

**BeSt Services User Guide**

Date: 29/11/2021

Version: 1.7

# 

## Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Modified by | Modification |
| 0.1 | 11/02/19 | Gert De Jonge | First draft |
| 0.2 | 02/04/19 | Gert De Jonge | Modifications |
| 0.3 | 16/04/19 | Eddy Corthouts | Modifications and formatting |
| 0.4 | 16/04/19 | Gert De Jonge | Added life cycle address objectIdentifier chapter |
| 0.5 | 25/04/19 | Gert De Jonge | Small modifications |
| 0.6 | 06/05/19 | Gert De Jonge | Added life cycle ObjectIdentifier for regions |
| 1.0 | 09/05/19 | Gert De Jonge | Added information MFT services + reshaping the request/response tables + grammar check |
| 1.1 | 14/05/2019 | Gert De Jonge | Changed filename convention for MFT service |
| 1.2 | 20/06/2019 | Gert De Jonge | Adjustments release data + removing services that are not yet available + renaming |
| 1.3 | 10/09/2019 | Gert De Jonge | Updating the lifecycle of an ID matrix |
| 1.4 | 24/10/2019 | Eddy Corthouts | Update planning dates, update descriptions S354, S355 |
| 1.5 | 24/03/2020 | Eddy Corthouts | Update section 4, ‘MFT services’ |
| 1.6 | 18/03/2021 | Eddy Corthouts | Update section 4, ‘MFT services’ Processing of region files by BOSA, replace figure 1  Update descriptions of BeginLifeSpanVersion and EndLifeSpanversion |
| 1.7 | 29/11/2021 | Eddy Corthouts | Added section 6, Open data  Updated section 4.2, S350, “Address mutations file service”. Mutations are being reworked to achieve full harmonisation between the different regions.  A new mutations process is planned for release in Q2 2022 |

## Conventions

|  |  |
| --- | --- |
| Font | use |
| Italic | accentuation |

## Purpose of this document

This document describes the BeSt Address Web- and MFT services.

## Intended Audience

This document is addressed at any developer or analyst who wants to make use of the BeSt Address services

## Contact information

|  |  |
| --- | --- |
| Service Owner | Sebastiaan Taes  [Sebastiaan.Taes@bosa.fgov.be](mailto:Sebastiaan.Taes@bosa.fgov.be) |
| Service Desk | [ServiceDesk@bosa.fgov.be](mailto:ServiceDesk@bosa.fgov.be)  +32 78 150312  +32 2 2129674 |
| Service Release Date | TBD |

All BeSt services are subject to the latest BOSA FSB Terms & Conditions, such as described in [this document](file:///C:/Users/snichelson/Federale%20Horizontale%20Overheidsdiensten/OneDrive%20-%20Federale%20Horizontale%20Overheidsdiensten/BeST%20Business%20Analyse/WIP/fsb_governance_guidelines_fsb-user-guide-for-service-consumers%20NL%20v2.6.docx). The document describes the governance principles of the BOSA Service Bus as well.

Table of Contents

[Version History i](#_Toc89091285)

[Conventions i](#_Toc89091286)

[Purpose of this document i](#_Toc89091287)

[Intended Audience i](#_Toc89091288)

[Contact information ii](#_Toc89091289)

[1 Introduction 1](#_Toc89091290)

[1.1 BeSt Application Overview 1](#_Toc89091291)

[1.2 Web services 2](#_Toc89091292)

[1.3 MFT services 3](#_Toc89091293)

[2 Data Model 4](#_Toc89091294)

[2.1 The Identifier life cycle 7](#_Toc89091295)

[3 Webservices 12](#_Toc89091296)

[3.1 S352 - SearchMunicipalityService 13](#_Toc89091297)

[3.2 S353 - SearchStreetnameService 15](#_Toc89091298)

[3.3 S354 - SearchAddressService 17](#_Toc89091299)

[3.4 Webservices – Errors 21](#_Toc89091300)

[4 MFT Services 22](#_Toc89091301)

[4.1 S349 - FullDownloadService 22](#_Toc89091302)

[4.2 S350 - AddressMutationsFileService 25](#_Toc89091303)

[6 Open Data 32](#_Toc89091304)

[7 Webservices - Technical 33](#_Toc89091305)

[7.1 Service access parameters 33](#_Toc89091306)

[8 Known issues 34](#_Toc89091307)

[8.1 Flanders 34](#_Toc89091308)

[8.2 Brussels 34](#_Toc89091309)

[8.3 Wallonia 34](#_Toc89091310)

[Document Information 35](#_Toc89091311)

[General 35](#_Toc89091312)

[Approbation 35](#_Toc89091313)

[Distribution 35](#_Toc89091314)

**List of Tables**

[Table 1, ‘Region Life Cycle Models’ 11](#_Toc11910877)

[Table 2, ‘Simplified Delete mutation table’ 28](#_Toc11910878)

[Table 3, ‘Simplified Add mutation table’ 28](#_Toc11910879)

**List of Figures**

[Figure 1, ‘BeSt Application Overview’ 1](#_Toc11910872)

[Figure 2, ‘Inspire Address model’ (next page) 4](#_Toc11910873)

[Figure 3, ‘Best Services Address Model’ 6](#_Toc11910874)

[Figure 4, ‘Full download data structure’ 24](#_Toc11910875)

[Figure 5, ‘Mutations data structure’ 27](#_Toc11910876)

# Introduction

BeSt stands for “Belgian Street Codes”. The objective of the BeST Address service project is to provide address services on a federal level that will bundle address information coming from the three regional master data sets.

