

Documentation

- [Deployment of new releases](#)
- [Add new product to an existing distribution](#)
- [Add new distribution](#)

Deployment of new releases

Process to deploy a new release manually:

- 1. **Create the json files using one of the following options:**
 - 1.1. **yaml templates** - this option needs to use yaml2json converter to generate json file
 - 1.2. **json templates** - use the json SW/templates/distribution.template.json or product.template.json
- 2. **Prepare distribution and product**
 - **a)** If it's a product from a **new distribution** we need to add the distribution first:
 - **add-new-distribution** using the templates in SW/templates/distribution.template.json
 - Install/Copy the template into json_dir/distributions/distribution-name-version.json
 - **add new product** to the distribution file json_dir/distributions/distribution-name-version.json products array
(example of a json product item to add to the products array)

```
{ "name" : "emi.apel", "display" : "APEL publisher", "version" : "3.2.7", "apicallback" :  
"/static/SW/APEL publisher3.2.7.json", "target" : [ { "platform" : "sl5", "arch" :  
"x86_64", "repositoryURL" : "sw/production/umd/1/sl5/x86_64/base" } ] }
```

- **b)** If it's a **product from an existing distribution** we need to:
 - **add "productname.json" full data || "name"=name+version + capabilities=[] to the distribution file**
 - json_dir/distributions/distribution-name-version.json
 - **add product to the catalog**
 - json_dir/software-catalog.json
 - **Finally add product.json to the respective directory and create the release.json manually**
 - json_dir/products/distribution-name/distribution-version/
 - and add the same product file to the root of the json_dir/ with the name release.json

Add new product to an existing distribution

Ways to add a new product to an existing distribution (Work in progress)

dir structure:

```
/var/www/be/  
data/ (The latest json data for the FE)   
├─ config.js  
├─ distributions/  
│   ├── CMD-ONE  
│   │   ├── 0.0.0.json  
│   │   └─ ...  
│   ├── CMD-OS  
│   │   ├── 0.0.0.json  
│   │   └─ ...  
│   └─ UMD  
│       ├── 1.0.0.json  
│       └─ ...  
├─ products/  
│   ├── CMD-ONE  
│   │   ├── 0.0.0/  
│   │   │   └─ productname.json  
│   │   │   └─ ...  
│   ├── CMD-OS  
│   │   ├── 0.0.0/  
│   │   │   └─ productname.json  
│   │   │   └─ ...  
│   └─ UMD  
│       ├── 1.0.0/  
│       │   └─ productname.json  
│       │   └─ ...  
├─ software-catalog.json (Complete list of products)  
├─ software-distributions.json (Complete list of distributions)  
└─  
templates/ (All the templates needed to convert yaml to json or using json directly)  
├─ product.yaml  
├─ distribution.yaml  
└─ product.json
```

```
└─ distribution.json
```

For both approaches the first step is always convert/prepare the release.json

```
cd /var/www/be
```

Copy the template:

```
cp SW/templates/product.yaml ./release.yaml
```

Convert the yaml to json:

```
yaml2json release.yaml (this will create a release.json file in the same dir)
```

Now we have a json file:

```
be$ release.json
```

1st approach - using the Database (DB)

Add the data manually to the DB:

- Product
- Distribution product data Commands...

Examples:

```
cat create_table.sql |sqlite3 db/egi-repos-fe-test.db
```

```
sqlite3 db/egi-repos-fe-test.db ".read insert_data.sql"
```

```
sqlite3 db/egi-repos-fe-test.db "select MAX(f1) from tbl2;" ".exit"
```

```
sqlite3 db/egi-repos-fe-test.db "INSERT INTO tbl2 ('f1','f2','f3') VALUES('001','teste','tttt');" ".exit"
```

```
INSERT INTO "main"."products" ("id", "name", "productName", "version", "majorVersion", "minorVersion", "revision",  
' , 'deployed', 'parak@cesnet.cz', 'parak@cesnet.cz', 'sw/production/cmd-one/1', 'CESNET', 'CESNET', '2018-11-12 1
```

Export/prepare all the data from DB (replacing the data in data/*)

```
$ export-full-dataset.sh
```

```
.....  
php export-distributions.php
```

```
php export-distributions.php --individual-distributions
```

```
php export-products.php
php export-products.php --individual-products
```

