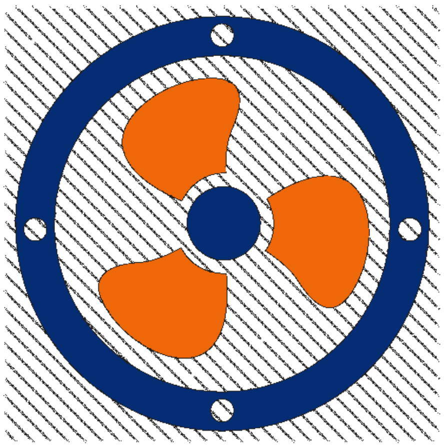
Data Exchange Component (DEC)
Software User Manual



March 13, 2020

Contents

1 Acronyms

Table 1.1: Acronyms

Acronym	**Text**
COTS	Commercial Off The Shelf
DEC	Data Exchange Component
FAQ	Frequently Asked Questions
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I/F	Interface
ICD	Interface Control Document
IERS	International Earth Rotation Service
OS	Operative System
RPF	Reference Planning Facility
RVM	Ruby Version Manager
SFTP	Secure File Transfer Protocol
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Socket Layer
SW	Software
TCP	Transfer Control Protocol
URL	Universal Resource Locator
UTC	Universal Time Coordinated
WebDAV	Web Distributed Authoring & Versioning
XML	Extensible Markup Language

2 Introduction

2.1 Purpose & Scope

The Data Exchange Component (DEC) is a SW helper to gather, transform, circulate, and archive files autonomously amongst different interfaces.

The scope of DEC SW usually lies on the different ICD defined for communication and exchange of data (generally *files*). As such, it relies on different COTS to delegate the implementation of the different protocols supported.

2.2 Design Drivers

This section enumerates a high level overview for the main design drivers that allows DEC to interface efficiently for the configured exchanges:

- *Flexibility* : the ability to *pull* & *push* files from configurable interfaces and configurable circulation rules
- *Robustness* : this is to perform "atomic" operations during file circulation ; the state for each operation is always known being network errors tolerant
- *Performance* : parallelisation of the circulation to fruit the available network bandwidth, support of file compression mechanisms to reduce the footprint of transfers and local disseminations for which duplication of files by *cp* or *mv* can be avoided by usage of *hardlinks*
- *Resiliency* : to *autonomously* recover from network errors, downtime and eventual glitches

2.3 Features

This section enumerates some of the main high-level features offered by DEC.

2.3.1 Pull Circulations

The DEC SW offers the capacity to *autonomously* pull files from the different configuration interfaces for which the frequency of polling can be configured for each interface to adapt for the workflows settled between the suppliers and the consumers.

2.3.2 Push Circulations

The DEC SW offers the capacity and command line interfaces to *push* files towards the different configured interfaces.

It is as well possible to automate *push* transfers on regular basis with configurable frequency for every interface.

2.4 Download DEC SW

This section contains the links to download the DEC SW and the *Gemfile* with the definition of the *gem* dependencies. It is necessary to be logged-in with your DEIMOS *gmail* account to *authorise* the download.

- *Gemfile* : file with the *gem* dependencies
- *dec-stable.gem* : DEC gem installer
- *dec_test.bash* : definition of the environment variables used by the unit tests for bash console
- *dec_test.env* : definition of the environment variables used by the unit tests for docker container parametrisation

3 Install

3.1 Install Environment

This section covers the install of *Ruby* interpreter and the *gem* dependencies needed by DEC SW. It is recommended that the *Ruby* interpreter is done locally for the Linux user who shall execute the DEC SW.

3.1.1 Install Ruby

The *Ruby* interpreter can be obtained and installed in many different ways ; this manual describes how to install it local to the user which shall execute the DEC SW.
The install is based on RVM, which is a command-line tool which allows you to easily install, manage, and work with multiple ruby environments from interpreters to sets of gems.

```
$> \curl -sSL https://get.rvm.io | bash -s stable --ruby
```

The following command installs a specific version of ruby interpreter ; the version specified below corresponds to the one used for the unit tests and it is required as a mandatory precondition at installation time according to the gem file definition.

```
$> rvm install ruby-2.6.5
```

3.1.2 Install Bundler

The ruby pre-requisites can be installed using on bundler gem, by tracking and installing the exact gems and versions which are needed by DEC.

```
$> gem install bundler
```

3.1.3 Install Gems

The *gems* pre-requisites by DEM are defined in the *Gemfile* ; in order to install the exact gems and versions which are needed by DEC from the directory in which the *Gemfile* is placed execute the following command in the shell.

```
$> bundle install
```

3.2 Install DEC

In order to install DEC SW, execute the following command in the shell at the directory in which the DEC gem file is placed:

```
$> gem install --local dec_stable.gem
```

3.3 Uninstall DEC

In order to uninstall DEC SW, execute the following command in the shell:

```
$> gem uninstall dec.gem
```

```
Remove executables:
decValidateConfig, decCheckConfig, decCheckSent, decConfigInterface2DB, decDeliverFiles, decGetFi

in addition to the gem? [Yn]
```

Press 'Y' key to remove the executables as well

3.4 COTS Required

This section enumerates the COTS which are used by DEC SW for exchange of file by some network protocol implementation, or file transformations associated to those exchanges.
It is not part of the scope of this manual how to provision these COTS ; since they can be obtained naturally with OS distribution, downloaded with its native package manager, or manually downloaded.

