DG Digitale Transformatie
FOD Beleid en Ondersteuning
DG Transformation digitale
SPF Stratégie et Appui

BeStAddress User Guide
Part 2: WEB services

Date: 11/09/2023
Version: 3.2

Version History

Version	Date	Modified by	Modification
1.0	09/05/19	Gert De Jonge	First version
2.0	23/05/2022	Luc Mertens	Adaptation to new services
2.3	24/10/2022	Bart Hanssens	Rewrote section 4 JSON services and responses Added info on section 6 open data “full download”
2.4	10/11/2022	Eddy Corthouts	Split original BeSt user guide into 3 separate documents
2.5	6/12/2022	Eddy Corthouts	Minor changes
2.6	15/12/2022	Bart Hanssens	Corrections on JSON services
2.7	18/01/2023	Bart Hanssens	Additional search parameters JSON services
2.8	14/02/2023	Bart Hanssens	Additional search parameters JSON services
2.9	22/05/2023	Bart Hanssens	Minor changes
3.0	19/07/2023	Eddy Corthouts	Updated table in section 1.3.2, “new JSON services”
3.1	07/08/2023	Evelyn Barreto	Minor changes
3.2	11/09/2023	Eddy Corthouts	Minor changes

Conventions

Font	use
Italic	accentuation

Contact information

Service Owner	Sebastiaan Taes Sebastiaan.Taes@bosa.fgov.be
Service Desk	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. The document describes the governance principles of the BOSA Service Bus as well.

Table of Contents

List of Figures

Glossary

These are terms specific to this document, general terms known inside the BeSt environment are not added.

Term	Description
Object	Object is a general term, it represents an independent element such as building, parcel, Address, Municipality, StreetName, PostalInfo…
Class	Template or blueprint that is used in modelling techniques to describes an object.
(BeSt) Identifier	Combination of the namespace, objectIdentifier and versionIdentifier which uniquely identify an object.
Entity	Representation of the BeSt object in the real world. The entity is identified by a complete BeStIdentifier. (So every version of an object is an entity) The entity corresponds with 1 record in our dataset.
Component	A sub part of an address or a StreetName. (Objects that are linked to another object) Address has following sub parts: Municipality, StreetName, PostalInfo, PartOfMunicipality A StreetName has 1 sub part: Municipality Municipality, PostalInfo and PartOfMunicipality don’t have sub parts.
Linkable & Linked entity	Entities that have no components but are linked together because there is an address that defines this link. It concerns Municipality, PostalInfo and PartOfMunicipality. Linked entities are always from another type considering 1 address can contain only 1 entity of each type.
History Chain	The history chain allows to retrieve the history of a particular Address (or address component), it consists of a chain of entities that make up the history of the address (or address compontent)
Parameter	(Input) item of the request interface.
Sub-parameter	Parameter that is part of a “combined” parameter.
Enumerated parameter	Parameter with a limited number of allowed values. This includes all Boolean parameters.
(output) Field	Output field in the reply interface of a service.
Prefix mun	For referrals to municipality in the name of a property, it is prefixed with mun
Prefix pom	For referrals to PartOfMunicipality in the name of a property, it is prefixed with pom
Prefix post	For referrals to PostalInfo in the name of a property, it is prefixed it with post
Prefix street	For referrals to StreetName in the name of a property, it is prefixed with street
Predecessor	The BeSt-identifier of the record that will be replaced. This will only be filled in on the ‘Add’ element
Successor	The BeSt-identifier of the record that is the replacemen,t of the current record. This will only be filled in on the ‘Update’ element
MFT	Managed File Transfer
EventType	Type of change . This can be filled on or left blank. Each Region can have their own list of events. The Lists are added in the extensions.
SourceType	Indicates the original regional source of the address (Flanders, Brussels, Wallonia).
Correction	A cosmetic change or a spelling correction

1 Introduction

BeSt stands for “Belgian Streets”. The BeSt services provide address information on a federal level based on the three regional address master data sets from Brussels, Flanders and Wallonia.

Beside the 3 regions, the 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

1.1 Intended Audience

This document is intended for any analyst or developer who wants to make use of the BoSa BeSt Address services.

1.2 Available documentation

The next table provides an overview of the documentation available:

User guide	Purpose
1 BeSt_Userguide_INTRO_and_DATA	Provides an overview of the BeSt application and describes the BeSt data, including the data model and the different data entities with their elements.
2 BeSt_Userguide_WEB_services	Describes the webservices that are available to the consumer to consult the BeSt address data
3 BeSt_Userguide_MFT_services	Describes the Managed File Services that are available to the consumer to obtain a full download file of BeSt addresses or to obtain daily mutations

Purpose of this document

This document describes the Bosa BeSt Address *Webservices.* The webservices are intented for customers who want to consult address information on a “request by request” basis.

For an overview of the application, please refer to the document “1. BeSt_Userguide_INTRO_and_DATA”.

1.3 BeSt Webservices

Two types of services can be distinguished:

- XML services

  These are SOAP services retrieve their information *directly* from the regions by calling region APIs. These services have been operational since 2019 but are planned to be phased out once the new JSON services are live.

- JSON services

  These services retrieve their information from the BoSa BeSt address database and are REST services using JSON formatting.

1.3.1 XML services

The existing XML services retrieve their information *directly* from the regions by calling the region APIs. These existing services use XML formatting and will be referred to as “XML” services.

