	Non
int8	10	un nombre compris entre -128 et +127	Non
uint8	11	un nombre compris entre 0 et 255	Non
int16	12	un nombre compris entre -32768 et +32767	Non
uint16	13	un nombre compris entre 0 et 65535	Non
int32	14	un nombre compris entre -(2^31) et (2^31)-1	Non
uint32	15	un nombre compris entre 0 et (2^32)-1	Non
int64	16	un nombre compris entre -(2^63) et (2^63)-1	Non
uint64	17	un nombre compris entre 0 et (2^64)-1	Non
float	18	un nombre flottant interprétable en simple précision	Non
double	19	un nombre flottant interprétable en double précision	Non
décimal	20	un nombre décimal sans limite de stockage	Oui
Chaîne de caractères	21	la chaîne de caractères	Oui
Date (locale)	22	un nombre entier (positif ou négatif) de secondes écoulées ou à venir depuis ou vers le **01/01/1970** qui est la référence (**Unix Epoch**). Cette date ne porte pas d'information de timezone (même pas l'UTC). Elle correspond donc à un temps local.	Oui
Time stamp	23	un nombre entier (positif ou négatif) de secondes écoulées ou à venir depuis ou vers le **01/01/1970** qui est la référence (**Unix Epoch**). Contrairement au code 22, ce time stamp est exprimé en UTC.	Oui
Couleur	24	un nombre entier positif contenant la représentation en 24 ou 32 bits de la couleur. 24 bits = RRGGBB. 32 bits = TTRRGGBB ou TT est la transparence (0x00 = opaque, 0xFF = totalement transparent)	Oui

Data	25	un nombre entier positif correspondant à la longueur originale en octets de la donnée transmise, suivi d'une chaîne de caractères contenant les données encodées en Base64.	Oui
Natural array	*26	un nombre entier positif N correspondant au nombre d'entiers naturels positif contenus dans le tableau suivi de N entiers naturels (uniquement en version 1.2.1)	Oui
inutilisées	27 - 29		
Dictionnaire	30	un nombre entier positif N correspondant au nombre de couples clé-valeur du dictionnaire. Suivent ensuite N séquences clé-valeurs du dictionnaire.	Oui
Array	31	un nombre entier positif N correspondant au nombre d'éléments du tableau. Suivent ensuite N séquences correspondants aux éléments du tableau.	Oui
Couple	32	2 séquences de tokens correspondant au premier et au second membre du couple d'objets.	Oui
inutilisées	33 - 49		
Type personalisé	50	l'objet contenant le nom du type personalisé suivi d'un nombre entier positif N correspondant au nombre de couples clé-valeur du dictionnaire. Suivent ensuite N séquences clé-valeurs du dictionnaire.	Oui

Tokens MSTE0101

Type	Code	Suite attendue	Ref
Objet null	0	Aucune	Non
Valeur "vraie"	1	Aucune	Non
Valeur "fausse"	2	Aucune	Non
Entier	3	un nombre entier sans limite de stockage	Oui
Décimal	4	un nombre décimal sans limite de stockage	Oui
Chaîne de caractères	5	la chaîne de caractères	Oui
		un nombre entier (positif ou négatif) de secondes	

Time stamp	6	écoulées ou à venir depuis ou vers le **01/01/1970** qui est la référence (**Unix Epoch**).	Oui
Couleur	7	un nombre entier positif contenant la représentation en 24 ou 32 bits de la couleur. 24 bits = RRGGBB. 32 bits = TTRRGGBB ou TT est la transparence (0x00 = opaque, 0xFF = totalement transparent)	Oui
Dictionnaire	8	un nombre entier positif N correspondant au nombre de couples clé-valeur du dictionnaire. Suivent ensuite N séquences clé-valeurs du dictionnaire.	Oui
Référence	9	1 token contenant l'index dans le tableau des objets déjà décodés de l'objet à utiliser comme référence **FORTE**.	Non
int8	10	un nombre compris entre -128 et +127	Non
uint8	11	un nombre compris entre 0 et 255	Non
int16	12	un nombre compris entre -32768 et +32767	Non
uint16	13	un nombre compris entre 0 et 65535	Non
int32	14	un nombre compris entre -(2^31) et (2^31)-1	Non
uint32	15	un nombre compris entre 0 et (2^32)-1	Non
int64	16	un nombre compris entre -(2^63) et (2^63)-1	Non
uint64	17	un nombre compris entre 0 et (2^64)-1	Non
float	18	un nombre flottant interprétable en simple précision	Non
double	19	un nombre flottant interprétable en double précision	Non
Array	20	un nombre entier positif N correspondant au nombre d'éléments du tableau. Suivent ensuite N séquences correspondants aux éléments du tableau.	Oui
Natural array	21	un nombre entier positif N correspondant au nombre d'entiers naturels positif contenus dans le tableau suivi de N entiers naturels	Oui
Couple	22	2 séquences de tokens correspondant au premier et au second membre du couple d'objets.	Oui
Data	23	un nombre entier positif correspondant à la longueur originale en octets de la donnée transmise, suivi d'une chaîne de caractères contenant les données encodées en Base64.	Oui
Date passé	24	Date infinie dans le passé.	Oui