3.4.1 Databases

This section enumerates the different databases which can be used by DEC SW. Only *sqlite3* is *mandatory* to allow the execution of the entire set of *unit tests*. Below the different databases that have used at some deployment. It is recalled that it is possible to execute the DEC SW without any database by usage of flag *"–nodb"*.

- *sqlite3* : is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine
- *PostgreSQL* : object-relational database system with a strong reputation for reliability, feature robustness, and performance
- *MySQL* : a high performance, scalable database management system

3.4.2 Network Tools

This section enumerates the network tools which can be used by DEC SW.

- *ncftp* : application programs implementing the File Transfer Protocol (FTP)
- *sftp* : application programs implementing the Secure File Transfer Protocol (SFTP)
- *curl* : command line tool and library for transferring data with URLs (WebDAV)

3.4.3 File Compression Tools

This section enumerates the file compression tools which can be used by DEC SW.

- *7-zip* : is a file archiver with a high compression ratio ; name of the package can be *"p7zip"*
- *zip / unzip* : provide free, portable, high-quality versions of the *Zip* and *UnZip* compressor-archiver utilities
- *gzip* : The *gzip* reduces the size of the named files using Lempel–Ziv coding (LZ77)
- *compress* : The *compress* utility reduces the size of the named files by using adaptive Lempel-Ziv coding algorithm

3.4.4 File Transformation Tools

This section enumerates the file transformation tools which can be used by DEC SW.

- *xmllint* : is a command line XML parser which is part of the *libxml2* utils

3.5 Installation Verification

3.5.1 Verification with Unit Tests

This section describes how to execute the *unit tests*, which have been designed to be transparent and harmless in front of the potential different execution environments (i.e. development, integration, production), being their execution a simple and effective manner to verify the correct installation of the DEC SW.

The prerequisites to be able to successfully execute the *unit tests* are :

- an OS user *dectest*
- SFTP server running on *localhost* allowing login to *dectest* using the SSH keys
- FTP server running on *localhost* allowing login to *dectest* using the *password dectest*

```
$> decUnitTests
```

The results of the unit test should show no failures neither errors ; the execution time in a 2.66 GHz Intel Core 2 Duo is about 10 minutes approximately.

```
.
Finished in 590.39725 seconds.
------------------------------------------------------------------------------------------
17 tests, 112 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifi
100% passed
------------------------------------------------------------------------------------------
0.03 tests/s, 0.19 assertions/s
```

3.5.2 Verification with Operational Interface

This section describes how to verify the correct installation of DEC SW by execution of a test with the IERS *operational* service, for which Internet connectivity is required for the FTP protocol. Note that it is not possible to ensure the connectivity availability by such service and sometimes test may fail by reply of *530 connect failed: Address already in use. No response from server.*

```
$> decUnitTests_IERS
```

```
.
Finished in 54.38868 seconds.
--------------------------------------------------------------------------------
2 tests, 25 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifica
100% passed
--------------------------------------------------------------------------------
0.02 tests/s, 0.33 assertions/s
```

In case of deployment without any database, the tests can be restricted to the ones which make usage of the *"–nodb"* execution option:

```
$> decUnitTests_IERS -n test_decGetFromInterface_NODB
```

```
.
Finished in 54.38868 seconds.
--------------------------------------------------------------------------------
1 tests, 18 assertions, 0 failures, 0 errors, 0 pendings, 0 omissions, 0 notifica
100% passed
--------------------------------------------------------------------------------
0.02 tests/s, 0.23 assertions/s
```

4 Configuration

4.1 Interfaces

This section covers the *interfaces* configuration of a DEC node.
The configuration of every interface is defined in *dec_interfaces.xml*.

4.1.1 TXRXParams

The *TXRXParams* configuration item defines the parameters which rule the circulation in pull and push mode.

- *Enabled4Sending* : *boolean* flag for the activation of the circulations in *pull* mode
- *Enabled4Receiving* : *boolean* flag for the activation of the circulations in *push* mode
- *ImmediateRetries* : number of retries to be applied to recover eventual *upload* failures when *pushing* a file
- *LoopRetries* : number of loops to be applied to recover eventual *upload* failures when *pushing* a file
- *LoopDelay* : delay in *seconds* between two consecutive loops (if needed) for push
- *PollingInterval* : delay in *seconds* in between consecutive iterations for *pull* circulations
- *PollingSize* : maximum number of files to be *pulled* during an iteration
- *ParallelDownload* : number of simultaneous file downloads for pull circulations

4.1.2 Server

The *Server* configuration item defines the parameters which rule the network protocol of choice selected to rule the file circulations either in *pull* and *push* mode.

- *Protocol* : unit tests are covering today *"FTP"*, *"SFTP"*, *"WEBDAV"* - http and http(s) -, and *"LOCAL"* ; if configuration item is set to *"LOCAL"* then network configuration items such as *Hostname*, *Port*, *User* and *Pass* are not used
- *Hostname* : hostname of the interface
- *Port* : TCP port

- *User* : registered user for the server at the *Hostname* & *Protocol* server
- *Pass* : password for authentication ; in case of secure *SecureFlag* set to true, the SSH keys associated to the system user are used instead
- *RegisterContentFlag* : *boolean* flag to activate tracking of the files available for *pulling* without effective download
- *RetrieveContentFlag* :*boolean* flag to activate the download of the files available for *pull circulation*
- *SecureFlag* : *boolean* flag to activate *encrypted* SSH communications
- *CompressFlag* : *boolean* flag to activate *compression* of the *encrypted* SSH communications
- *DeleteFlag* : *boolean* flag to delete the file upon successful circulation
- *PassiveFlag* : *boolean* flag to activate *compression* of the *encrypted* SSH communications
- *CleanUpFreq* : defined in seconds used by the clean-up daemon if enabled to clean-up previously circulated files in *push mode*

4.1.3 DeliverByMailTo

List of email *Address* configuration items used to *push* files using SMTP.