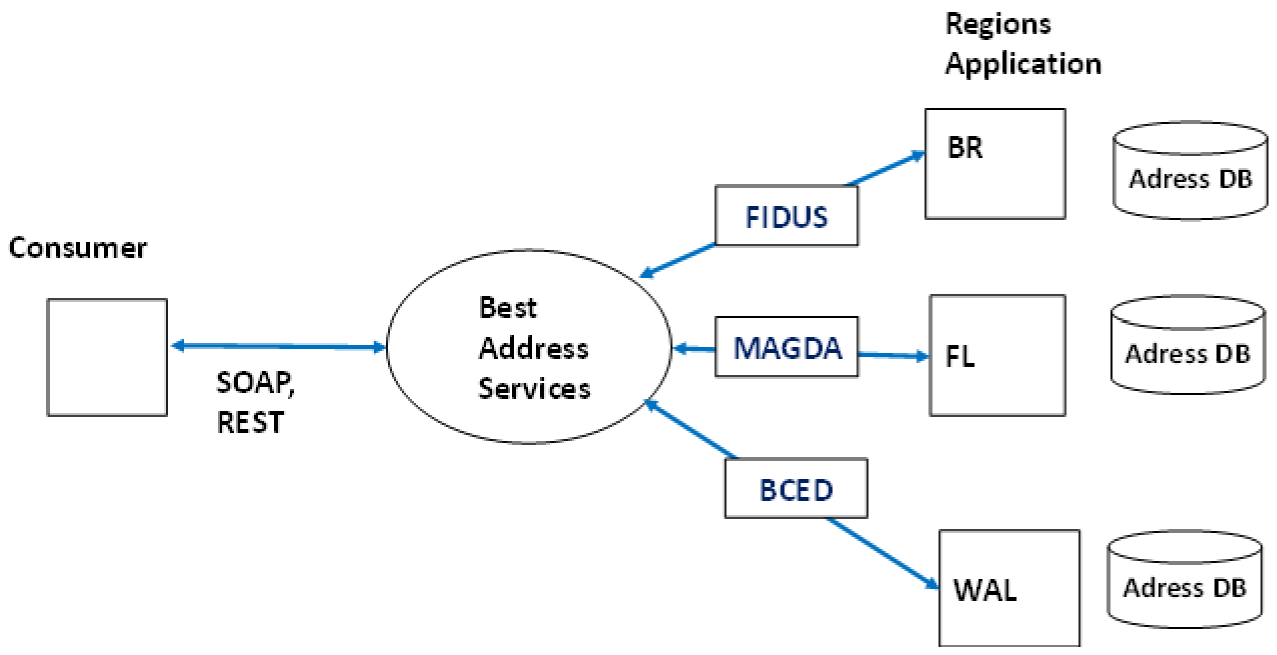


Figure 1, ‘BeSt existing pass-thru XML web services’

The following services are available:

Webservices	**Description**	**Availability Date**
S352 – SearchMunicipality	Search for a municipality based on info from municipality	Q3 2019
S353– SearchStreetname	Search for a street	Q3 2019
S354 – SearchAddress	Search for address info (multiple criteria)	Q3 2019

SOAP and REST

Consumers can invoke these 3 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.

Routing of Webservice requests

For some requests, BOSA does not have to contact *each* of the 3 regions.
The objective of routing BeSt-requests to the right region by BOSA is to reduce the traffic between BOSA and the backends. 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

1.3.2 New JSON services

These services retrieve their information from the new BoSa database and are REST services using JSON formatting. They will be referred to as the “JSON” services.

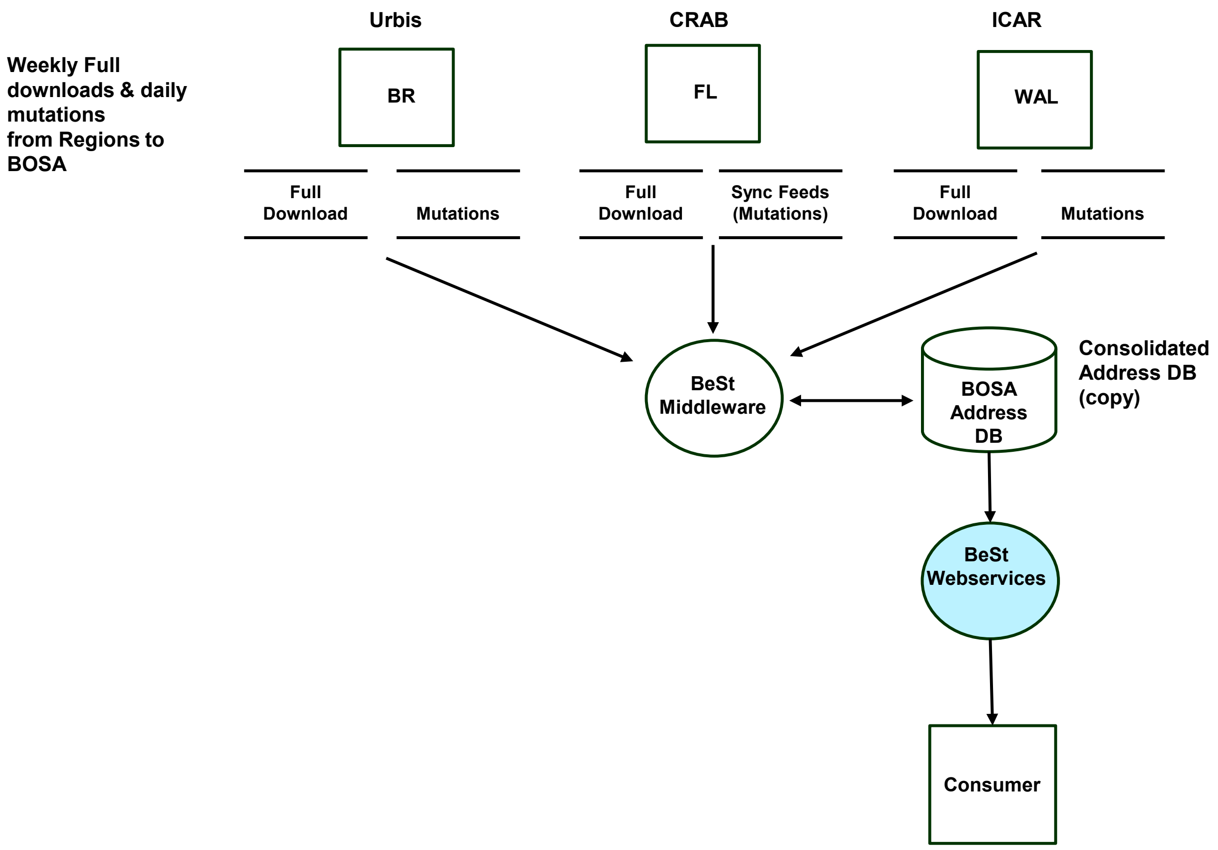


Figure 2, ‘BeSt JSON Services’

The following table provides an overview of the planned JSON Address services:

API	**Description**
/api/best/v1/addresses/belgianAddress/v2	
/Addresses	Search addresses using one or more search parameters (ID, postal code, REFNIS code, postID, municipalityID, gps or Lambert coordinates, status)
/Municipalities	Search municipalities using one or more search parameters (ID, name, postal code, REFNIS code, postID)
/Postalinfos	Search Postalinfos using one or more search parameters (ID, name, postal code)
/StreetName	Search streets using one or more search parameters (ID, name, postal code, REFNIS code, postID, municipalityID, status)
AddressHistory	Search address history
AddressAnomaly	Report an anomaly in BeSt data to region

2 XML Webservices (deprecated)

Below, the following XML webservices[1] 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: XML Web services), 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.
- The used XML version for these services is currently 23.3.1

[1] These are the existing services. Considering they are deprecated, the presentation of the interfaces is not updated.