Following organizations have participated in the development and implementation of BeSt:

• The National Geographic Institute (NGI)

• The General Administration of the Patrimony Documentation (AAPD) from the FPS Finance

• The National Registry (NR) from the FPS Internal Affairs

• Statistics Belgium from the FPS Economy

• The Crossroad Bank for Enterprises (CBE) from the FPS Economy

• The Directorate general Security and Prevention from the FPS Internal Affairs

• The FPS Governance and Support (BOSA)

• The Agency for Administrative Rationalization (DAV)

• The supplier of the universal postal services

### BeSt Application Overview

Each region (Flemish, Brussels, Wallonia) has its’ own address master data set (“authentic source”) that contains the data to uniquely identify all addresses in the region.

The BeST Address service project provides address services that bundle the information of these 3 data sources on a federal level.

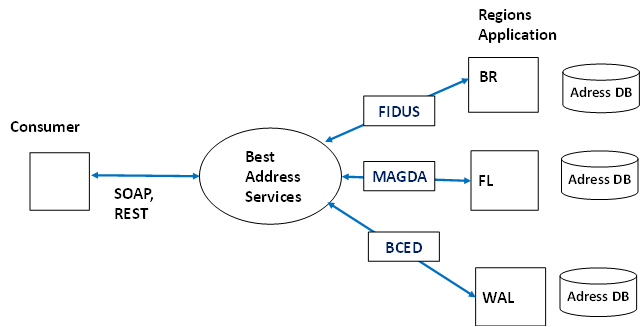


Figure 1, ‘BeSt Application Overview’

***Types of services***

Two types of services are available:

* Webservices
* Managed File Transfer Services

The following table provides an overview of the BeSt Address services and their planned availability date:

|  |  |
| --- | --- |
| **Webservices** | **Availability Date** |
| S352 – SearchMunicipalityService | 22/07/2019 |
| S353 – SearchStreetnameService | 22/07/2019 |
| S354 – SearchAddressService | 22/07/2019 |
|  |  |
| *S355 – SearchAddressRepresentationService* | *Q4 2022* |
| *S359 – SearchPartOfMunicipalityService* | *Q4 2022* |
| *S357 – SearchStreetnameHistoryService* | *Q4 2022* |
| *S358 – SearchAddressHistoryService* | *Q4 2022* |
| *S356 – AnomalyService* | *Q4 2022* |

|  |  |
| --- | --- |
| **MFT services** | **Availability Date** |
| S349 – FullDownloadService | 15/05/2019 |
| S350 – AddressMutationsFileService | *Q2 2022* |

***SOAP and REST***

Consumers can invoke the BeSt services using SOAP or REST.

The REST version is implemented with XML body (not JSON).

For Brussels and Wallonia, the BeSt webservice requests and replies transit through the respective service bus of these regions (Brussels: FIDUS, Wallonia: BCED).

For Flanders, the requests and replies do not pass via a service bus.

### Web services

The webservices are aimed at customers who want to consult address information on a “request by request” basis.

***Routing of Webservice requests***

The objective of routing BeSt-requests to the right region by BOSA is to reduce the traffic between

BOSA and the backend. If 50% of the requests can be routed, this results in a total reduction of traffic

of 33%.

There are two possible execution scenarios for all webservices:

*Without* routing:

There is no way to identify the destination region with full certainty. The request is sent to all regions. The individual responses of the regions are joined by BoSa and the combined response is sent to the consumer.

*With* routing:

The destination region can be identified with certainty. The request is only sent to this region. BOSA receives the response from this region and joins it with an errorOrWarningtype for the other regions:

<xs:enumeration value="007 - Source: No search performed due to routing"/>

If the request contains at least one of the following elements, the destination region can be

identified.

* All Identifier namespaces: namespace of municipalitycode, streetnamecode, addresscode, postalcode, partofMunicipalitycode. If the request contains any Identifier <namespace>, that has an

exact match with the fixed namespace list from the application.

* Municipalitycode: objectIdentifier: all municipalitycodes’ <objectIdentifier> if there is an

exact match with a NIScode in the NIS-list of municipalities.

* Postalcode: objectIdentifier: all postalcodes’ <objectIdentifier> if there is an exact

match with a postalcode in the postalcode list.

* MunicipalityName: All municipalitynames’ <spelling> if there is a match with the spelling in

the NIS-list of municipalitynames that enables BOSA to exclude at least one region.

If there are several elements present in the request that allow routing, the following priority will be

given:

1. Namespace municipality

2. Namespace streetname

3. Namespace address

4. Namespace postalinfo

5. Namespace PartofMunicipality

6. Municipalitycode: objectIdentifier

7. Postalinfo: objectIdentifier

8. MunicipalityName

### MFT services

MFT stands for ‘Managed File Transfer’. The three MFT services allow the user to download address information by means of files. The MFT services are aimed at customers that want to download a lot or all address information.

# Data Model

The next diagram shows the underlying data model which the services’ requests and responses are based on.

The model is based on the European Inspire data model.

The INSPIRE Directive aims to create a European Union spatial data infrastructure for the purposes of EU environmental policies and policies or activities which may have an impact on the environment. This European Spatial Data Infrastructure will enable the sharing of environmental spatial information among public sector organizations, facilitate public access to spatial information across Europe and assist in policy-making across boundaries.