4.1.4 Notify

The *Notify* configuration item serves to activate the generation and delivery of an email receipt carrying the name of the files which have been *pushed* towards a given *Interface*.

- *SendNotification* : *boolean* flag to activate the
- *To* : list of email *Address* configuration items which will receive the notification

```
<Notify>
        <SendNotification>true</SendNotification>
<To>
        <Address>mario.bros@gmail.com</Address>
        <Address>mario.draghi@deimos-space.com</Address>
        <Address>mario.cipollini@esa.int</Address>
</To>
</Notify>
```

4.1.5 Events

The *Events* configuration item host the activation and shell command associated to the event when circulating towards the interface.

```
<Events>
<Event Name="OnReceiveNewFilesOK"       executeCmd="echo :-)"/>
<Event Name="OnReceiveError"            executeCmd="echo :-("/>
<Event Name="NewFile2Intray"            executeCmd="echo %F"/>
</Events>
```

Below the possible *Event Name* values which can be configured:

- *OnSendOK* : it is raised upon a successful loop to push
- *OnSendNewFilesOK* : it is raised upon a successful *push* loop of file(s)
- *OnSendERROR* : it is raised if any file failed the *push* circulation
- *OnReceiveOK* : it is raised upon a successful loop for pull
- *OnReceiveNewFilesOK* : it is raised upon a successful *pull* loop of file(s)
- *OnReceiveNewFile* : it is raised every new file *pulled*
- *OnReceiveERROR* : it is raised if any *pull* of file has failed
- *OnTrackOK* : it is raised upon a successful loop to check availability pull
- *NewFile2Intray* : it is raised upon every new file locally disseminated after pulling it ; %f specifies the *filename* disseminated ; %F specifies the *full path filename* disseminated ; %d specifies the directory name in which the file has been disseminated

4.2 Pull circulations

The mechanism to *select*, *download* and locally *disseminate* files to a DEC node is referred to *pull mode*.
The configuration of the pull circulation rules is defined in *dec_incoming_files.xml*.

4.2.1 Interface pick-up points

The *Interface* configuration item defines the interfaces pick-up point(s) from which the exposed files are polled.

- *Name* : this is the configuration item *identifier* which is used to identify a given interface and refer to it in other sections of the configuration
- *LocalInbox* : this is the *local* directory in which files are initially placed upon successful download ; if no dissemination rule is applied, this directory becomes the final destination
- *DownloadDirs* : a list of *Directory* configuration items to define the *remote* directories of this interface from which the files will be pulled ; the attribute *DepthSearch* defines the directory sub-levels which are polled.

```
<ListInterfaces xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<Interface>
        <Name>LOCALHOST_SECURE</Name>
        <LocalInbox>/tmp/in_basket_if_localhost_secure</LocalInbox>
        <DownloadDirs>
                <Directory DepthSearch="0">/download1</Directory>
                <Directory DepthSearch="0">/download2</Directory>
        </DownloadDirs>
</Interface>

<Interface>
        <Name>IERS</Name>
        <LocalInbox>/tmp/dec/if_iers_in_basket</LocalInbox>
        <DownloadDirs>
                <Directory DepthSearch="0">ser7</Directory>
        </DownloadDirs>
</Interface>

</ListInterfaces>
```

4.2.2 File filters

This configuration item defines the filters to select the files to be pulled from the interface pick-up point(s).

- *File Type* attribute : it defines the file to be pulled, allowing the definition of wildcards
- *Description* : free text to capture a human friendly definition of the interface
- *FromList* : a list of *Interface* identifiers from which the file will be pulled

```
<ListFiles xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<File Type="tai-utc*">
        <Description>TAI UTC correlation</Description>
        <FromList>
                <Interface>IERS</Interface>
                <Interface>LOCALHOST_NOT_SECURE</Interface>
        </FromList>
</File>

<File Type="finals.a*">
        <Description>TAI UTC correlation consolidated</Description>
        <FromList>
                <Interface>IERS</Interface>
                <Interface>LOCALHOST_NOT_SECURE</Interface>
        </FromList>
</File>
```

```
<File Type="S2?_*">
        <Description>Sentinel-2 Ground Segment</Description>
        <FromList>
                <Interface>LOCALHOST_SECURE</Interface>
                <Interface>LOCALHOST_NOT_SECURE</Interface>
        </FromList>
</File>

</ListFiles>
```

4.2.3 Dissemination rules

This configuration item defines the local Intrays and the dissemination rules to be applied upon successful pull of files.
Firstly it is defined the *ListIntrays Intray* configuration items:

- *Name* : this is the intray *identifier* which is used to identify the final location the files pulled are placed
- *Directory* : this is the *local* directory associated to the in-tray

```
<ListIntrays>

<Intray>
        <Name>S2ALL</Name>
        <Directory>/local_dissemination/S2ALL</Directory>
</Intray>

<Intray>
        <Name>S2A</Name>
        <Directory>/local/S2A</Directory>
        <Compress>7z</Compress>
</Intray>

<Intray>
        <Name>S2B</Name>
        <Directory>/local/S2B</Directory>
</Intray>

<Intray>
        <Name>GPS</Name>
        <Directory>/local/GPS</Directory>
</Intray>

</ListIntrays>
```

Then the rules for local dissemination of the pulled files into the in-tray(s) are defined. Only the first rule which matches the file name or wildcard defined in the *File Type* attribute is applied ; as such it is recommended to sort the rules ranking from the more restrictive ones.