2.1 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

2.2 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	1 possible statuses: • Proposed • Reserved • Current • Archived	StreetnameStatusValueType
streetNameType	1 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.

2.3 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	1 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	CharStringType	0	1

	the address, if any.			
houseNumber	The house number associated to the address	CharStringType	1	1
officiallyAssigned	Declares if the address is officially 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

2.4 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.

2.5 Known Issues

Brussels region

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"

Walloon region

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

3 JSON Webservices

The REST services (JSON format) provide the user with the possibility to search for all types of BeSt data as well as a service to report anomalies.

- SXXX – Addresses
- SXXX – Municipalities
- SXXX – PostalInfos
- SXXX – Streets
- SXXX - History
- S356 – AnomalyService

The first 4 services allow to search information by either the BeStIdentifier or properties of the entity. The services will return 1 or more BeStIdentifiers together with extra information to identify the wanted entity in the list of returned entities.

The History service will be used to retrieve all details for all entities in the history chain of the entity that matches the BeStIdentifier.

3.1 *Interpreting the JSON interface*

This representation is schematic. For the correct structure, we refer to the Swagger / OpenAPI config file https://public.fedservices.be/api/best/v1/doc/swagger.json

3.2 *Special Parameter types*

Enumerated parameters are parameters with a limited number of allowed values. The values will be mentioned in the description of the parameter.

Booleans are enumerated parameters that must be “true” or ”false”.

Dates follow the international format: “YYYY-MM-DD”.

Datetimes have the following format: “YYYY-MM-DD(T)HH:mm:ss(.ssssss)”.

3.3 *Languages and names*

Languages can have 3 values: ‘nl’, ‘fr’ and ‘de’. Not all languages are always present in the data, only the not-null languages found are returned.

3.4 Common logic

All filled-in input parameters are combined using the “AND” parameter.

Name searches are done “accent-independent”: the search algorithm does not make any distinction between “Liège” or “liege”.

3.4.1 Missing references

Entity types may refer to other entity types (e.g. an Address always references a Municipality, a PostalInfo and a StreetName).

However, on rare occasions this referenced record is not present in the database: BOSA stores the data it receives from the Regions and does report this kind of structural anomalies to the Regions, but these missing references obviously have an impact on the responses.

In case a link to a component (foreign key) is incorrect, the BeStIdentifier will be returned without any additional information about that component.

E.g. if, when searching for an address, the municipality is missing, the address information is returned without the name (or other info) of the municipality.

3.5 Common parameters

3.5.1 Identifiers

A BeSt identifier is composed of 3 parts: a region-specific namespace, an object identifier and a version identifier.

3.5.2 Status

The status field is only used for the entity that is subject of the search operation. E.g. in address this is only the address (not street, municipality….)

This parameter must be one of the values:

- “reserved”
- “proposed”
- “current”
- “retired”

In case this parameter is missing, there will be no check on the status attribute of the entity searched.

Note: in Wallonia, only the status values “current” and “retired are used”.

3.5.3 NIS Code and Postal Code

NisCode

This is the code assigned by the Belgian statistical office StatBel. Currently the municipality objectIdentifier is exactly the same as the NIS code but this may change in the future.

PostCode

This is the code assigned by the Belgian postal office bPost. Currently the postalinfo objectIdentifier is exactly the same as the postal code but this may change in the future.

3.6 Common return codes

As per the Belgif REST guidelines, the following HTTP status codes are used to indicate the result of the request.

200: OK

400: client-side error: incorrect or missing parameter

404: resource not found

500: a server-side error has occurred

3.7 Common GET parameters

PageNo

For performance reasons, the addresses, postal infos and streets results are paginated
(currently no the case for municipalities, since the number of these results will be limited anyway)
The pageNo parameter is the number of the desired result page (first page is 1)

Status

Status of an address, street… e.g. “current”, “retired”.

3.8 S - Addresses

3.8.1 Input

3.8.1.1 HTTP PATH parameter Best-ID

Get one address by ID, return a 404 when the address could not be found.

This parameter is the BeST-ID (composed of namespace, objectidentifier and versionidentifier), and must be URL-encoded. I.e. all '/' and ':' must be percent-encoded.

3.8.1.2 HTTP GET parameters

Search all addresses either within a radius of a specific point, by NIS code of the municipality or a series of other parameters

The results are paginated, with (configurable) 200 results per page.
If no parameters are given, an error will be thrown.

Geo search :

• xLong	{decimal}
• yLat	{decimal}
• crs	(Optional) {String}
• radius	{int}

Search by Post ID / code or NIS code :

• postId	<OR> {String}
• postCode	<OR> {String}
• municipalityId	<OR> {String}
• nisCode	<OR> {String}
• streetId	<OR> {String}
▪ house number	(Optional) {String}
▪ box number	(Optional) {String}

Only return the address with the lowest box number:

• firstOfBuilding	(Optional){boolean}

Embed info about municipalities etc ..

• embed	(Optional){Boolean}

Common:

• status	(Optional) {String}
• pageNo	(Optional) {integer}

3.8.1.3 Parameter description

Geo search: center point and radius

The service returns all addresses inside a circle, using a center point and a radius.

The service supports 3 positioning systems, indicated by the "crs" parameter. GPS/WGS84 is the default if none is given

• lam72	(Lambert 72)
• lam08	(Lambert 2008)
• wgs84	(WGS 84 / GPS)

Depending on the system chosen, an X/Y axis system or a latitude/longitude system will be used for the center.The xLong/yLat parameters define this center point of the circle in all cases:

- xLong {decimal}
- yLat {decimal}

Lam72 and Lam08 are expressed in meters.

