

Object Configuration (Draft)

Object Configuration illustrates how the engineering objects are used for supporting the functionalities commonly shared among more than one RI such as Identification and Citation, Curation, [Cataloguing Configurations](#), Processing, and Provenance.

Cataloguing Configurations

There are seven cataloguing processes (D8.3) which need to be supported using EV objects. These basic EV configurations are mapped to these processes in [Table 01](#). As shown in the table, the combination of these basic configurations allows supporting different cataloguing processes.

Table 1. Basic Cataloguing Processes	
Processes (User Stories/Requirements) (Stage 1 Description)	Configuration Name
1. Store an asset (e.g. dataset) with metadata sufficient for cataloging purposes;	Store Asset Configuration
2. Discover an asset using the metadata – the richer the metadata and the more elaborate the query the greater the precision in discovering the required asset(s);	Discover Asset Configuration
3. Download an asset based on metadata	Download Asset Configuration
4. Select/project an asset in situ location accessed via metadata	Select Asset Configuration
5. Move (selected parts of) asset to location accessed via metadata	Move Asset Configuration
6. Compose (selected parts of) asset into workflow accessed via metadata	Compose Asset Configuration
7. Update an asset and/or metadata	Update Asset Configuration

This section define set of seven independent cataloguing services that support each process on [Table 01](#). This clearly separates the functionalities vertically and shows how the BEOs interact with each other. The structure of the following configurations is based on the [Container Structure](#).

In addition to the EV BEOs all the configurations require a management component object and a catalogue system object. These engineering objects together with the collection of configurations conform the complete catalogue service.

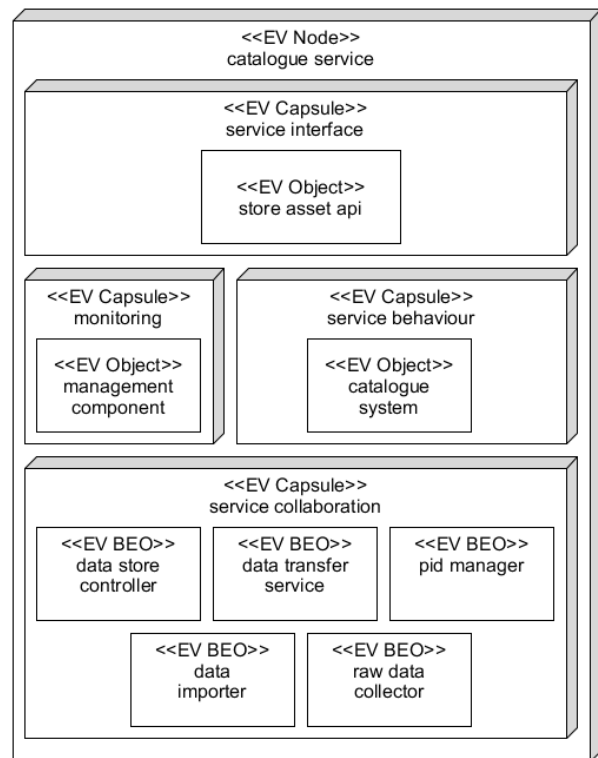
Store Asset Configuration

The configuration for storing a new asset depends on six BEOs: [catalogue service](#), [data transfer service](#), [raw data collector](#), [data importer](#), [pid manager](#), and [data store controller](#).

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for three possible service variations: collect from sensor, import from another repository, and assign global unique pid.

Name:	Store Asset
Process:	Store an asset (e.g. dataset) with metadata sufficient for cataloging purposes
Resources	<ul style="list-style-type: none">assetmetadata
Outputs	<ul style="list-style-type: none">asset [registered]metadata [registered]asset pid [registered]
Options	<ul style="list-style-type: none">collect data from sensorsimport data from external sourceassign global unique pid



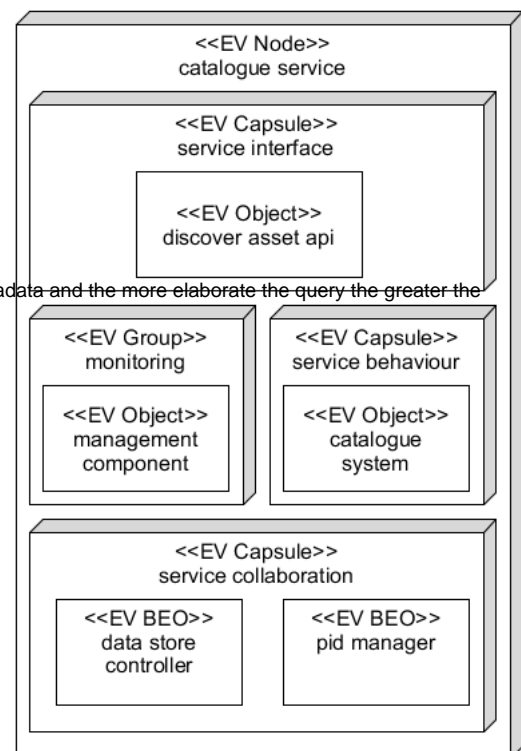
Discover Asset Configuration

The configuration for discovering an asset depends on three BEOs: [catalogue service](#), [pid manager](#), and [data store controller](#).