- *Name* : this is the intray *identifier* which is used to identify the final location the files pulled are placed
- *HardLink* : this is a *boolean* flag to hardlink in the file-system a pulled file in case of definition of different in-trays for multiple dissemination into
- *ToList* : this is the list of *Intray* identifiers configuration item the defined *File Type* is *disseminated* into

```
<ListFilesDisseminated>

<File Type="tai-utc*">
        <HardLink>False</HardLink>
        <ToList>
                <Intray>GPS</Intray>
        </ToList>
</File>

<File Type="finals*">
        <HardLink>false</HardLink>
        <ToList>
                <Intray>GPS</Intray>
        </ToList>
</File>

<File Type="S2A_*.*">
        <HardLink>true</HardLink>
        <ToList>
                <Intray>S2A</Intray>
                <Intray>S2ALL</Intray>
        </ToList>
</File>

<File Type="S2B_*.*">
        <HardLink>true</HardLink>
        <ToList>
                <Intray>S2B</Intray>
                <Intray>S2ALL</Intray>
        </ToList>
</File>

<File Type="S2__*">
        <HardLink>False</HardLink>
        <ToList>
                <Intray>S2ALL</Intray>
        </ToList>
</File>

</ListFilesDisseminated>
```

4.3 Push circulations

The mechanism to fetch and upload files from a DEC node is referred to *push mode*.
This needs to be written

4.4 Environment Variables

The execution of DEC SW can be parametrised through the usage of environment variables which are loaded from the shell. This section describes them.

4.4.1 Database Variables

This section describes the environment variables used by DEC SW to define the database settings.

- *DEC_DB_ADAPTER* : this is the database used for recording the operations unless execution is performed with the *–nodb* execution flag is used ; values can be *"postgresql"*, *"sqlite3"*, etc driven by your database of choice
- *DEC_DATABASE_NAME* : this is a database name in which the circulation operations and file availability tracking are recorded
- *DEC_DATABASE_USER* : this is the database user name
- *DEC_DATABASE_PASSWORD* : this is the database user password

4.4.2 General Variables

This section describes the environment variables used by DEC SW to define the database settings.

- *DEC_CONFIG* : this is the directory in which the DEC config files are present
- *DEC_TMP* : this is the directory used for temporal / transient operation prior final availability in the data sinks
- *HOSTNAME* : this environment variable defines the current hostname which is used for mail authentication purposes

4.4.3 File Transfer for RPF

This section describes the specific environment variables used by DEC SW acting as File Transfer for the RPF.

- *FTPROOT* : this is the directory which is used to place the files which will be *pushed* as part of each ROP delivery
- *RPF_ARCHIVE_ROOT* : this is RPF archive root directory from which the files to be transferred are gathered
- *RPFBIN* : this is RPF binary directory in which the RPF tools invoked by the File-Transfer are placed (i.e. *put_report.bin*, *removeSchema.bin*, *write2Log.bin*)

4.4.4 Push Circulations

This section describes the specific environment variables used by DEC SW for *push circulations*.

- *DEC_DELIVERY_ROOT* : this is the source directory of the files to be *pushed*

4.5 Database

This section describes how to initialise the database configuration for DEC driven by some environment variable settings described below. It is however recalled that it is possible to execute the DEC *without* a database by usage of the flag *"–nodb"* which precludes some features such as control flow, throughput computation however allowing essential file data circulations.

4.5.1 Creation

This section describes how to initialise the database by creation of the tables and index used by DEC SW.
In order to create the database, execute the following command in the shell:

```
$> decManageDB --create-tables
```

In order to remove the database, execute the following command in the shell:

```
$> decManageDB --drop-tables
```

4.5.2 Configuration of Interfaces

This section describes how to configure within the database the interface identifiers defined in the configuration file *dec_interfaces.xml* :

```
$> decConfigInterfaceDB --process EXTERNAL
```

It is also possible to register the interface identifiers according to its name definition in *dec_interfaces.xml*:

```
<Interface Name="IERS">
```

The following command can be used to register an interface identifier in the database:

```
$> decConfigInterfaceDB --add IERS
```

4.6 Circulation Reports

This section describes how to configure the DEC SW to generate the *circulation reports* containing information of the files pulled, pushed or which are available published by a given interface.