Wgs84 are expressed as a decimal grades. This means that values in “grades, minutes and seconds” must be converted to a decimal value in grades: 23° 27′ 30″ becomes 23.45833

The radius parameter is in meters, expressed as a decimal and must be > 0. A maximal radius can be set by BOSA for performance reasons (currently 1000 meters)

Embed

This parameter indicates if the details of municipalities, streets are to be included (this avoids extra GET request for obtaining the names and other info about these referenced objects).The default value is “true”.

- True: all details from the referenced entities are in the response
- False: only the identifier of the referenced entities are in the response

municipalityId, postId, streetId

This parameter is the BeST-ID (composed of namespace, objectidentifier and versionidentifier)

firstOfBuilding

This parameter indicates if, for a building / house number with multiple apartments / box numbers, only the address with the lowest box number is to be included (no default = no filtering):

- True: only include addresses with the lowest box number

3.8.2 Output

3.8.2.1 JSON

BeStResponse

- errorOrWarning
  - errorOrWarningCode {int}
  - errorOrWarningDesc {String}
- address [List]
  - id {String}
  - bestidentifier
    - nameSpace {String}
    - objectIdentifier {String}
    - versionIdentifier {String}
  - hasStreetName
    - id {String}
    - name
      - nl {String}
      - fr <AND/OR> {String}
      - de <AND/OR> {String}
    - homonymAddition (Optional) {String}
    - streetNameType {string}
  - hasMunicipality
    - id {String}

▪ nisCode	{String}
▪ name	
♦ nl	{String}
♦ fr	<AND/OR> {String}
♦ de	<AND/OR> {String}
- hasPostalInfo	
▪ id	{String}
▪ postCode	{String}
▪ name	
♦ nl	{String}
♦ fr	<AND/OR> {String}
♦ de	<AND/OR> {String}
▪ hasPartOfMunicipality	(Optional)
▪ id	{String}
▪ name	
♦ nl	{String}
♦ fr	<AND/OR> {String}
♦ de	<AND/OR> {String}
- houseNumber	{String}
- boxNumber	(Optional) {String}
- addressSortField	(Optional) {String}
- status	{String}
- validFrom	{datetime}
- validTo	(Optional) {datetime}
- addressPosition	(Optional)
▪ lambert72	
♦ x	{decimal}
♦ y	{decimal}
▪ lambert08	
♦ x	{decimal}
♦ y	{decimal}
▪ wgs84	
♦ lat	{decimal}
♦ long	{decimal}
▪ positionGeometryMethod	(Optional) {String}
▪ positionSpecification	(Optional) {String}
• isOfficiallyAssigned	(Optional) {String}

3.8.2.2 Description

PositionGeometryMethod

The way how this point was defined

PositionSpecification

The object on which the point was defined (parcel, building …)

AddressPosition

Note that currently there is a know issue for addresses in Wallonia: about 100.000 – 150.000 addresses in the Walloon Region have no coordinates (0,0)

3.9 S – Municipalities

3.9.1 Input

3.9.1.1 HTTP PATH parameter Best-ID

Get one municipality by ID, return a 404 when the municipality could not be found.

This parameter is the BeST-ID (composed of namespace, objectidentifier and versionidentifier), and must be URL-encoded. I.e. all '/' and ':' must be percent-encoded.

3.9.1.2 HTTP GET parameters

Search all municipalities by NIS code or postal code.

When no parameters are given, a list of all the municipalities is returned.

• name	(Optional) {String}
• nisCode	(Optional) {String}
• postCode	<OR>(Optional) {String}

3.9.1.3 Parameter description

Name

Substring search in NL, FR and DE municipality names. Case-insensitive and accent-insensitive.

NisCode

Nation-wide unique municipality NIS/INS code, assigned by the national statistical office Statbel.

PostCode

Postal code, assigned by the Belgian post office bPost.
Note that a postal code is not guaranteed to be assigned to only 1 municipality (nor vice versa), even when this often the case.

3.9.2 Output

3.9.2.1 JSON

BeStResponse

- errorOrWarning
  - errorOrWarningCode {int}
  - errorOrWarningDesc {String}
- municipality (Optional) [List]
  - id {String}
  - bestidentifier
    - nameSpace {String}
    - objectIdentifier {String}
    - versionIdentifier {String}
  - nisCode {String}
  - name
    - nl {String}

▪ fr	<AND/OR> {String}
▪ de	<AND/OR> {String}
- status	(Optional) {datetime}
- validFrom	(Optional) {datetime}
- validTo	(Optional) {datetime}

3.10 S – PostalInfo

3.10.1 Input

3.10.1.1 HTTP PATH parameter Best-ID

Get one postal info by ID, return a 404 when the postal info could not be found.

This parameter is the BeST-ID (composed of namespace, objectidentifier and versionidentifier), and must be URL-encoded. I.e. all '/' and ':' must be percent-encoded.

3.10.1.2 HTTP GET parameters

Search all streets by postal code or name.
If no parameters are given, a list of all postal infos is returned.

• postCode	(Optional) {String}
• name	(Optional) {String}

3.10.1.3 Parameter description

Name

Substring search in NL, FR and DE municipality names. Case-insensitive and accent-insensitive.

Note that there is a known issue: it does not work for the Walloon Region since that Region does not provide the names in the source files.

PostCode

Postal code, assigned by the Belgian post office bPost.
Note that a postal code is not guaranteed to be assigned to only 1 municipality (nor vice versa), even when this often is the case.

3.10.2 Output

3.10.2.1 JSON

BeStResponse

• errorOrWarning	
- errorOrWarningCode	{int}
- errorOrWarningDesc	{String}
• postalInfo	(Optional) [List]
- id	{String}
- bestidentifier	
▪ nameSpace	{String}
▪ objectIdentifier	{String}
▪ versionIdentifier	{String}
- postCode	{String}
- name	
▪ nl	(Optional) {String}
▪ fr	<AND/OR> {String}
▪ de	<AND/OR> {String}
- status	(Optional) {datetime}
- validFrom	(Optional) {datetime}
- validTo	(Optional) {datetime}

3.11 S - Streets

3.11.1 Input

3.11.1.1 HTTP PATH parameter Best-ID

Get one street by ID, return a 404 when the street could not be found.

This parameter is the BeST-ID (composed of namespace, objectidentifier and versionidentifier), and must be URL-encoded. I.e. all '/' and ':' must be percent-encoded.