For more information on the address theme, go to <https://inspire.ec.europa.eu/Themes/79/2892>

Figure 2, ‘Inspire Address model’ (next page)





Figure 3, ‘Best Services Address Model’

Below the Enterprise Architect EAP file, for better comprehension:



The Address entity is the central entity in the model: every entity is attached to it.

An address is a collection of other entities like a Street, a Municipality, possibly a part-of-municipality (only Wallonia) and a Postal info.

As we look at the model, we can see following relations:

* Every address has a street
* Every address has a municipality
* Every address has a postal code
* Every part of municipality has a municipality
* Every street has a municipality

In the Inspire based model we can see that a logical relation in real life: the link between a postal code and a municipality, is not present. This means that we cannot derive a postal code from a municipality and vice versa.

The address representation entity is a summary of an address: it will give you all the essential attributes of an address in full writing (so no pointers to objectIdentifiers)

Each attribute has a functional meaning and some of them only allow certain values. This is further clarified in the sections that explain the services. Each response is fully detailed and contains all attribute explanations.

### The Identifier life cycle

The life cycle of an Identifier (address, street, municipality…) is treated differently among the regions. It is important to know how a region treats the lifecycle of Identifiers.

In general

Below, the global definition (taken from the data model) of an Identifier in BeST:



An Identifier (of any kind) is composed of 3 attributes:

* namespace
* objectIdentifier
* versionIdentifier

These 3 fields uniquely identify a street, a municipality, an address,…

A versionIdentifier could be a datetimestamp (as Wallonia and Flanders do) or just a number (like Brussels does).

Example of an address Identifier:

Namespace: <https://data.vlaanderen.be/id/adres>

objectIdentifier: 12345

versionIdentifier: 18/04/2019 12:35

The persistence of an ID (Identifier)

The regions are owner of the address data and therefore decide the business processes concerning the persistence of an address Identifier. Those business processes differ among the regions.

NOTE: when an Identifier changes, it could mean a couple of things:

* the namespace changes
* the objectIdentifier changes
* the version changes

In the table on the following page, we make a difference between the objectIdentifier and the version. In an exceptional case the namespace could change but this would be very seldom (or never).

Note: if the ‘version’ cell is empty, this means that a new version (number, date…) is generated for the new objectIdentifier.

Example: It is possible that for a certain use case A the objectIdentifier of an address can change in region 1 but remains the same in region 2 and only the version is increased.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Delta / Impact** |  | **Sel** | **Municipality ID  - *objectIdentifier*** | **Municipality ID  - *version*** | **Streetname ID  - *objectIdentifier*** | **Streetname ID  - *version*** | **Adres ID  - *objectIdentifier*** | **Adres ID  - *version*** |
| 1 | Streetname change |  |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | = | changes | = | = |
|  |  |  | WAL | NA | NA | = | changes | = | = |
|  |  |  | FL | NA | NA | changes |  | changes |  |
| 2 | Adres number change - house or appartement number |  |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | NA | NA | = | changes |
|  |  |  | WAL | NA | NA | NA | NA | = | changes |
|  |  |  | FL | NA | NA | NA | NA | changes |  |
| 3 | Municipality name change |  |  |  |  |  |  |  |  |
|  |  |  | BRU | = | changes | = | = | = | = |
|  |  |  | WAL | = | (1) | (2) |  | (2) |  |
|  |  |  | FL | changes |  | changes |  | changes |  |
| 4 | SPLIT: Streetname split into 2 Streetnames |  |  |  |  |  |  |  |  |
|  |  | Old Street |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | = | = | = | = |
|  |  |  | WAL | NA | NA | = | = | = | = |
|  |  |  | FL | NA | NA | = | = | = | = |
|  |  | New Street |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | changes |  | = | changes |
|  |  |  | WAL | NA | NA | changes |  | = | changes |
|  |  |  | FL | NA | NA | changes |  | changes |  |
| 5 | JOIN Streetname: 2 streetnames merged into one new streetname (A+B = A) |  |  |  |  |  |  |  |  |
|  |  | Old Street 1 (B) |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | Not Tested | Not Tested | Not Tested | Not Tested |
|  |  |  | WAL | NA | NA | NA | NA | = | changes |
|  |  |  | FL | NA | NA | NA | NA | Changes for addresses sent to the new street |  |
|  |  | Remaining Street A |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | Not Tested | Not Tested | Not Tested | Not Tested |
|  |  |  | WAL | NA | NA | = | = | = | = |
|  |  |  | FL | NA | NA | = | = | = | = |
| 5 | JOIN Streetname: 2 streetnames merged into one new streetname (A+B = C) |  |  |  |  |  |  |  |  |
|  |  | Old Street 1: A |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | Not Tested | Not Tested | Not Tested | Not Tested |
|  |  |  | WAL | NA | NA | NA | NA | NA | NA |
|  |  |  | FL | NA | NA | NA | NA | Changes for addresses sent to the new street |  |
|  |  | Old Street 2: B |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | Not Tested | Not Tested | Not Tested | Not Tested |
|  |  |  | WAL | NA | NA | NA | NA | NA | NA |
|  |  |  | FL | NA | NA | New street has a new id |  | Changes for addresses sent to the new street |  |
|  |  | New Street C |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | Not Tested | Not Tested | Not Tested | Not Tested |
|  |  |  | WAL | NA | NA | NA | NA | NA | NA |
|  |  |  | FL | NA | NA | New street has a new id |  | Changes for addresses sent to the new street |  |
| 6 | Geo X-Y coordinate change  possible cases: -a reconstruction of a building -a correction of an error |  |  |  |  |  |  |  |  |
|  |  |  | BRU | NA | NA | NA | NA | = | = |
|  |  |  | WAL | NA | NA | NA | NA | = | changes |
|  |  |  | FL | NA | NA | NA | NA | = | changes |