2nd approach - add the data manually to the json files

Manually add product

```
cp release.json data/
cp release.json data/products/distribution-type/distribution-version/$productname(change this).json
```

add product data (to the products list in distribution):

data/distributions/distribution-type/\$distribution-version.json (products[] array)

ex. of product item:

```
{
  "productId": 32,
  "name": "emi.apel",
  "version": "3.2.7",
  "capabilities": ["Accounting"],
  "status": 1
}
```

add product full data (to the software catalog):

data/software-catalog.json (item[] array)

ex. of product item: (DON'T FORGET TO ADD A COMMA , BEFORE THE BRACKET)

```
,
{
  "name": "bdii.site-bdii",
  "version": "1.2.1",
  "documentationLinks": [],
  "target": [
    {
      "platform": "centos7",
      "arch": "x86_64",
      "repositoryURL": "sw\\production\\cmd-one\\1\\centos7\\x86_64\\updates",
      "gpgkey": "",
      "rpms": [],
      "status": 1
    }
  ],
  "capabilities": ["Client Tools"],
  "currentState": "deployed",
  "technologyProvider": "BDII",
  "contact": "email of contact",
```

```
"technicalContact": "email of contact",
"description": "text description...",
"releaseDate": "2018-11-12 12:34:40",
"majorVersion": 1,
"minorVersion": 2,
"revisionVersion": 1,
"releaseNotes": "http://gridinfo.web.cern.ch/developers/resource-bdii",
"changeLog": "Same version as in SL6\n",
"repositoryURL": "sw/production/cmd-one/1",
"distributionType": "CMD-ONE",
"distributionVersion": "1.1.0",
"softwareConfigurationLink": ""
}
```

Add new distribution

Ways to add a new distribution (Work in progress)

```
cd /var/www/be
```

Create distribution folder:

```
mkdir data/distributions/%distribution-name%
```

Use/Copy the template:

```
cp SW/templates/distribution.yaml ./
```

Convert the yaml to json: (need to have node.js installed)

```
yaml2json distribution.yaml (this will create a distribution.json file in the same dir)
```

Now we have a json file:

```
distribution.json
```

Rename it to the distribution-name:

```
mv distribution.json %distribution-version%.json  
mv %distribution-version%.json data/distributions/%distribution-name%/
```

1st approach - using the Database (DB)

Add the data manually to the DB:

- Product

- Distribution product data Commands...

Examples:

```
cat create_table.sql |sqlite3 db/egi-repos-fe-test.db
```

```
sqlite3 db/egi-repos-fe-test.db ".read insert_data.sql"
```

```
sqlite3 db/egi-repos-fe-test.db "select MAX(f1) from tbl2;" ".exit"
```

```
sqlite3 db/egi-repos-fe-test.db "INSERT INTO tbl2 ('f1','f2','f3') VALUES('001','teste','ttttt');" ".exit"
```

```
INSERT INTO `distributions` (
    distributionType, distributionVersion, currentState,
    majorVersion, minorVersion, revisionVersion,
    releaseDate, created,
    lastUpdated, description,
    repositoryURL, releaseNotes,
    additionalDetails, installationNotes,
    contact, technicalContact,
    knownIssues, changeLog
)
VALUES (",,,,,,,,,,,,,,,,,,,,,,,,,,,,");
```

Export/prepare all the data from DB (replacing the data in data/*)

```
$ export-full-dataset.sh
```

.....

```
php export-distributions.php
```

```
php export-distributions.php --individual-distributions (Each distribution) php export-products.php
```

```
php export-products.php --individual-products (Each product)
```

2nd approach - add the data manually to the json files

Manually add distribution

add distribution data (to the software distributions catalog):

data/software-distributions.json (item[] array)

ex. of distribution item: (DON'T FORGET TO ADD A COMMA , BEFORE THE BRACKETS)

```
{
    "distributionType": "",
    "distributionVersion": "%version%0.0.0",
    "currentState": "deployed",
```

```
"releaseDate": "",
"description": "",
"products": [
  {
    "name": "product-name",
    "version": "0.0.0",
    "capabilities": ["Client Tools"],
    "status": 1
  }
]
}
```