3.11.1.2 HTTP GET parameters

Search all streets by municipality code, NIS code, postal code, postal ID or name.

Note that searching by postal code / postal ID may produce surprising results: postal codes are assigned on the address level, not street / municipality level. Which means that streets without any address (like parcs, country/forest roads, …) will not be found when searching for postal codes, so use the municipality ID / nisCode instead.

• postId	(Optional) {String}
• postCode	<OR> {String}
• municipalityId	<OR> {String}
• nisCode	<OR>{String}
• name	(Optional){String}

3.11.1.3 Parameter description

Name

Substring search in NL, FR and DE municipality names. Case-insensitive and accent-insensitive.

municipalityId, postId
This parameter is the BeST-ID (composed of namespace, objectidentifier and versionidentifier)

NisCode

Nation-wide unique municipality NIS/INS code, assigned by the national statistical office Statbel.

PostCode

Postal code, assigned by the Belgian post office bPost.
Note that a postal code is not guaranteed to be assigned to only 1 municipality (nor vice versa), even when this often the case.

3.11.2 Output

3.11.2.1 JSON

BeStResponse

• errorOrWarning	
- errorOrWarningCode	{int}
- errorOrWarningDesc	{string}
• streetName	(Optional) [List]
- id	{String}
- bestidentifier	

▪ nameSpace	{String}
▪ objectIdentifier	{String}
▪ versionIdentifier	{String}
- name	
▪ nl	{String}
▪ fr	<AND/OR> {String}
▪ de	<AND/OR> {String}
- homonymAddition	(Optional) {String}
- streetNameType	(Optional) {String}
- status	{String}
- validFrom	(Optional) {datetime}
- validTo	(Optional) {datetime}
- isAssignedByMunicipality	(Optional)
▪ id	{String}
▪ name	
♦ nl	{String}
♦ fr	{String}
♦ de	{String}
▪ nisCode	{String}

3.11.2.2 Description

StreetNameType

Type of the street, e.g. street, hamlet…

isAssignedByMunicipality

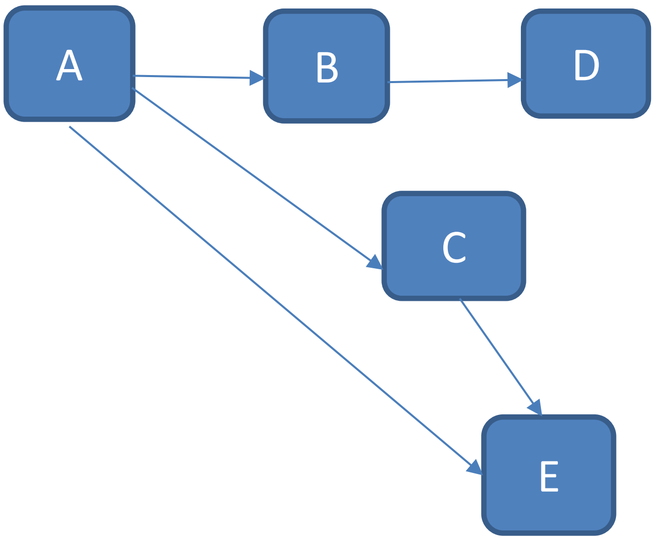
A street can belong to multiple municipalities. The isAssignedByMunicipality element specifies the Municipality where the street was found.

3.12 S – History (not yet implemented)

3.12.1 History Chain example

Data
In this example, we have a history chain for a street. The BeStIdentifiers are represented to 1 character.

On 1/1/1990 a StreetName A is created (current).
On 1/1/2020: A is split into A+B (A now has successor B but still exist)
On 1/6/2021: A (remaining part) is again split into A + C (A now has a second successor C and still exist)
On 1/1/2022: rename of street A into E (A gets an end date and an additional successor)
On 1/1/2022: rename of street B into D (B gets an end date and a successor)
½1/2/2022: Merge from street C and E into E (E remains existing)



Available Data:

StreetName

StreetnameId	StreetName	validfrom	validto	status
A	Name A	1/01/1990	1/01/2022	Current
B	Name B	1/01/2020	1/01/2022	Current
C	Name C	1/06/2021	1/02/2022	Current
D	Name B2	1/01/2022		Current
E	Name A2	1/01/2022		Current

StreetNameHistory:

Predecessor	Successor	Eventdate
A	B	1/01/2020
A	C	1/06/2021
A	E	1/01/2022
B	D	1/01/2022
C	E	1/02/2022

Request A without history

When we request StreetName A without History, what do we get?

There is no History chain.

Streetname chain is limited to 1 record[2]:

StreetnameId	Name	validfrom	validto
A	Name A	1/01/1990	1/01/2022

Request A with history

When we request StreetName A with History, what do we get?

History chain:

predecessor	Successor	EventDate
A	B	1/01/2020
A	C	1/06/2021
A	E	1/01/2022
B	D	1/01/2022
C	E	1/02/2022

Streetname chain:

StreetnameId	Name	validfrom	validto
A	Name A	1/01/1990	1/01/2022
B	Name B	1/01/2020	1/01/2022
C	Name C	1/06/2021	1/02/2022
D	Name B2	1/01/2022	
E	Name A2	1/01/2022	

Request B

When we ask for StreetName B (with history), what do we get?
History chain

predecessor	Successor	EventDate
A	B	1/01/2020
B	D	1/01/2022

StreetName chain

StreetnameId	Name	validfrom	validto
B	Name B	1/01/2020	1/01/2022
A	Name A	1/01/1900	1/01/2022
D	Name B2	1/01/2022	

As one can see, the data in both chains (history and entity chain) is limited to those entities that are in the history of the requested entity. StreetName C and E are only related to A and not to B. A is a predecessor from B while C and E are successors of A. This makes that they are not related to B at all. Therefore, they are not added to this response.

[2] The examples only show the fields that are needed to illustrate the response, all properties will be displayed in the real service..

3.13 S3–6 – AnomalyService (not yet implemented)

The anomaly service allows the federal partners to report anomalies concerning the BeSt data.
The service provides a uniform way to notify the regions so that they can react on these notifications.

From the point of view of BoSa, it is considered a “Fire and Forget” service: BoSa creates the anomaly report with the data provided by the submittor and sends it to the destination. At no time in this process, BoSa stores any anomaly report data, does any logging or makes any correction to the data in the anomaly report.