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for supporting the discovery process.

Name:	Discover Asset
Process:	Discover an asset using the metadata – the richer the metadata and the more elaborate the query the greater the precision in discovering the required asset(s)
Resources:	<ul style="list-style-type: none"> query parameters (metadata)
Outputs:	<ul style="list-style-type: none"> metadata <ul style="list-style-type: none"> asset pid asset reference (publishable) asset description
Options:	<ul style="list-style-type: none"> faceted discovery (filtering of results)
Correspondences	Technology Viewpoint: Catalogue Discovery Object



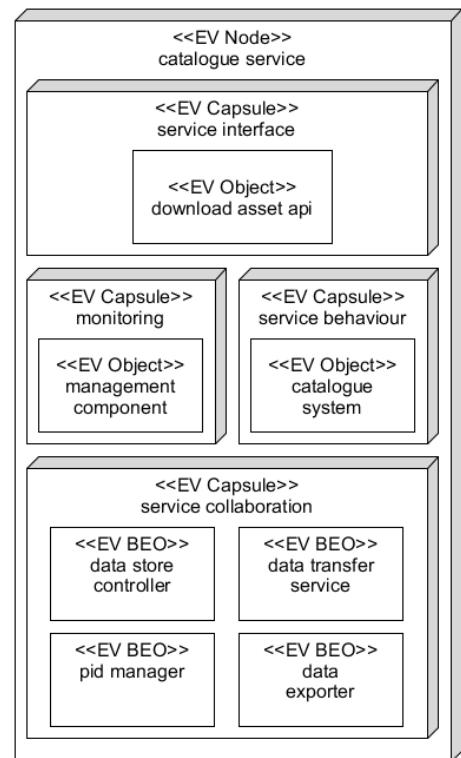
Download Asset Configuration

The configuration for storing a new asset depends on five BEOs: [catalogue service](#), [data transfer service](#), [data exporter](#), [pid manager](#), and [data store controller](#)

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for supporting the discovery process.

Name:	Download Asset
Process:	Download an asset based on metadata
Resources	<ul style="list-style-type: none"> asset pid (precondition: discovery) filtering parameters (metadata, optional)
Outputs	<ul style="list-style-type: none"> asset (copy) metadata <ul style="list-style-type: none"> asset pid asset reference (publishable) asset description
Options	<ul style="list-style-type: none"> download complete download partial



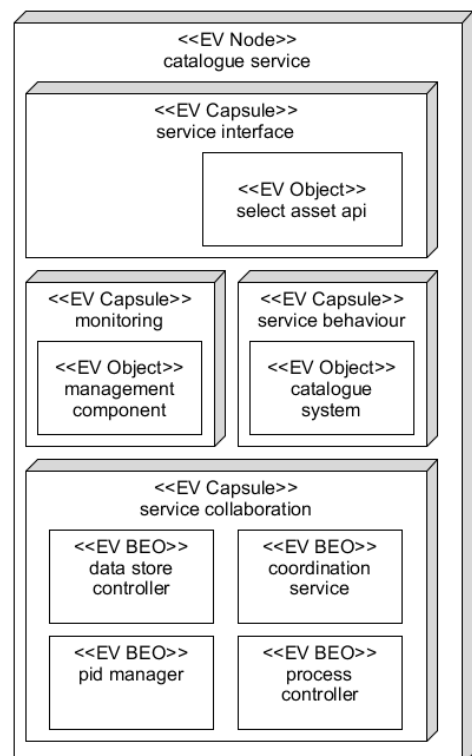
Select Asset Configuration

The configuration for selecting an asset depends on five BEOs: [catalogue service](#), [coordination service](#), [process controller](#), [pid manager](#), and [data store controller](#)

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for supporting the discovery process.

Name:	Select Asset
Process:	Select/project an asset in situ location accessed via metadata
Resources	<ul style="list-style-type: none"> asset pid (precondition: discovery) filtering parameters (metadata, optional)
Outputs	<ul style="list-style-type: none"> asset (processed view) metadata <ul style="list-style-type: none"> asset pid asset reference (publishable) asset description asset view details
Options	<ul style="list-style-type: none"> select project