For Brussels there are some cases that still need to be tested

Table 1, ‘Region Life Cycle Models’

# Webservices

Below, the following webservices are described in detail:

* S352 - SearchMunicipalityService
* S353 - SearchStreetnameService
* S354 - SearchAddressService

The following principles apply for all webservices:

* Unless routing can be applied (see section 1.2 ), each webservice will perform its search by accessing 3 sources. These sources represent the 3 regions in Belgium: Flanders, Brussels and Wallonia.
* All webservices follow the principle that when an objectIdentifier is given (in the request), the response will contain the last version of that objectIdentifier (when there is no versionnumber specified in the request)
* Empty requests (no search attributes present) will result in an “nothing found” business error for each region.
* The response is either a successful response when results have been found, a business error response or a technical error response.

### S352 - SearchMunicipalityService

This service returns a municipality or a list of municipalities based on the search criteria provided.

***Basic concepts***

The service logic is based on the AND operator.

This means that the search is being performed using ALL filled out parameters. So, if the user fills out the objectIdentifier of the municipality and (a part of) a name, the search logic will use these 2 parameters in combination to search for the right municipality.

Example: when the users fills out ‘Brussels’ and ‘2342’, the search logic uses these parameters in an AND clause: searching for municipalities with name = ‘Brussels’ AND ObjectIdentifier = 2342

Request

|  |  |  |
| --- | --- | --- |
| Input parameter | Description | Type |
| Identifier |  | Identifiersearchtype |
| Identifier:nameSpace | Namespace of the municipalityCode. Assigned per region. | NameSpace |
| Identifier:objectIdentifier | The NIS code of the municipality. | String |
| Identifier:versionIdentifier | The version Identifier of the municipalityCode | String |
| municipalityName |  | GeographicalNameSearchType |
| municipalityName:spelling | Here the user can fill out (a part of) the name of the municipality | String |
| municipalityName:language | Language of the municipalityName (or part of) : GeographicalName. Dutch, French or German | LanguageCodeValueType |
| muncicipaliltyName:searchType | The type of search that has to be performed on the spelling of the municipalityName: ‘contains’, ‘equals, ‘phonetic’ | SearchType |

Response

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output parameter | Description | Type | Min. Occurs | Max. Occurs |
| municipalityCode |  | IdentifierType | 1 | 1 |
| municipalityCode: namespace | namespace of the municipality | charStringType | 1 | 1 |
| municipalityCode: objectIdentifier | objectIdentifier of the municipality (NIS code) | charStringType | 1 | 1 |
| municipalityCode: versionIdentifier | versionIdentifier of the municipality | charStringType | 0 | 1 |
| municipalityName |  | GeographicalNameType | 0 | n |
| municipalityName: language | Language of the municipality | LanguageCodeValueType | 1 | 1 |
| municipalityName: spelling | The municipality name | String | 1 | 1 |

### S353 - SearchStreetnameService

This service returns a street with its attributes or a list of streetnames based on the search criteria provided.

***Basic concepts***

The service logic is based on the AND operator.

This means that the search is being performed using ALL filled out parameters. So if the user fills out the objectIdentifier of the street and (a part of) a name, the search logic will use these 2 parameters in combination to search for the right municipality.

Example: when the users fills out the name ‘Stationstraat’ and ObjectIdentifier ‘2342’, the search logic uses these parameters in an AND clause: searching for streets with name = ‘Stationstraat’ AND ObjectIdentifier = 2342

Request

|  |  |  |
| --- | --- | --- |
| Input parameter | Description | Type |
| Language | The language that will be used for retrieving the street | LanguageCodeValueType |
| municipalityCode |  | Identifiersearchtype |
| municipalityCode: namespace | Namespace of the municipalityCode. Assigned per region. | NameSpace |
| municipalityCode: objectIdentifier | The code of the municipality. | String |
| municipalityCode: versionIdentifier | The version Identifier of the municipalityCode | String |
| streetName |  | GeographicalNameSearchType |
| streetName: spelling | Spelling of the streetName (or part of) | String |
| streetName: language | Language of the streetName: Dutch, French or German | LanguageCodeValueType |
| streetName: SearchType | The type of search that has to be performed on the spelling of the streetName: ‘contains’, ‘equals, ‘phonetic’ | SearchType |
| streetNameCode |  | Identifiersearchtype |
| streetNameCode: namespace | Namespace of the streetnameCode. Assigned per region. | NameSpace |
| streetNameCode: objectIdentifier | The objectIdentifier of the Streetname. | String |
| streetNameCode: versionIdentifier | The version Identifier of the streetCode | String |
| streetNameStatus | 4 possible statuses:   * Proposed * Reserved * Current * Archived | StreetnameStatusValueType |
| streetNameType | 2 possible values:   * Hamlet * Street | StreetnameTypeValueType |