3.13.1 Structure and functioning of the anomaly service

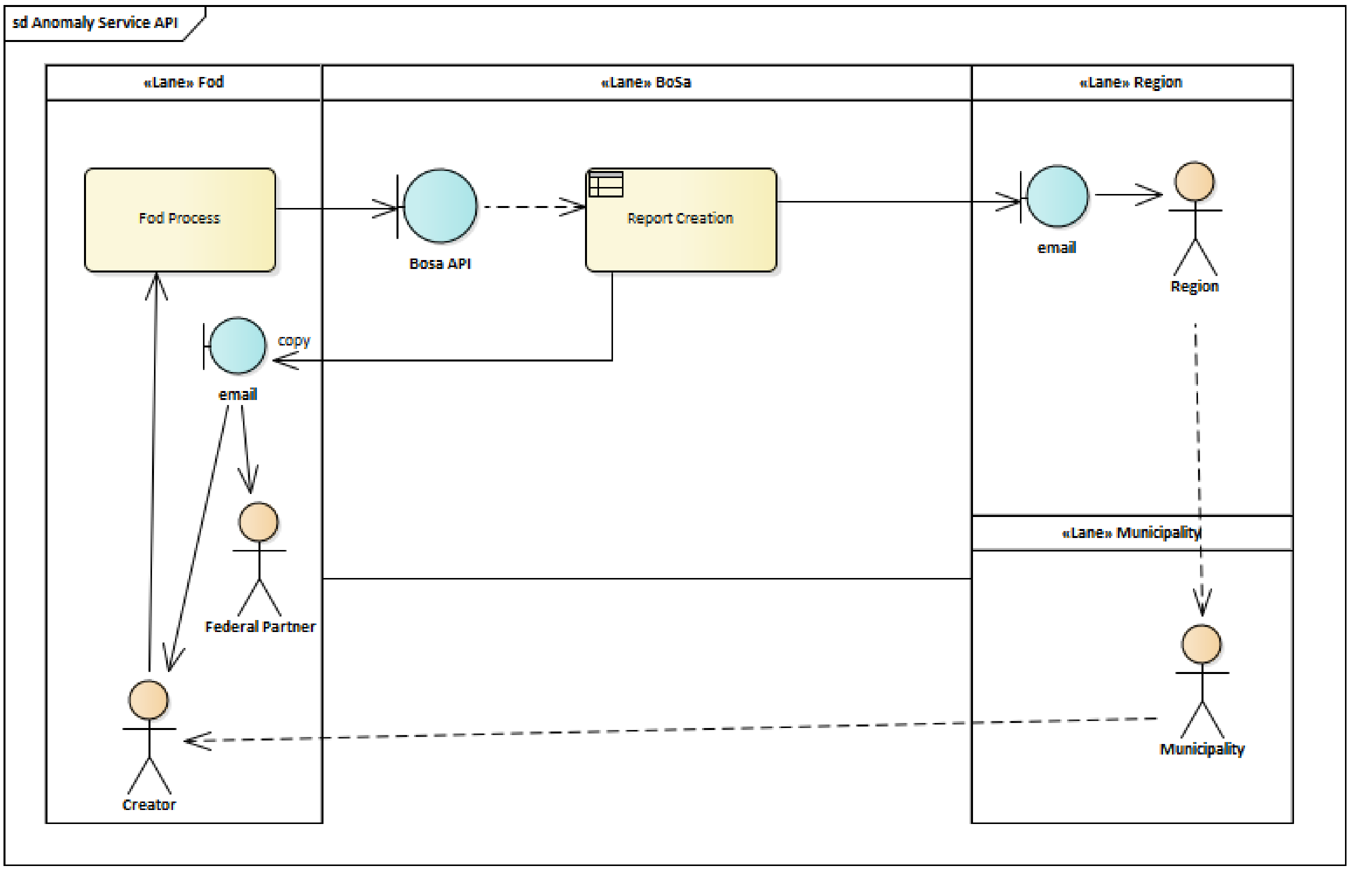


Figure 3, ‘Anomaly service operation’

The federal partner accesses the service by connecting to the API.

Once the data is submitted, the service creates the anomaly report and mails it to the destination (region). A copy is sent to the submitter using the contact email(s) that are provided in the request.

It is the task of the regions to send the anomaly report to the concerned municipality so that they can react on the anomaly report. Using the contact information (including all emails the report is sent to), the municipality can inform the submitter(s) of the result.

The service will:

- Allow consumers to directly access the service from within their own applications.
- Mail the report to the regions.
- Sent a copy of the anomaly report to the submitter[3].

3.13.2 Input

The user can choose to request a correction for OR an Address AND/OR a streetname AND/OR postalInfo AND/OR municipality AND/OR partOfMunicipality in the same request.

3.13.2.1 JSON Request

BeStRequest

- anomalyType {String}
- anomalyDescription {String}
- contact
  - email {String}
  - name (Optional) {String}
  - telephone (Optional) {String}
  - reference (Optional) {String}
- destination {String}
- subject (Optional)
  - address
    - bestidentifier
      - BeStIdentifier (Optional) {String}
      - objectIdentifier {String}
      - versionIdentifier {String}
    - houseNumber (Optional)
      - current {String}
      - correct <AND/OR> {String}
    - boxNumber (Optional)
      - current {String}
      - correct <AND/OR> {String}
    - addressSortField (Optional)
      - current {String}
      - correct <AND/OR> {String}
    - status (Optional)
      - current {String}
      - correct <AND/OR> {String}
    - validFrom (Optional)
      - current {date}
      - correct <AND/OR> {date}
    - validTo (Optional)
      - current {date}
      - correct <AND/OR> {date}
    - isOfficiallyAssigned (Optional)
      - current {String}
      - correct <AND/OR> {String}
    - addressPosition (Optional)
      - lam72 (Optional)
        - current
          - x {decimal}
          - y {decimal}
        - correct <AND/OR>
          - x {decimal}

[3] Besides sending a copy of the anomaly report to the submitter, a copy can also be sent to the federal partner if they automatically adds a generic email to the email field before calling the API.

