CSP Server Configuration API

Version 1.0.2

Paths

`/v1/updates/{cspId}`

GET /v1/updates/{cspId}

Description

Retrieves a list of available updates, for registered modules of the CSP. See the "register" API call on how to register modules

Parameters

Name	Located in	Description	Required	Schema
cspId	path	a unique identifier that defines a Registered and Known CSP. The csp identifier follows the UUID formatted as text, for 36 characters total, arranged as 8-4-4-4-12. Example = 123e4567-e89b-12d3-a456-426655440000	Yes	⇄string

Responses

Code	Description	Schema
200	List of available updates by module	⇄ ▼ **UpdateInformation {** *contains update information for the selected CSP* dateChanged: ▶ string * available: ▶ { } }
		▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer *

404	cspId not found; failure to identify cspId presented	⇄	responseText: ▶ string * responseException: string * }
500	Systemic failure	⇄	▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }

Try this operation

/v1/appInfo/{cspId}

POST /v1/appInfo/{cspId}

Description

Submits a body that contains information of the CSP

Parameters

Name	Located in	Description	Required	Schema
cspId	path	a unique identifier that defines a Registered and Known CSP. The csp identifier follows the UUID formatted as text, for 36 characters total, arranged as 8-4-4-4-12. Example = 123e4567-e89b-12d3-a456-426655440000	Yes	⇄ string
heartbeatInfo	body		Yes	▼ **AppInfo {** *AppInfo object* name: ▶ string * ⇄ recordDateTime: ▶ string *

the information

moduleInfo: ▶ **ModulesInfo** { }

}

Responses

Code	Description	Schema
200	Successful response	⇄ ▼ **Response {** *Accepted Response object* responseCode: ▶ integer * responseText: ▶ string * }
404	cspId not found; failure to identify cspId presented	⇄ ▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }
500	Systemic failure	⇄ ▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }

Try this operation

/v1/register/{cspId}

POST /v1/register/{cspId}

Description

Register a NEW csp or register for an existing CSP the modules that are being installed

Parameters

Name	Located in	Description	Required	Schema
cspId	path	a unique identifier that defines a Registered and Known CSP. The csp identifier follows the UUID formatted as text, for 36 characters total, arranged as 8-4-4-4-12. Example = 123e4567-e89b-12d3-a456-426655440000	Yes	⇄ string
cspRegistration	body	a block of information to register the CSP being installed	Yes	⇄ ▼ **Registration {** *Registration object for a new CSP* name: ▶ string * domainName: string registrationDate: ▶ string * externalIPs: ▶ [] internalIPs: ▶ [] registrationIsUpdate: ▶ boolean * contacts: ▶ [] moduleInfo: ▶ **ModulesInfo { }** }

Responses

Code	Description	Schema
		▼ **Response {** *Accepted Response object* responseCode: ▶ integer

200	received when the CSP is registered successfully	⇄ * responseText: ▶ string * }
400	received when CSP updates entries with "registrationIsUpdate = false"	⇄ ▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }
404	received when CSP requests update but original entry does not exist OR when cspId is not recognized	⇄ ▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }
500	received when the CSP is not registered due to system error	⇄ ▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }

Try this operation

/v1/update/{cspId}/{updateHash}

GET /v1/update/{cspId}/{updateHash}

Description

Retrieves a list of available updates, for registered modules of the CSP. See the "register" API call on how to register modules

Parameters

Name	Located in	Description	Required	Schema
cspId	path	a unique identifier that defines a Registered and Known CSP. The csp identifier follows the UUID formatted as text, for 36 characters total, arranged as 8-4-4-4-12. Example = 123e4567-e89b-12d3-a456-426655440000	Yes	⇄string
updateHash	path	a unique identifier hash for the given update. The system must verify that this hash is available for this cspId to download; then it provides the byte stream for this update object.	Yes	⇄string

Responses

Code	Description	Schema
200	a byte stream for the data requested	⇄ ▼ file (application/octet-stream)
400	received when this update is not available for this CSP	⇄ ▼ **ResponseError {** *Failed Response object* responseCode: ▶ integer * responseText: ▶ string * responseException: string * }

404	cspId is not recognized	⇄ ▼ **ResponseError** {
		Failed Response object
		responseCode: ▶ integer *
		responseText: ▶ string *
		responseException: string *
		}

Try this operation

Models

VersionDefinition

⇄ ▶ **VersionDefinition** integer *

Registration

⇄ ▶ **Registration** { }

ContactDetails

⇄ ▶ **ContactDetails** { }

AppInfo

⇄ ▼ **AppInfo** {
AppInfo object
name: ▶ string *
recordDateTime: ▶ string *
moduleInfo: ▼ **ModulesInfo** {
The json object that contains information on the modules installed
modules: ▶ []
data: ▶ { }

```
                    }
  }
```

Response

```
  ▼ Response {
    Accepted Response object
⇄   responseCode: ► integer *
    responseText: ► string *
  }
```

ResponseError

```
  ▼ ResponseError {
    Failed Response object
    responseCode:      ► integer *
⇄
    responseText:      ► string *
    responseException: string *
  }
```

ModulesInfo

```
  ▼ ModulesInfo {
    The json object that contains information on the modules installed
    modules: ▼ [
               array of the modules names, see example
               string
             ]
    data:    ▼ {
               name:                  string
               additionalProperties: ▼ ModuleData {
                                         Map of data specific to a module
⇄                                        fullName:     string *
```

```
                version:        ▶ VersionDefinition integer *
                installedOn:    ▶ string *
                active:         boolean *
                hash:           string *
                startPriority:  number
            }
        }
    }
```

ModuleData

```
▼ ModuleData {
  Map of data specific to a module
  fullName:       string *
  version:        ▼ VersionDefinition integer * minimum:10000 maximum:90000
                    the version definition used for communicating with the CSP APIs; it has a visual and a
⇄                   numeric format. The visual format can be created by splitting the numeric at specific points,
                    (0 based) 1, 2 example: 12000 split as 1.2.000
  installedOn:    ▼ string * (date)
  active:         boolean *
  hash:           string *
  startPriority:  number
}
```

UpdateInformation

```
▼ UpdateInformation {
  contains update information for the selected CSP
⇄   dateChanged:  ▶ string *
    available:    ▶ { }
}
```

ModuleUpdateInfo

▼ **ModuleUpdateInfo {**

list of available updates for a module

name: string *

description: string *

version: ▼ **VersionDefinition** integer * minimum:10000 maximum:90000

the version definition used for communicating with the CSP APIs; it has a visual and a numeric format. The visual format can be created by splitting the numeric at specific points, (0 based) 1, 2 example: 12000 split as 1.2.000

released: ▶ string *

isIncremental: ▶ boolean *

hash: ▶ string *

}