Response

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output parameter | Description | Type | Min. Occurs | Max. Occurs |
| homonymAddition | A homonym for this streetname | CharStringtype | 0 | 1 |
| streetnameCode |  | IdenifierType | 1 | 1 |
| streetnameCode: namespace | Namespace of the street | CharStringtype | 1 | 1 |
| streetnameCode: objectIdentifier | Objectidentifier of the street | CharStringtype | 1 | 1 |
| streetnameCode: versionIdentifier | Versionidentifier of the street | CharStringtype | 0 | 1 |
| Streetname |  | GeographicalNameType | 1 | n |
| streetName: spelling | Name of the street | String | 1 | 1 |
| streetname: language | Language | LanguageCodeValueType | 1 | 1 |
| streetnameStatus |  | streetnameStatusType | 1 | 1 |
| streetnameStatus:  Status | The status of the street | streetnameStatusvalueType | 1 | 1 |
| streetnameStatus: validFrom | Begin date of the status | dateTime | 1 | 1 |
| streetnameStatus: validTo | End date of the status | dateTime | 0 | 1 |
| streetnameType | Type of street | streetNameTypeValueType | 1 | 1 |
| isAssignedBy : Municipality |  | LinkType | 0 | 1 |
| Municipality:  namespace | Namespace of the municipality | CharStringtype | 1 | 1 |
| Municipality : objectIdentifier | objectIdentifier of the municipality | CharStringtype | 1 | 1 |
| Municipality: versionIdentifier | versionIdentifier of the municipality | CharStringtype | 0 | 1 |
| isAssignedTo: RoadObject |  | RoadObject (LinkType) | 0 | n |
| isAssignedTo: streetSide |  | SideCodeValueType | 0 | n |
| Beginlifespanversion | date and time at which this version of the object was inserted or changed in the database | DateTime | 1 | 1 |
| Endlifespanversion | date and time at which this version of the object was superseded or retired in the database | Datetime | 0 | 1 |

Note: the values for attributes *isAssignedTo:RoadObject* and *isAssignedTo:streetSide* are not (yet) present in the responses. Neither is this information present in the full download.

### S354 - SearchAddressService

This service returns an address with its attributes or a list of addresses based on the search criteria provided. The reply contains pointers to streetname and municipality (not a written description of the streetname nor the municipality).

***Basic concepts***

The service logic is based on the AND operator.

This means that the search is being performed using ALL filled out parameters. So if the user fills out the objectIdentifier of an address and (a part of) a streetname, the search logic will use these 2 parameters in combination to search for the right address.

Example: when the users fills out ‘Stationsstraat’ and housenumber ‘22’, the search logic uses these parameters in an AND clause: searching for addresses with streetname = ‘Stationsstraat’ AND housenumber = 22

Note:

The concept “Part of Municipality” is only used in Wallonia. Brussels and Flanders do not have ‘part of municipality’ in their data. Officially, this concept does not exist anymore in these 2 regions. This means that, whenever there is a request that contains a ‘part-of-municipality name’ or ‘part-of-municipality ID’, the response will be empty and the following error will be returned: “this concept is not available for Flanders/Brussels”.

Request

|  |  |  |
| --- | --- | --- |
| Input parameter | Description | Type |
| addressCode |  | IdentifierSearchType |
| addressCode: namespace | Namespace of the address | NameSpace |
| addressCode: objectIdentifier | objectIdentifier of the address | String |
| addressCode: versionIdentifier | VersionIdentifier of the address | String |
| addressStatus | 4 possible values:   * Current * Proposed * Reserved * Retired | AddressStatusValueType |
| boxNumber | Boxnumber | String |
| houseNumber | Housenumber | String |
| municipalityCode |  | IdentifierSearchType |
| municipalityCode: namespace | Namespace of the municipality. Assigned per region. | NameSpace |
| municipalityCode: objectIdentifier | The code of the municipality. | String |
| municipalityCode: versionIdentifier | The version Identifier of the municipalityCode | String |
| municipalityName |  | GeographicalNameSearchType |
| municipalityName: spelling | Spelling of the municipalityname (or part of) | String |
| municipalityName: language | Language of the municipality: Dutch, French or German | LanguageCodeValueType |
| municipalityName: SearchType | The type of search that has to be performed on the spelling of the municipalityName: ‘contains’, ‘equals, ‘phonetic’ | SearchType |
| PartOfMuniciaplityCode |  | IdentifierSearchType |
| partOfMunicipalityCode: namespace | Namespace of the partOfMunicipalityCode. Assigned per region. | NameSpace |
| partOfMunicipalityCode: objectIdentifier | The objectIdentifier of the partOfMunicipality. | String |
| partOfMunicipalityCode: versionIdentifier | The version Identifier of the partOfMunicipalityCode | String |
| PartOfMunicipalityName |  | GeographicalNameSearchType |
| partOfMunicipalityName: spelling | Spelling of the part-of-mun (or part of) | String |
| partOfMunicipalityName: language | Language of the part-of-mun: Dutch, French or German | LanguageCodeValueType |
| partOfMunicipalityName: SearchType | The type of search that has to be performed on the spelling of the part-of-mun name: ‘contains’, ‘equals, ‘phonetic’ | SearchType |
| postCode |  | IdentifierSearchType |
| postCode:  namespace | NameSpace of the postcode | NameSpace |
| postCode:  objectIdentifier | objectIdentifier of the postcode | String |
| PostCode: VersionIdentifier | versionIdentifier of the postcode | String |
| streetName |  | GeographicalNameSearchType |
|  |  |  |
| streetName:  spelling | Spelling of the streetName (or part of) | String |
| streetName:  language | Language of the streetName: Dutch, French or German | LanguageCodeValueType |
| streetName:  SearchType | The type of search that has to be performed on the spelling of the street name: ‘contains’, ‘equals, ‘phonetic’ | SearchType |
| streetNameCode |  | Identifiersearchtype |
| streetNameCode: namespace | Namespace of the streetName. Assigned per region. | NameSpace |
| streetNameCode: objectIdentifier | The objectIdentifier of the Streetname. | String |
| streetNameCode: versionIdentifier | The version Identifier of the streetname | String |