- *RetrievedFiles* : this report enumerates the files *pulled* from a given interface during a polling loop
- *DeliveredFiles* : TBW
- *UnknownFiles* : TBW
- *EmergencyDeliveryFiles* : TBW

The configuration for the generation of the *circulation reports* is defined in the configuration file *dec_config.xml*.
The generated circulation reports are placed in directory location defined by the full path specified in the *ReportDir* configuration item:

```
<ReportDir>/dec/dec_reports</ReportDir>
```

4.6.1 Pulled Files

The DEC SW can generate a report enumerating the files which have been *pulled* from a given interface during a polling loop. The name of this *circulation report* is *"RetrievedFiles"*. Below there are the associated configuration items:

```
<Report Name="RetrievedFiles">
<Enabled>true</Enabled>
<Desc>List of Files Retrieved</Desc>
<FileClass>OPER</FileClass>
<FileType>DEC_R_PULL</FileType>
</Report>
```

4.6.2 Unknown Files

The DEC SW can generate a report enumerating the files which did not match any filtering rule for their retrieval from a given interface during a polling loop. The name of this *circulation report* is *"UnknownFiles"*. Below there are the associated configuration items:

```
<Report Name="UnknownFiles">
<Enabled>true</Enabled>
<Desc>List of unknown Files present</Desc>
<FileClass>OPER</FileClass>
<FileType>DECUNKNOWN</FileType>
</Report>
```

4.7 Circulation Options

This section describes different circulation options or the DEC SW execution behaviour in particular cases. The associated configuration items are defined in the *dec_config.xml* configuration file.

4.7.1 Pulled Files

The associated configuration items are defined in *Options/Download*

4.7.1.1 Unknown Files

The DEC SW is able to delete without download the files published by any interface in a *Directory DownloadDir* which do not meet any of the *ListFiles* matching criteria rules defined in dec_incoming_files.xml.

```
<DeleteUnknownFiles>false|true</DeleteUnknownFiles>
```

4.7.1.2 Duplicated Files

The DEC SW controls and blocks the retrieval of files duplication published by a given interface to avoid feeding the consumer in-trays with the same inputs previously received.

Those duplicated files which are not pulled can become a problem introducing performance penalties when filtering them during each iteration. By configuration below it is possible to instruct DEC remove these duplicated files from the interface pick-up point without downloading them using the *DeleteDuplicatedFiles* configuration item:

```
<DeleteDuplicatedFiles>false|true</DeleteDuplicatedFiles>
```

4.7.2 Pushed Files

TBW

4.8 Configuration Verification

This section describes how to verify the DEC configuration settled according to the previous sections.

To validate the XML files of the DEC configuration, there is an associated a DEC command line tool called *decValidateConfig* which verifies the syntax and semantics associated to configuration items.

Execute the following command in the shell to verify the configuration:

```
$> decValidateConfig --all
```

4.8.1 Check Interfaces

In order to check the entire configuration defined and the connectivity to the different configured interfaces execute the following command:

```
$> decCheckConfig --all
```

If your DEC deployment and configuration does not record the operations into the database (cf. execution with *"–nodb"* flag), alternatively execute the following command in the shell to verify the configuration:

```
$> decCheckConfig --all --nodb
```

5 Reference Log Messages

5.1 Information Messages

5.1.1 Index of Messages

- **[DEC_005]** ”interface” I/F: Polling Started
- **[DEC_050]** ”interface” I/F: Polling Completed / No file(s) available for pull
- **[DEC_060]** ”interface” I/F: Polling Completed / New file(s) available for pull
- **[DEC_105]** ”interface” I/F: ”filename” is available
- **[DEC_110]** ”interface” I/F: Downloaded ”filename” with size ”num_bytes” bytes
- **[DEC_115]** Disseminated ”filename” into ”directory” Intray
- **[DEC_116]** Compressed ”filename” in ”method” at ”directory” Intray
- **[DEC_120]** Deleting unknown file ”filename” available at ”interface” I/F
- **[DEC_125]** Deleting duplicated file ”filename” previously received from ”interface” I/F
- **[DEC_126]** ”interface” I/F: Deleted downloaded file ”filename”

5.1.2 [DEC_005] Interface Polling Start

TBW

5.1.3 [DEC_050] Interface Polling Complete / No File(s) Available

TBW

5.1.4 [DEC_060] Interface Polling Complete / New File(s) Available

This message is logged upon completion filtering the files published in the download interface and verification of eventual duplications in the database. It can be used to measure the time required to perform the filtering operations.

5.1.5 [DEC_105] File Available

The file referenced in the message met any of the filtering rules defined in the configuration *dec_incoming_files.xml* and it is therefore available for download from the interface. This message is logged only when the *–list* mode is used, otherwise the files matching the rules are directly downloaded and logged accordingly.

5.1.6 [DEC_110] File Downloaded

The file referenced in the message has been downloaded from the interface and placed into the *LocalInbox* associated to such interface. If the same file is available again at the same interface, the DEC will detect a file duplication condition.

5.1.7 [DEC_115] File Disseminated

The file referenced in the message met any of the and it has been moved from *LocalInbox* into the Intray(s) matching the first *dissemination rule*.

5.1.8 [DEC_116] File Compressed

The file referenced in the message has been compressed according to the *dissemination rule Compress* optional configuration item of the *Intray* entity.

5.1.9 [DEC_120] Unknown File Deleted

According to the configuration item *DeleteUnknownFiles* defined in *dec_config.xml* configuration, a file available in the *pull* interface, which did not match any filtering rule defined in the configuration *dec_incoming_files.xml* has been deleted from the *DownloadDirs Directory* of such interface it was present.

5.1.10 [DEC_125] Duplicated File Deleted

According to the configuration item *DeleteDuplicatedFiles* defined in *dec_config.xml* configuration, a file available in the *pull* interface, which has been previously downloaded successfully such interface is considered duplicated and it is removed directly in download directory without downloading it.