▪ y	{decimal}
♦ lam08	(Optional)
➢ current	
▪ x	{decimal}
▪ y	{decimal}
➢ correct	<AND/OR>
▪ x	{decimal}
▪ y	{decimal}
♦ wgs84	(Optional)
➢ current	
▪ lat	{decimal}
▪ long	{decimal}
➢ correct	<AND/OR>
▪ lat	{decimal}
▪ long	{decimal}
♦ positionGeometryMethod	(Optional)
➢ current	{String}
➢ correct	<AND/OR> {String}
♦ positionSpecification	(Optional)
➢ current	{String}
➢ correct	<AND/OR> {String}
▪ hasStreetName	(Optional)
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ objectIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ versionIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ hasMunicipality	(Optional)
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ objectIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ versionIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ hasPostalInfo	(Optional)
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ objectIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ versionIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ hasPartOfMunicipality	(Optional)
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ objectIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
▪ versionIdentifier	
♦ current	{String}
♦ correct	<AND/OR> {String}
- streetName	
▪ BeStIdentifier	(Optional)

        - namespace {String}
        - objectIdentifier {String}
        - versionIdentifier {String}
    - name (Optional)
        - nl
            - current {String}
            - correct <AND/OR> {String}
        - fr <AND/OR>
            - current {String}
            - correct <AND/OR> {String}
        - de <AND/OR>
            - current {String}
            - correct <AND/OR> {String}
    - homonymAddition (Optional)
        - current {String}
        - correct <AND/OR> {String}
    - streetNameType (Optional)
        - current {String}
        - correct {String}
    - status (Optional)
        - current {String}
        - correct <AND/OR> {String}
    - validFrom (Optional)
        - current {date}
        - correct <AND/OR> {date}
    - validTo (Optional)
        - current {date}
        - correct <AND/OR> {date}
        - isAssignedByMunicipality (Optional)
            - current {String}
            - correct <AND/OR> {String}
        - objectIdentifier
            - current {String}
            - correct <AND/OR> {String}
        - versionIdentifier
            - current {String}
            - correct <AND/OR> {String}
- municipality
    - BeStIdentifier (Optional)
        - namespace {String}
        - objectIdentifier {String}
        - versionIdentifier {String}
    - name (Optional)
        - nl
            - current {String}
            - correct <AND/OR> {String}
        - fr <AND/OR>
            - current {String}
            - correct <AND/OR> {String}
        - de <AND/OR>
            - current {String}
            - correct <AND/OR> {String}
    - nisCode (Optional)
        - current {String}
        - correct <AND/OR> {String}
    - status (Optional)
        - current {String}
        - correct <AND/OR> {String}
    - validFrom (Optional)

```
            ♦  current                 {date}
            ♦  correct                 <AND/OR> {date}
        ▪  validTo                     (Optional)
            ♦  current                 {date}
            ♦  correct                 <AND/OR> {date}
    -   postalInfo             <OR>
        ▪  BeStIdentifier              (Optional)
            ♦  namespace               {String}
            ♦  objectIdentifier        {String}
            ♦  versionIdentifier       {String}
        ▪  name                        (Optional)
            ♦  nl
                ➢  current             {String}
                ➢  correct             <AND/OR> {String}
            ♦  fr                      <AND/OR>
                ➢  current             {String}
                ➢  correct             <AND/OR> {String}
            ♦  de                      <AND/OR>
                ➢  current             {String}
                ➢  correct             <AND/OR> {String}
            ♦  current                 {String}
            ♦  correct                 <AND/OR> {String}
        ▪  status                      (Optional)
            ♦  current                 {String}
            ♦  correct                 <AND/OR> {String}
        ▪  validFrom                   (Optional)
            ♦  current                 {date}
            ♦  correct                 <AND/OR> {date}
        ▪  validTo                     (Optional)
            ♦  current                 {date}
            ♦  correct                 <AND/OR> {date}
    -   partOfMunicipality
        ▪  BeStIdentifier              (Optional)
            ♦  namespace               {String}
            ♦  objectIdentifier        {String}
            ♦  versionIdentifier       {String}
        ▪  name                        (Optional)
            ♦  nl
                ➢  current             {String}
                ➢  correct             <AND/OR> {String}
            ♦  fr                      <AND/OR>
                ➢  current             {String}
                ➢  correct             <AND/OR> {String}
            ♦  de                      <AND/OR>
                ➢  current             {String}
                ➢  correct             <AND/OR> {String}
        ▪  status                      (Optional)
            ♦  current                 {String}
            ♦  correct                 <AND/OR> {String}
        ▪  validFrom                   (Optional)
            ♦  current                 {date}
            ♦  correct                 <AND/OR> {date}
        ▪  validTo                     (Optional)
            ♦  current                 {date}
            ♦  correct                 <AND/OR> {date}
```

3.13.2.2 Input parameter description

In this section, we only discuss the JSON API and new input parameters or parameters with a (slightly) different functionality compared with previous explanations in this chapter.

Types of anomalies.
The user can choose from 4 types of anomalies:

- Redundant
  - The submitter thinks an element that is no longer needed (Redundant) and should be put to in-active (add end date)
  - The submitter thinks this element should be retired.
- Missing
  - The submitter thinks an element is missing.
- Correction
  - The submitter thinks an element needs correction.
- Question
  - The submitter has a question because something is not clear to him.

Anomaly Description
Description of the anomaly. This text field is the main field in the whole anomaly service and must be filled

Destination: Region
With the destination parameter, the user defines who will get the anomaly report.

- Missing or invalid value
  The destination is unknown so an error code will be generated. No report is made.
- "F"
  Destination is Flanders
- "W"
  Destination is Wallonia
- "B"
  Destination is Brussels

Contact
Contact collects all information the receiver needs to get in contact with the submitter of the report.

Email
This mandatory fierld must contain 1 or more valid email addresses, the receiver can use to contact the submitter(s).

This field is also used to send a copy of the anomaly report to the submitter.

Multiple valid email addresses separated by a ";" are allowed.

If the partner that uses this services wants an additional copy (e.g. for central management in their department), they always can add a generic email to the list of emails.

Name
The name(s) of the person(s) to contact. This is a simple text field, any value can be entered.

This field is optional as names can usually be deducted from the email addresses.

Telephone
1 or more telephone numbers can be added where the receiver can contact the submitter(s). This is a simple text field, any value can be entered.

Reference
A reference that the submitter uses to identify the case / file where this anomaly happened so he can trace it back to his work within his organization. This is a simple text field, any value can be entered.