Response

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Output parameter | Description | Type | Min. Occurs | Max. Occurs |
| addressCode |  | IdentifierType | 1 | 1 |
| addressCode: NameSpace | Namespace of the address | CharStringType | 1 | 1 |
| addresscode: ObjectIdentifier | ObjectIdentifier of the address | CharStringType | 1 | 1 |
| addressCode: versionIdentifier | versionIdentifier of the address | CharStringType | 0 | 1 |
| addressPosition |  | GeographicalPositionType | 1 | 1 |
| addressPosition: PointGeometry:gml:Point: gml:Id | Identifier of the point | Gml:PointType | 1 | 1 |
| addressPosition: PointGeometry:gml:Point: gml:pos | Position of the point | DirectPositionType | 1 | 1 |
| addressPosition: positionGeometryMethod | The manner how this point was defined | PositionGeometry-MethodValueType | 1 | 1 |
| addressPosition: positionSpecification | The object on which the point was defined | PositionSpecification-ValueType | 1 | 1 |
| Addresssortfield | Transformation of the house number and the box number (eg. By adding extra 0’s before) so this value can be sorted | CharStringType | 0 | 1 |
| addressStatus |  | AddressStatusType | 1 | 1 |
| addressStatus:  status | Status of the address | AddressStatusValueType | 1 | 1 |
| addressStatus:  validFrom | Begin date of the status | dateTime | 1 | 1 |
| addressStatus:  validTo | End date of the status | dateTime | 0 | 1 |
| boxNumber | The box number associated to the address, if any. | CharStringType | 0 | 1 |
| houseNumber | The house number associated to the address | CharStringType | 1 | 1 |
| officiallyAssigned | Declares if the address is officialy granted (True/False) | Boolean | 1 | 1 |
| hasStreetname: streetName |  | LinkType | 1 | 1 |
| streetName:  namespace | Namespace of the street | CharStringType | 1 | 1 |
| streetName: objectIdentifier | objectIdentifier of the street | CharStringType | 1 | 1 |
| streetName: versionIdentifier | versionIdentifier of the street | CharStringType | 0 | 1 |
| hasMunicipality: Municipality |  | LinkType | 1 | 1 |
| municipality:  namespace | Namespace of the municipality | CharStringType | 1 | 1 |
| municipality: objectIdentifier | objectIdentifier of the municipality | CharStringType | 1 | 1 |
| municipality: versionIdentifier | versionIdentifier of the municipality | CharStringType | 0 | 1 |
| hasPostalInfo: PostalInfo |  | LinkType | 1 | 1 |
| postalInfo:  namespace | Namespace of the postalInfo | CharStringType | 1 | 1 |
| postalInfo: objectIdentifier | objectIdentifier of the postalInfo | CharStringType | 1 | 1 |
| postalInfo: versionIdentifier | versionIdentifier of the postalInfo | CharStringType | 0 | 1 |
| isAssignedTo addressable object | (Foreseen for later stages of project) | LinkType | 0 | n |
| isSituatedIn: PartofMunicipality |  | LinkType | 0 | 1 |
| partOfMunicipality:  namespace | Namespace of the part-of-mun | CharStringType | 1 | 1 |
| partOfMunicipality: objectIdentifier | objectIdentifier of the part-of-mun | CharStringType | 1 | 1 |
| partOfMunicipality: versionIdentifier | versionIdentifier of the part-of-mun | CharStringType | 0 | 1 |
| Beginlifespanversion | date and time at which this version of the object was inserted or changed in the database | dateTime | 1 | 1 |
| Endlifespanversion | date and time at which this version of the object was superseded or retired in the database | dateTime | 0 | 1 |

## Webservices – Errors

Below, the errors that may occur when using the webservices are listed.

The errors are divided into technical errors and business errors

***Technical errors***

001 – Main Error: Back-end not yet available

Occurs when one of the sources (or all) is not available due to network or other problems.

002 – Main Error: minimum parameters not filled in

003 – Main Error: WSDL Validation

004 – Main Error: XSD Validation

Occurs when the request is not valid against the predefined XSD. This could mean that an attribute is wrongfully used or is missing when it should be present.

***Business errors***

005 – Source: Too many results

Occurs when the request has too many results to send through the service. Further specification in the request could solve this.

Note:: this is an error that the regions throw when the amount of results transcends the limit that is set. Example: when a region sets its limits at 100 responses and the request has a result of 102 responses, this error will be thrown.

006 – Source: Nothing found

Occurs when the parameters in the request do not produce a response (or an empty response)

007 – Source: No search performed due to routing

Occurs when the request contains a parameter that can be used to route the request towards one region. The other regions will throw this error.

008 – Source: Back-end error

Occurs when an unexpected problem happened on the side of the source

009 – Source: Time out

Occurs when the request has been launched towards one (or more) regions but the response takes too much time to be sent back.

010 – Flanders and Brussels do not support searches based on ‘part-of-municipality’.

Occurs when a request is launched containing ‘part-of-municipality’. Flanders and Brussels do not support this so BOSA will throw this error for those 2 regions.