Date future	25	Date infinie dans le futur.	Oui
Chaîne vide	26	Aucune	Non
Référence	27	1 token contenant l'index dans le tableau des objets déjà décodés de l'objet à utiliser comme référence **FAIBLE**.	Non
Type personalisé	50	l'objet contenant le nom du type personalisé suivi d'un nombre entier positif N correspondant au nombre de couples clé-valeur du dictionnaire. Suivent ensuite N séquences clé-valeurs du dictionnaire.	Oui
Type personalisé	51	l'objet contenant le nom du type personalisé suivi d'un nombre entier positif N correspondant au nombre de couples clé-valeur du dictionnaire. Suivent ensuite N séquences clé-valeurs du dictionnaire.	Oui

Exemples

Une chaîne de caractères

```
"toto" // json
["MSTE0102",7,"CRCD45ACB10",0,0,21,"toto"]
["MSTE0101",7,"CRC2B8F345A",0,0,5,"toto"]
```

Un tableau

```
["toto"] // json
["MSTE0102",9,"CRCD4E14B75",0,0,31,1,21,"toto"]
["MSTE0101",9,"CRC43E85E76",0,0,20,1,5,"toto"]
```

```
["toto", "tata", "toto"] // js
["MSTE0102",13,"CRC7311752F",0,0,31,3,21,"toto",21,"tata",9,1]
["MSTE0101",13,"CRC0F51FFA9",0,0,20,3,5,"toto",5,"tata",9,1]
```

Un dictionnaire

```
{"mykey":"toto"} // js
["MSTE0200",30,1,"mykey",21,"toto"]
["MSTE0102",11,"CRC1C9E9FE1",0,1,"mykey",30,1,0,21,"toto"]
["MSTE0101",11,"CRC7356C782",0,1,"mykey",8,1,0,5,"toto"]
```

```
[{"mykey":"toto"}, {"mykey":"toto"}] // js
["MSTE0102",18,"CRCDF6E36C0",0,1,"mykey",31,2,30,1,0,21,"toto",30,1,0,9,2]
["MSTE0101",18,"CRCCA3A73E2",0,1,"mykey",20,2,8,1,0,5,"toto",8,1,0,9,2]
```

```
t = {"mykey":"toto"}, [t, t] // js
["MSTE0102",15,"CRCFFC790D3",0,1,"mykey",31,2,30,1,0,21,"toto",9,1]
["MSTE0101",15,"CRCB3BA14EE",0,1,"mykey",20,2,8,1,0,5,"toto",9,1]
```

Un type personalisé

```
class Person {
    firstName: string;
    lastName: string;
    mother?: Person;
    father?: Person;
    childrens: Person[];

    constructor(initValues: { firstName: string, lastName: string }) { ...
}
    setMSTEValues(values: { mother?: Person, father?: Person, childrens:
Person[] }) { ... }
}
let son = new Person({ firstName:"Mickey", lastName: "Mouse" });
let mother = new Person({ firstName:"Mother", lastName: "Mouse" });
let father = new Person({ firstName:"Father", lastName: "Mouse" });
son.mother = mother; mother.childrens.push(son);
son.father = father; father.childrens.push(son);
son;

["MSTE0102",49,"CRCAF1171C0",0,5,"childrens","firstName","lastName","mothe
r","father",30,5,0,31,0,1,21,"Mickey",2,21,"Mouse",3,30,3,0,31,1,9,0,1,21,
"Mother",2,9,3,4,30,3,0,31,1,9,0,1,21,"Father",2,9,3]
```