5.1.11 [DEC_126] Downloaded File Deleted

According to the configuration item *DeleteFlag* defined in *dec_interfaces.xml* for every interface, the file which has been successfully *pulled* is deleted subsequently.

5.2 Warning Messages

5.2.1 Index of Messages

- **[DEC_301]** Detected duplicated file ”filename” already received from ”interface” I/F
- **[DEC_320]** Detected unknown file ”filename” available at ”interface” I/F
- **[DEC_330]** ”filename” is stuck in ”temporal” directory

- **[DEC_331]** "filename" is stuck in LocalInbox directory

5.2.2 [DEC_301] Duplicated File Available

The file reported is available again at the *pull* interface ; if the inventory is active, DEC retrieves just once a given file from each interface. This message will be reported during every polling iteration unless the *duplicated* file is removed from the interface.

5.2.3 [DEC_320] Unknown File Available

The file reported is available in the *pull* interface, but did not match any filtering rule defined in the configuration *dec_incoming_files.xml*. This message will be repeated every polling iteration unless the *unknown* file is removed from the interface.

5.2.4 [DEC_330] File Stuck in Temporal Directory

This warning is associated to error message *[DEC_620]* informing that the file was successfully downloaded and stuck in the temporal directory available for contingency recovery.

5.2.5 [DEC_331] File Stuck in LocalInbox Directory

TBW.

5.3 Error Messages

- *[DEC_000] interface IF is not configured correctly*: TBW
- *[DEC_002] Directory is unreachable*: TBW
- **[DEC_610]** "interface" I/F: Unable to connect to "Hostname" with FTP
- **[DEC_611]** "interface" I/F: reason of the connection failure
- **[DEC_612]** "interface" I/F: Cannot reach directory defined in any of the the "Download-Dirs"
- **[DEC_613]** "interface" I/F: server message associated to the failure associated to the change of directory
- **[DEC_614]** "interface" I/F: Cannot GET "URL"
- **[DEC_620]** "interface" I/F: Could not copy "file" into local "directory"
- **[DEC_625]** Dissemination failure of "file" into intray "directory" using mv command
- **[DEC_626]** Dissemination failure of "file" into intray "directory" using hard-links command

5.3.1 [DEC_000]

Interface xxxx is not configured correctly: TBW

5.3.2 [DEC_200]

Failed sending to interface: TBW

5.3.3 [DEC_610] Unable to Connect with FTP

This error is raised when it is not possible to connect to "Hostname" with FTP.

5.3.4 [DEC_611] Server Error Message

This error code can bring different messages associated to the connection failure [DEC_610] is reporting. Errors can driven by miss-configuration of the Server associated to such interface or its unavailability due to network problems.

5.3.4.1 Hostname is unknown

```
[DEC_611] LOCALHOST_NOT_SECURE I/F: getaddrinfo: nodename nor servname provided, or not known
```

5.3.4.2 Service not enabled by host

```
[DEC_611] LOCALHOST_NOT_SECURE I/F: Connection refused - connect(2) for 127.0.0.1:21
```

5.3.4.3 Failed authentication

```
[DEC_611] LOCALHOST_NOT_SECURE I/F: 530 Login authentication failed
```

5.3.5 [DEC_612] Cannot Reach Interface Directory

Failed to reach one of the *Directory* of the "DownloadDirs" of a given interface.

5.3.6 [DEC_613] Server Error Message

The information replied by the server when error condition [DEC_612] arises is usually meaningless to understand the root reason to limit the risks of security attacks. It usually is related to file permissions on the server or miss-configuration problems referring to a non existing directory.

```
[DEC_613] LOCALHOST I/F: 550 Can't change directory to /a/if/local/: No such file or directory
```

5.3.7 [DEC_614] Cannot Get URL

Failed to get the URL associated to one *Directory* of the "DownloadDirs" of a given interface when using the HTTP protocol.

5.3.8 [DEC_620] Downloaded File not Copied into Final Dir

The file retrieved from the interface could not be copied into the final directory defined in the configuration. This is usually a problem of file permissions.

5.3.9 [DEC_625] File Dissemination Failure

This error is raised when DEC could not disseminate a file into a given *Intray* according to the *Dissemination rules* using a mv command. This is usually associated to file permission problems of the *Intray* directory.

5.3.10 [DEC_626] File Dissemination Failure

This error is raised when DEC could not disseminate a file into a given *Intray* according to the *Dissemination rules* using hard-links. This is usually associated to file permission problems of the *Intray* directory.

5.4 Debug Messages

6 Reference Commands

6.1 decCheckConfig

```
== Synopsis

This is a command line tool that checks the coherency of the DEC configuration.
DEC configuration is distributed amongst different XML files. The information set up
must be coherent. This tool ensures that all configuration critical elements are correct.
(All DEC config files must be placed in the $DEC_CONFIG directory).
So, run this tool everytime a configuration change is performed.

-e flag:

With this option the Interfaces (Entities) configuration placed in dec_interfaces.xml
is checked. As well it is checked the coherency between the dec_interfaces.xml
configuration file and the DEC Inventory (DEC Database).
(Note: if the network link to a given I/F is broken, the tool will not be able to connect and it
will report a configuration error of this I/F).


-i flag:

With this option the Incoming file-types registered in the dec_incoming_files.xml are checked.
Mainly what it is done is to check that the interfaces from a File is retrieved are configured in
the dec_interfaces.xml file.


-m flag:

With this option the DEC Mail configuration placed in the ft_mail_config.xml is checked.


-s flag:

With this option the DCC Services configured in the dcc_services.xml file are checked.
The check performed with this flag is that the executable set in the command of the service
can be found in the $PATH environment variable.
```

-t flag:

With this option the In-Trays configured in the dec_incoming_files.xml file are checked.