# MFT Services

Below, the following MFT services are described in detail:

* S349 - FullDownloadService
* S350 - AddressMutationsFileService

***Availability for Consumer***

The files will be available on the BOSA server for downloading at the following time:

|  |  |  |  |
| --- | --- | --- | --- |
| **Service** | **File Name** | **Frequency** | **Time** |
| S349 – FullDownloadService | FDBelgiumxxxxxxxx.zip | Weekly | Monday 02:00 AM |
| S350 - AdresMutationsFileService | MBelgiumxxxxxxxx.zip | Daily on weekdays | Tue, We, Th, Fr – 02:00 AM |

Important: it is necessary that every client that uses the Full Download uses the mutations . All the changes that were made during the week (between two full download dates) will be available in the mutations and spatial transactions. These are necessary to keep one’s own address database up-to-date.

***Retention Policy***

BOSA keeps the BeSt download files available for 30 days.

***Processing of incoming Region files by BOSA***

***Processing of Region files by BOSA***

BOSA will pick up 1 full download per week on Sundays from 23:00 PM from each of the regions.

BOSA will pick up mutations on a daily basis (week days) from 23:00 onwards

BOSA does not delete the files from the region pick-up sites.

BOSA will prepare the consumer download files by 01:00

BOSA processing of Region files encompasses:

* validate the files are conform with the XSD’s
* zip the files into a single zip file for downloading by consumers
* This process takes about 15 min.

As the files are processed in the night, it is assumed that no mutations can occur during the update window.

### S349 - FullDownloadService

***Main functionality***

The Full Download allows a user to download by means of MFT a zip file that contains one file for each entity of the BeSt data model per region. These files contain for each Region the latest versions of the following entities:

* municipalities
* streetnames
* addresses
* partofmunicipalities (Wallonia only)
* postalinfo

Each region produces 5 separate files containing the latest version of that specific entity. If on a certain day, no files would be received from a region for one or more entities, BOSA will place the previous version received in the BeSt full download for those entities.

***Data Structure of files***

The files are structured and based on the BeST address model (see section 2) but in a slightly different manner:



Figure 4, ‘Full download data structure’

***File Naming Conventions***

*Region files*

Each region produces 4 separate files:

The name of the files follows the following convention: RegionName+ entity + date + coordinate-system (e.g. L72)

Example for Brussels:

* BrusselsAddress20190319L72.zip
* BrusselsMunicipality20190319L72.zip
* BrusselsPostalinfo20190319L72.zip
* BrusselsStreetname20190319L72.zip

There is one additional file that only Wallonia provides, but not the other regions:

WalloniaPartOfMunicipalityxxxxxxxx.zip

*BOSA files*

BOSA combines the zipfiles from the 3 regions into a single zipfile.

The naming convention for BOSA’s zipfile is

“BestAddress\_” + UseCaseName (abbreviation) + ‘Belgium’ + Date

Full Download: example: BeStAddress\_**FD**Belgium20200319.zip

### S350 - AddressMutationsFileService

\*\*\* The mutations are currently being reworked in order to ensure full harmonisation between the different regions. Spatial transactions will be included in the regular mutations processing. The new mutations process is planned for release in Q2 2022. This section will be updated at that time \*\*\*

***Main functionality***

Mutations are Insert, Update and Delete transactions to an individual address record. Mutations are made available on a daily basis and users can use them a complement to the weekly full download to keep their own address database up-to-date on a daily basis.

Each region produces a single mutations file containing the mutations for all objects (addresses, streetNames, Municipalities, PostalInfo, partOfMunicipalities)

Each regions produces 1 mutation file per day. If no mutations are present, an empty but XSD valid XML file is provided by the region. The provided mutation file contains the mutations for all 5 entities (munitipalities, part of municipalities (optional), streetnames, addresses and postalinfo).

Types of mutations

A mutation is the creation of a new record, the update of an existing record, or the archiving of a record.

* Add

An add takes place when a new record is created.

Example: A new address or a new Streetname is created.

* Delete

When an Identifier is in the Delete table, it means it is no longer used by the region.

It is up to the clients’ interpretation and business logic to determine whether this should result in a physical delete or a logical delete.

Example: An address will not be used anymore.

* Update

This is when an Identifier appears in the Add and the Delete table. An update takes place when non-identifying information to a record is changed.

Example: The status of a Streetname is changed from ‘proposed’ to ‘current’

Structured file of mutations of addresses

The Insert, Update and Archive operations will be constructed on the basis of two data types:

* Table ‘Addobjectversion’ will contain all versions of records that need to be added in the clients’ copy
* Table ‘Deleteobjectversion’ will contain all records that need to be deleted.

In this way, the 3 types of mutations can be constructed:

* **Add**

If a record is shown in the ‘Add’ table only, the record is new and needs to be created for the first time by the client.

* **Delete**

If a record is shown in the ‘delete’ table only, the record can be deleted by the client. In the back-end, this can have the meaning of ‘status = archived’ or of a delete of a record (for instance, because of serious errors in the record).

* **Update**

If a record with the same unique ObjectIdentifier appears both in the ‘delete’ and in the ‘Add’ table, this means an update of the record. The ‘old’ version can be deleted and replaced by the updated version.