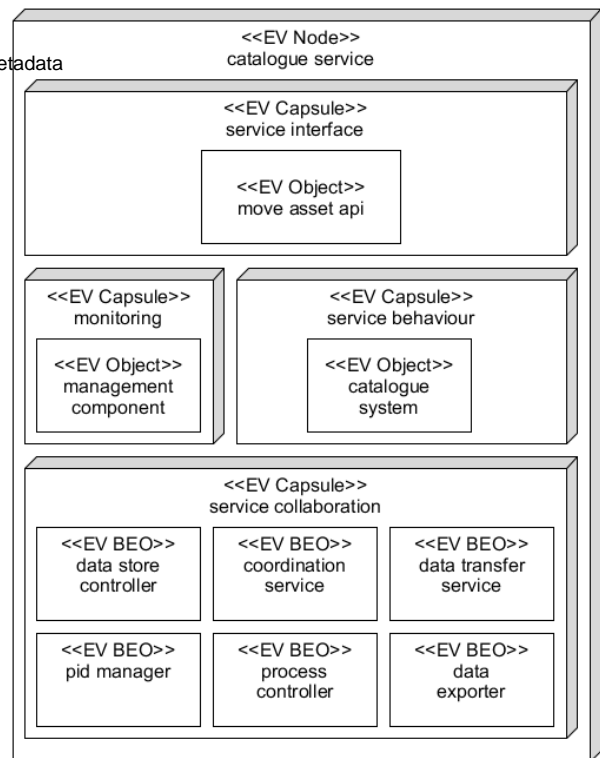
Move Asset Configuration

The configuration for moving an asset depends on seven BEOs: [catalogue service](#), [data transfer service](#), [data exporter](#), [pid manager](#), [data store controller](#), [coordination service](#), and [process controller](#),

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for supporting the move asset process.

Name:	Move Asset
Process:	Move (selected parts of) asset to location accessed via metadata
Resources	<ul style="list-style-type: none"> asset pid (precondition: discovery) filtering parameters (metadata, optional)
Outputs	<ul style="list-style-type: none"> asset (partial or completely relocated) metadata <ul style="list-style-type: none"> asset pid asset reference (publishable) asset description
Options	<ul style="list-style-type: none"> move complete (delete original) move partial (delete partial)



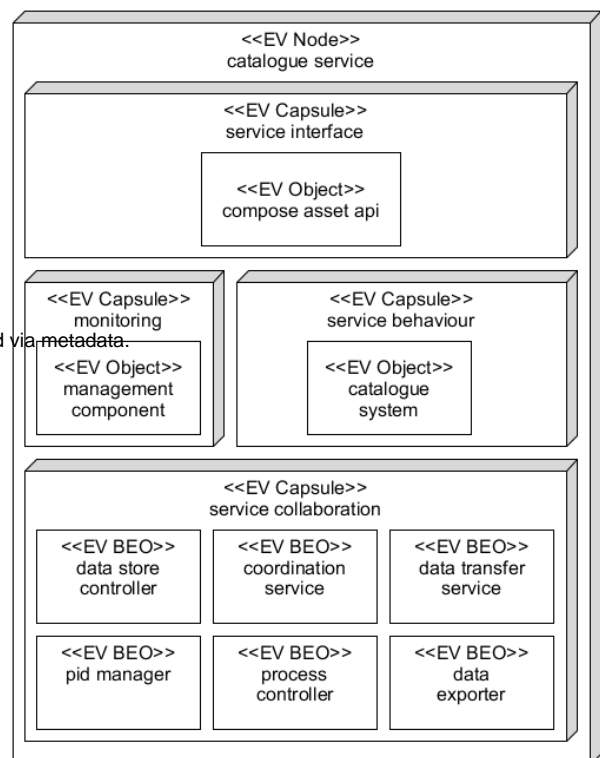
Compose Asset Configuration

The configuration for composing existing assets depends on seven BEOs: [catalogue service](#), [data transfer service](#), [data exporter](#), [pid manager](#), [data store controller](#), [coordination service](#), and [process controller](#),

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for supporting the compose asset process.

Name:	Compose Asset
Process:	Compose (selected parts of) asset into workflow accessed via metadata.
Resources	<ul style="list-style-type: none"> asset pid (precondition: discovery) filtering parameters (metadata, optional) destination parameters scheduling parameters
Outputs	<ul style="list-style-type: none"> asset (copy, partial or complete) metadata <ul style="list-style-type: none"> asset pid asset reference (publishable) asset description
Options	<ul style="list-style-type: none"> compose complete compose partial



Update Asset Configuration

The configuration for discovering an asset depends on four BEOs: [catalogue service](#), [annotation service](#), [pid manager](#), and [data store controller](#).

The figures on the right shows the configuration of the service using the recommended [Container Structure](#).

This configuration shows a self contained service which already contains the required objects for supporting the discovery process.

Name:	Discover Asset
Process:	Update an asset and/or metadata
Resources	<ul style="list-style-type: none"> • asset pid • update parameters (metadata) • asset (complement or replacement)
Outputs	<ul style="list-style-type: none"> • metadata <ul style="list-style-type: none"> • asset pid • asset reference (publishable) • asset description
Optionsup	<ul style="list-style-type: none"> • update metadata only • annotate asset • annotate metadata • update global pid