-a flag:

This is the all flag, which performs all the checks described before.

```
== Usage
decCheckConfig [--nodb]
-a    checks all DEC configuration
-e    checks entities configuration in dec_interfaces.xml
--nodb no Inventory checks
-i    checks incoming file-types configured in dec_incoming_files.xml
-o    checks outgoing file-types configured in dec_outgoing_files.xml
-m    checks the mail configuration placed in ft_mail_config.xml
-t    checks the In-Trays configuration placed in dec_incoming_files.xml
-l    checks the log configuration
-h    it shows the help of the tool
-u    it shows the usage of the tool
-v    it shows the version number
-V    it performs the execution in Verbose mode
-D    it performs the execution in Debug mode

== Author
DEIMOS-Space S.L. (bolf)

== Copyright
Copyright (c) 2005 ESA - DEIMOS Space S.L.
```

6.2 decCheckSent

```
== Synopsis

This is a DEC command line tool that lists the contents of the <Upload> directory
corresponding to a given interface configured in interfaces.xml

== Usage
decCheckSent -m <Interface_Name> [-t]
--mnemonic  <MNEMONIC> (mnemonic is case sensitive)
--temp      it shows the content of the <UploadTemp> directory
--Show      it shows all available I/Fs registered in the Inventory
--help      shows this help
--usage     shows the usage
--Debug     shows Debug info during the execution
--version   shows version number

== Author
Deimos-Space S.L. (bolf)

== Copyright
Copyright (c) 2006 ESA - Deimos Space S.L.
```

6.3 decConfigInterface2DB

```
== Synopsis

This is a Data Exchange Component command line tool that synchronizes the Entities configuration
with DEC Inventory. It extracts all the I/Fs from the dec_interfaces.xml file and
inserts them in the DEC Inventory.

As well it allows to specify a new I/F mnemonic to be loaded into the DEC Inventory with
the "--add" command line option.

== Usage
decConfigInterfaceDB --add <MNEMONIC> | --process EXTERNAL
--add <MNEMONIC>     (mnemonic is case sensitive) add the specified Entity
--process EXTERNAL   process $DEC_CONFIG/dec_interfaces.xml
--Show               it shows all I/Fs already loaded in the DCC Inventory
--Verbose            execution in verbose mode
--version            shows version number
--help      shows this help
--usage     shows the usage

== Author
Deimos-Space S.L. (bolf)

== Copyright
Copyright (c) 2006 ESA - Deimos Space S.L.
```

6.4 decDeliverFiles

6.5 decGetFiles4Transfer

```
== Synopsis

This is a DEC command line tool that retrieves files to be transferred (push)
from a source directory specified by $DEC_DELIVERY_ROOT environment variable.

Files present in the source directory are filtered according to the rules
defined in ft_outgoing_files.xml
=> wildcards such as  <File Type="S2A*">
=> children directories, such as <File Type="GIP_PROBA2">

Files gathered are finally placed into the directory defined in the
configuration <GlobalOutbox> present in dec_config.xml

-O flag:
The "ONCE" flag registers in the Inventory all the files sent. As well it checks
prior to the delivery whether a files has been previously sent or not to avoid
delivering it twice to the same Interface.

== Usage
decGetFiles4Transfer [-O] [-l]
--ONCE      The file is just sent once for each I/F
--list      list only
--help      shows this help
--usage     shows the usage
--Debug     shows Debug info during the execution
--version   shows version number

== Author
Deimos-Space S.L. (bolf)

== Copyright
Copyright (c) 2006 ESA - Deimos Space S.L.
```

6.6 decGetFromInterface

```
decGetFromInterface

== Synopsis

This is a DEC command line tool that polls the I/Fs for retrieving
files of registered filetypes. As well It retrieves the I/F
exchange directory file content linked to a time-stamp.

-l flag:

With this option, only "List" of new availables files for Retrieving and Tracking is done.
This flag overrides configuration flags RegisterContentFlag RetrieveContentFlag in dec_interfaces
So Check ONLY of new Files is performed anyway.

-R flag:

With this option (Reporting), DEC Reports will be created (see dcc_config.xml).
Report files are initally placed in the Interface local inbox and
if configured in files2InTrays.xml disseminated as nominal retrieved file.

--del-unknown:

It overrides the dcc_config.xml configuration parameter DeleteUnknown and explicitly
commands for removal of unknown files not configured in ft_incoming_files.xml


== Usage
decGetFromInterface -m <MNEMONIC>  [-l] [--nodb]
--mnemonic  <MNEMONIC> (mnemonic is case sensitive)
--list      list only (not downloading and no ingestion)
--nodb      no Inventory recording
--no-intray skip step of delivery to intrays
--del-unknown it deletes remote files not configured in ft_incoming_files.xmls
--receipt   create only receipt file-list with the content available
--Report    create a Report when new files have been retrieved
--Show      it shows all available I/Fs registered in the DEC Inventory
--help      shows this help
--usage     shows the usage
--Debug     shows Debug info during the execution
--Unknown   shows Unknown files
--Benchmark shows Benchmark info during the execution
--version   shows version number

== Author
DEIMOS-Space S.L. (bolf)
```