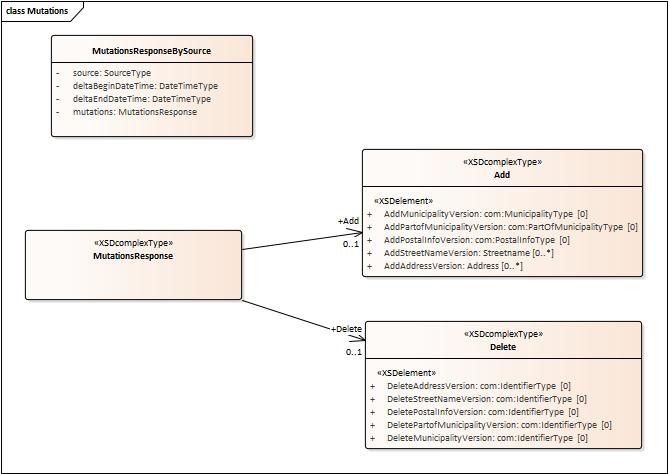


Figure 5, ‘Mutations data structure’

Example:

|  |
| --- |
| **Address: ObjectIdentifier** |
| 12 |
| 34 |
| 45 |
| 89 |

**Table 2, ‘Simplified Delete mutation table’**



**Table 3, ‘Simplified Add mutation table’**

These two tables allow to infer the following mutations:

1. **Insert**

Record 67: The record only occurs in the ‘Add’ table. It will be added to the client’s copy.

1. **Update**

Record 12, 34, 45: These records appear in the ‘Delete’ AND in the ‘Add’ table. So, the older version of the record is replaced by a new version.

1. **Delete**

Record 89: This record only appears in the ‘Delete’ table.

***File naming conventions***

Similar rules apply as for the Full Download. For each service, each Region will deliver 4 zipped files (and Wallonia 5):

*Region files*

The same naming convention as for the full download applies.

*BOSA files*

The naming convention for BOSA’s zipfile is

“BestAddress\_” + UseCaseName (abbreviation) + ‘Belgium’ + Date

Mutations: example: BeStAddress\_**M**Belgium20200312.zip

***Processing of mutations (and spatial transactions***)

Processing sequence

The processing of the mutations and spatial transaction files should take place in a specific sequence:

1. Process all the ‘deletes’ (mutation files) in the following order.
   1. Address
   2. Streetname
   3. PartofMunicipalty
   4. Postalcode
   5. Municipalities
2. Process all the ‘add’ (mutation files)
   1. Municipalities
   2. Postalcode
   3. PartofMunicipalty
   4. Streetname
   5. Address
3. Process the Spatial Transactions:
   1. Substitutions
   2. Mergers
   3. Splits

Errors

Whenever there is an error concerning the MFT services, it would result in a missing file for a certain region.

This could be a one-time event or a recurring error.

In every case, the client will only see that there are files missing.

If this is the case, BOSA will contact the responsible region(s) to solve the problem.

# Open Data

De full download, beschreven hierboven in sectie 4.1, “S349 - FullDownloadService” is ook beschikbaar op het BOSA OpenData platform.

Het bestand wordt wekelijks (maandagochtend) gekopieerd van de MFT naar de opendata website en heeft steeds volgende naam:

Mutaties worden niet gekopieerd naar de open data website, om die te downloaden is een toegang tot de MFT server nodig.

# Webservices - Technical

## Service access parameters

|  |  |
| --- | --- |
| **Endpoint URL (Test & Acceptance)** | [https://fsb.services.int.belgium.be/BeStServices/](https://fsb.services.int.belgium.be/BeStServices/SearchMunicipalityService) |
| **Endpoint URL (Production)** | <https://fsb.services.pr.belgium.be/BeStServices/> |
| **Message exchange pattern(s)** | Synchronous |
| **Message protocol** | SOAP+REST |
| **Transport-level security** | 1-way SSL with digital certificate |
| **Message-level security** | WS-Security X.509 certificate token for Timestamp signing and message body signing |

# Known issues

## Flanders

The use of a version number or version date timestamp is not yet implemented in Flanders. This will be done in the nearby future (summer 2019).

This means that, for all web services, when there is a version filled out in the request (for address, municipality and/or street) this will result in a ‘nothing found’ business error.

Flanders will use a date timestamp as a version.

## Brussels

Mutations file will be empty for now as the Brussels municipalities do not yet enter mutations in the application.

There are known ‘bugs’ in the webservices:

SearchMunicipality

* Sending an empty request results in "back-end error" It should be: "006 - nothing found"
* when we demand municipalities containing 'berg' in language DE, we get a back-end error. It should be: "006 - nothing found"

SearchStreetName and SearchAddress

* Sending an empty request results in Brussels returning "too many results" It should be: "006 - nothing found"

## Wallonia

Address versioning not yet available, planned for Q2 2020

There are known ‘bugs’ in the webservices:

For all webservices:

* "Source: Nothing found" should be "006 - Source: Nothing found"; 27/5: structure of error message still slightly different from design specifications
* If a request is issued with searchType = phonetic (searchType can be ‘contains’, ‘equals', ‘phonetic’), this causes an XSD validation error

# Document Information

## General

|  |  |
| --- | --- |
| Authors(s) : | Gert De Jonge |
| Document name: | BeST WEB Services Userguide |
| Location of the document: |  |
| Number of pages: | 32 |
| Version: | 1.3 |
|  |  |
| Print date: | 2021/11/30 |

## Approbation

|  |  |  |
| --- | --- | --- |
| Nom | Fonction | Organisation |
| Malik Weyns | Service Manager | BOSA |
| François Soumillion | Integration Architect | BOSA |

## Distribution

This document will be distributed to:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Function | Organisation | Objective of distribution |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### 