Subject
BeSt Element that is subject to this anomaly. All fields of 1 single element that are known in BeSt are available.

The submitter can choose to limit the data provided to the BeStIdentifier or fill in all fields that are useful to describe the anomaly.

When a subject is given, the system expect that the BeStIdentifier is filled in unless the AnomalyType is “Missing”.

Subject can be an Address, a StreetName, a Municipality, a PostalInfo or a partOfMunicipality

In case of a correction, the user can fill in the proposed correction in the field “correction” that is t available for each data field of the element.

Note
It is possible to identify the subject using only it’s BeStIdentifier. The receiver has enough information to identify the element and as such, has all current info.

In case a subject is added to the API data, the BeStIdentifier is mandatory, and we expect that all 3 parts are filled: namespace + ObjectIdentifier + versionIdentifier. This requirement (mandatory BeStIdentifier) is not valid for AnomalyType “Missing”.

3.13.3 Output

3.13.3.1 JSON

BeStResponse
- errorOrWarning
  - errorOrWarningCode {int}
  - errorOrWarningDesc {String}

3.13.3.2 Output structure description

Codes
The system returns one of following response codes.

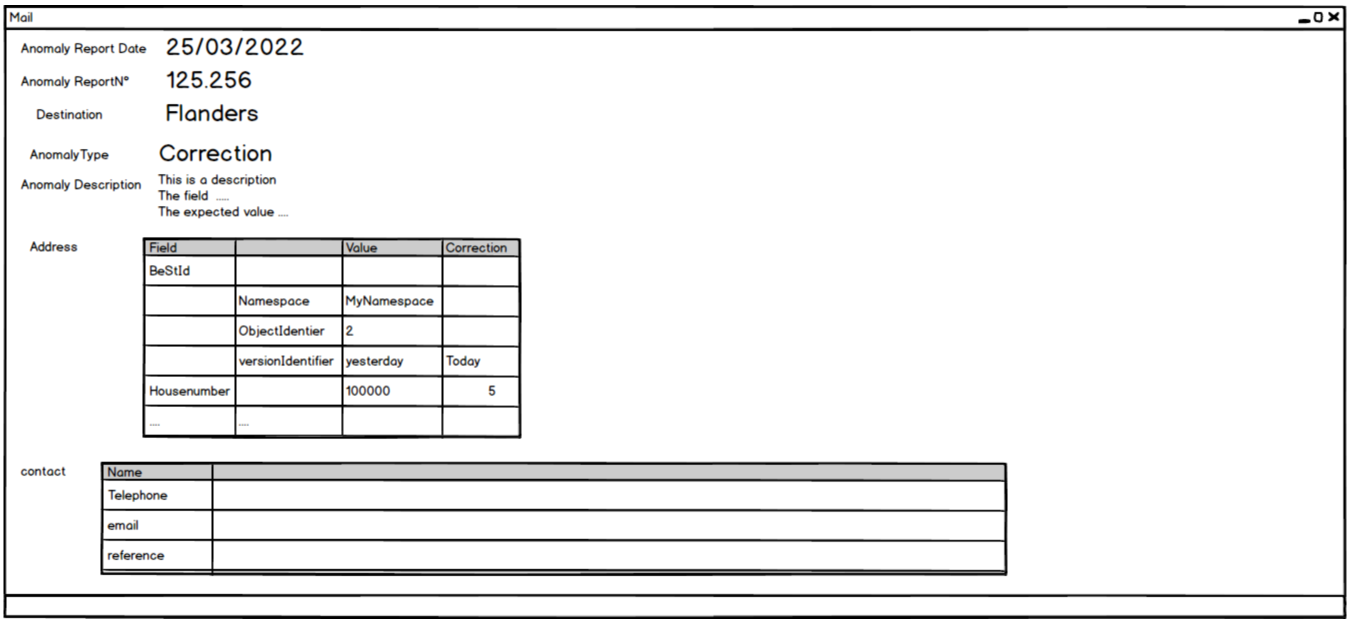
Code	Description
200	Anomaly report is sent to destination and all emails in the contact data.
450	No destination given (region not filled in).
451	No contact email is added, or the added email is incorrectly formatted. In case there are multiple email addresses, they should be correctly separated by a “;”.
452	Anomaly Type is missing or empty
453	Anomaly Description is missing or empty.
454	Subject without a full BeStIdentifier while AnomalyType is different from “Missing”.
Any other code	Technical error

3.13.3.3 Anomaly report

The anomaly report will be sent as email to the destination region specified in the request.

Every report gets a report date and a report number (simple sequence number) to be able to refer it later. These elements are automatically added to the anomaly report.

This anomaly report is formatted in the following way:


Mail

Anomaly Report Date 25/03/2022

Anomaly ReportN° 125.256

Destination Flanders

AnomalyType Correction

Anomaly Description This is a description
The field .....
The expected value ....

Address

Field		Value	Correction
BeStId			
	Namespace	MyNamespace	
	ObjectIdentier	2	
	versionIdentifier	yesterday	Today
Housenumber		100000	5
....	....		

contact

Name	
Telephone	
email	
reference	

Figure 4, ' Mail content'

4 Service Access parameters

4.1 XML Services

Endpoint URL (Test & Acceptance)	https://fsb.services.int.belgium.be/BeStServices/
Endpoint URL (Production)	https://fsb.services.pr.belgium.be/BeStServices/
Message exchange pattern(s)	Synchronous
Message protocol	SOAP
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

4.2 JSON Services

Endpoint URL (Test & Acceptance)	https://public.int.fedservices.be/api/BeSt/v1/belgianAddress/v2/
Endpoint URL (Production)	https://public.fedservices.be/api/BeSt/v1/belgianAddress/v2/
Message exchange pattern(s)	Synchronous
Message protocol	REST
Transport-level security	TLS
Message-level security	OAuth

Document Information

General

Authors(s) :	Luc Mertens, Eddy Corthouts, Bart Hanssens
Document name:	BeSt Userguide WEB Services
Number of pages:	46
Version:	2.6
Print date:	2023/09/14

Approbation

Nom	Fonction	Organisation
Johan Mertens	Service Manager	BOSA
François Soumillion	Integration Architect	BOSA

Distribution

This document will be distributed to:

Name	Function	Organisation	Objective of distribution