6.7 decListener

```
== Synopsis

This is a Data Exchange Component (DEC) command line tool
that manages the I/Fs listeners for data retrieval.

The DEC listeners automates the file pulling from the configured interface.
One listener is devoted for every interface configured.

The behaviour of the listener is driven by the settings defined
in the configuration file $DEC_CONFIG/dec_interfaces.xml

Alternatively the listener settings can be overriden with the command line options.

== Usage
decListener  --all [-R]| --mnemonic <MNEMONIC> --interval <seconds>
--all                 starts a listener for each I/Fs
--Reload              force a Restart of all listeners
--stop <MNEMONIC>     it stops of the listener for the given I/F
--Stop                it stops of all listeners
--check               it checks whether the listeners are running
--mnemonic <MNEMONIC> (mnemonic is case sensitive)
--interval            the frequency it is polled I/F given by MNEMONIC (in seconds)
--nodb                no Inventory recording
--no-intray           skip step of delivery to intrays upon download
--help                shows this help
--Debug               shows Debug info during the execution
--version             shows version number

== Author
DEIMOS-Space S.L. (bolf)

== Copyright
Copyright (c) 2005 ESA - DEIMOS Space S.L.
```

6.8 decManageDB

```
== Usage
decManageDB --create-tables | --drop-tables
--create-tables   create all minarc required tables
--drop-tables     drops all minarc tables
--rpf             selector to include reference planning tables
--Debug           shows Debug info during the execution
--help            shows this help
```

6.9 decSend2Interface

```
== Synopsis

This is a DEC command line tool that deliver files to a given I/F in PUSH mode.
It delivers files using the configured protocols (s)ftp and email.
Files sent can be registered in an Inventory and the delivery date is set to the latest one.

This command can be used in order to send a given file just once
(for each delivery method: ftp, email) for a given Interface.
Use "-O" flag to enable this behaviour.

-R flag:

With this option (Report), a Report "List" with the new files sent is created.
This Report file is initally placed in the Interface local inbox.


== Usage
decSend2Interface -m <MNEMONIC> [-O] [--nodb]
--mnemonic  <MNEMONIC> (mnemonic is case sensitive)
--ONCE      The file is just sent once for that I/F
--AUTO      local outbox Automatic management
--loops <n> n is the number of Loop retries to achieve the Delivery
--delay <s> s seconds of delay between each Loop Retry
[60 secs by default if it is not specified]
--retries <r>  r is the number of retries on each Loop for each file
--Report    create a Report with the list of files delivered to the Interface
--list      list only (not downloading and no ingestion)
--Nomail    avoids mail notification to the I/F after successfully delivery
--Show      it shows all available I/Fs registered in the Inventory
--nodb      no usage of the Inventory for recording operations
--help      shows this help
--usage     shows the usage
--Debug     shows Debug info during the execution
--version   shows version number

== Author
Deimos-Space S.L. (bolf)

== Copyright
Copyright (c) 2006 ESA - Deimos Space S.L.
```

6.10 decSmokeTests

6.11 decStats

```
== Synopsis

This is a DEC/DCC command line tool that shows file reception statistics

== Usage
decStats
--Hours <hours>            status of last n hours
--file <filename>          status of a given filename
--help                     shows this help
--usage                    shows the usage
--Debug                    shows Debug info during the execution
--version                  shows version number


== Author
DEIMOS-Space S.L.

== Copyright
Copyright (c) 2008 ESA - DEIMOS Space S.L.
```

6.12 decUnitTests

6.13 decUnitTests_IERS

6.14 decValidateConfig

```
== Synopsis

This is a DEC command line tool that checks the validity of DEC configuration
files according to DEC's XSD schemas. This tool should be run everytime a
configuration change is performed.

-e flag:

With this option the Interfaces (Entities) configuration file (interfaces.xml)
is validated using the schema interfaces.xsd

-g flag:

With the main DEC configuration file (dec_config.xml)
is validated using the schema dec_config.xsd

-i flag:

With this option the Incoming file-types configuration file (dec_incoming_files.xml)
is validated using the schema dec_incoming_files.xsd

-o flag:

With this option the Outgoing file-types configuration file (dec_outgoing_files.xml)
is validated using the schema dec_outgoing_files.xsd

-m flag:

With this option the DEC Mail configuration file (ft_mail_config.xml) is
validated using the schema ft_mail_config.xsd

-l flag:

With this option the DEC Logs configuration file (dec_log_config.xml) is
validated using the schema dec_log_config.xsd

-a flag:

This is the all flag, which performs all the checks described before.


== Usage
```

```
-a      Check all DEC configuration files
-g      Check DEC’s general configuration file dec_config.xml
-e      Check the Entities Configuration file dec_interfaces.xml
-m      Check the mail configuration file ft_mail_config.xml
-i      Check the incoming file-types configuration file dec_incoming_files.xml
-i      Check the outgoing file-types configuration file dec_outgoing_files.xml
-h      shows this help
-v      shows version number


== Author
DEIMOS-Space S.L. (rell)

== Copyright
Copyright (c) 2007 ESA - DEIMOS Space S.L.
```