SAP Hybris

SAP Hybris Cloud Services

Deployment Packaging Guidelines v2.3.3

CUSTOMER

SAP Run Simple

Table of Contents

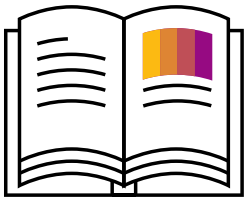
Section 1

Introduction

This document is intended to act as a guideline for hybris Cloud Services (hCS) customers or partners on how to package hybris application releases into deployment packages. The deployment packages are utilized by the hCS Platform DevOps teams for deployment of customer-specific code and configurations into the client's Staging and Production environments. A properly structured deployment package is essential for a correct and successful deployment process – especially automated deployments.

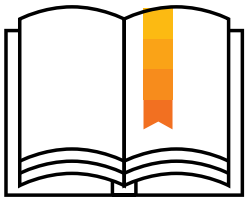
With the new and updated deployment packaging guideline, the deployment of each newly delivered release is independent of any previously delivered release, especially with regards to the configuration folder.

It is assumed that the reader of this document has knowledge of the following:



SOLID KNOWLEDGE...

of the SAP Hybris software platform



GENERAL KNOWLEDGE...

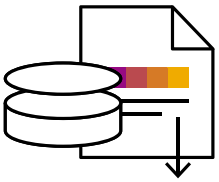
of the Unix/Linux operating system

Section 2

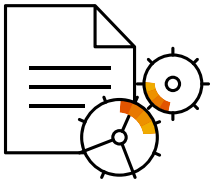
The SAP Hybris Deployment Package

The main items that need to be provided for a deployment performed by hCS are:

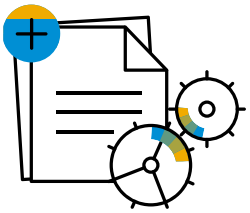
1. SAP Hybris Deployment Package (this document)



Platform and extension zip-files



Configuration files



Optional additional files and configurations

2. SAP Hybris Deployment Request Form

All artifacts should be provided as soon as possible when a deployment is required. hCS will review all documents prior to the deployment. Therefore, they must be attached to the corresponding ticket. If any issue with the provided information is identified, hCS will contact the requester to discuss and resolve before the actual change window for the deployment starts.

The **SAP Hybris Deployment Request Form** (see **Appendix** for link) must be used to specify further details of the deployment, for example if a database change such as an initialization or an update procedure is required. Additional instructions, which are not part of the standard deployment procedure must also be included in the SAP Hybris Deployment Request Form. For more details, please refer to the SAP Hybris Deployment Request Form instructions.

Section 3

Package Naming

Deployment packages must be properly named.

Naming scheme requirements for the hCS deployment package are as follows:

- The name must include the build version of the application release in order to easily identify each of them and their sequence.
- Names must uniquely identify any hCS deployment package.
- Names must follow a consistent and logical sequence so they are sortable and reflect the order in which deployment packages are released.
- Package names must include the major version and if available, minor version information (leading zeroes are recommended).
- Dates should not be included in package names. If including date information is preferred, please limit to including the year before the version number.
- A short identification string (aka project code) of the customer and project name must be added to the package name. This information will be provided by hCS at the beginning of a project.

Below is a list of valid formats for package names, with cid (CID - Customer ID) representing the code for a customer name and pi (PID - Project ID) representing the code for a customer project. This distinction is essential especially for customers with multiple projects.

CID and PI indication is always lowercase. No spaces are allowed, replace them with „_“ (underscore):

cid-pi_v01.02_RC1

cid-pi_01.03.06

cid-pi_build_0124

cid-pi_2014_v03.04

HEC ENVIRONMENTS:

For HEC hosted environments the package naming schema is part of the Hybris@HEC global naming schema: The CID and PID are predetermined since they are already defined for SAP Hybris hostnames. For more details on naming schema see **Appendix**.

Indication of patches or hotfix can be added to the package name:

cid-pi_v01.03

cid-pi_v02.01

cid-pi_v02.01_Patch1

PATCHES AND HOTFIX:

Patch or hotfix indication is for nominal purposes only, for the benefit of customers and partners. All deployment packages must still be provided as a full package. It is not possible to deploy single files.

The following table summarizes the naming requirements:

PACKAGE NAME ATTRIBUTE	REQUIRED	RECOMMENDED	NOT RECOMMENDED	PROHIBITED
Customer ID (CID)	X			
Project ID (PID)	X			
Build major version	X			
Build minor version (if applicable)	X			
Leading zeros in version numbers		X		
Indication of patch/ hotfix		X		
Date information			X	
FORMATTING				
Lower case	X			
Space characters				X

Table 1: Summary of package naming requirements

PACKAGE NAMING SCHEMA EXAMPLES

Customer name: ACME Inc.

- Project name: B2C electric tools shop www.acmetools.com
- CID (Customer ID): acm
- PID (Project ID): ts
- Project code: ACM-TS
- Package naming schema: acm-ts_v##.##
  acm-ts_v01.02
  acm-ts_v01.03
  acm-ts_v02.01
  ...

Customer name: Luthor Corp

- Project name: B2B professional equipment shop.luthor-pro.net
- CID (Customer ID):lth
- PID (Project ID):pr
- Project code: LTH-PR
- Package naming schema: lth-pr_##.##.##
  lth-pr_01.02.02
  lth-pr_01.02.04
  lth-pr_01.03.01
  ...

TIP: Do not use names which are ambiguous or do not follow the package naming schema described before.

EXAMPLES OF POOR PACKAGE NAMING:

$ ls -l /NFS_DATA/deployment

Deployment-April.zip	« No project code, no versioning
urgent-patch-production.zip	« No project code, no versioning
acm-ts_v01.03	« Not compressed
ACME New Deployment.zip	« No versioning, contains spaces
deploy_10-11-2013.zip	« No versioning, no project code, contains date (ambiguous anyway - is this 11th Oct or 10th Nov?)

Please note that the package name will also determine the installation directory of the application in the target environment:

$ ls -l /opt

lrwxrwxrwx	1 root	root	15 Jul 20 13:35	hybris --> hybris_acm-ts_v02.01/
drwxr-xr-x	hybris	hybris 4096	Jul 14 11:50	hybris_acm-ts_v01.02
drwxr-xr-x	hybris	hybris 4096	Jul 18 12:00	hybris_acm-ts_v01.03
drwxr-xr-x	hybris	hybris 4096	Jul 20 10:12	hybris_acm-ts_v02.01

Production vs Non-Production Packages

hCS deployment packages are deployed in Staging before Production. After proper testing in Staging they can be promoted to Production. A package that has not been previously deployed and successfully tested in Staging will not be permitted to be deployed into Production. The reverse however can occur. For example, a release might be deployed in Staging with no intention to promote it to Production. This might be for example to use the Staging environment for debugging purposes which cannot be done in Development.

The following diagram illustrates the possible pipelines and hCS's support case in each.

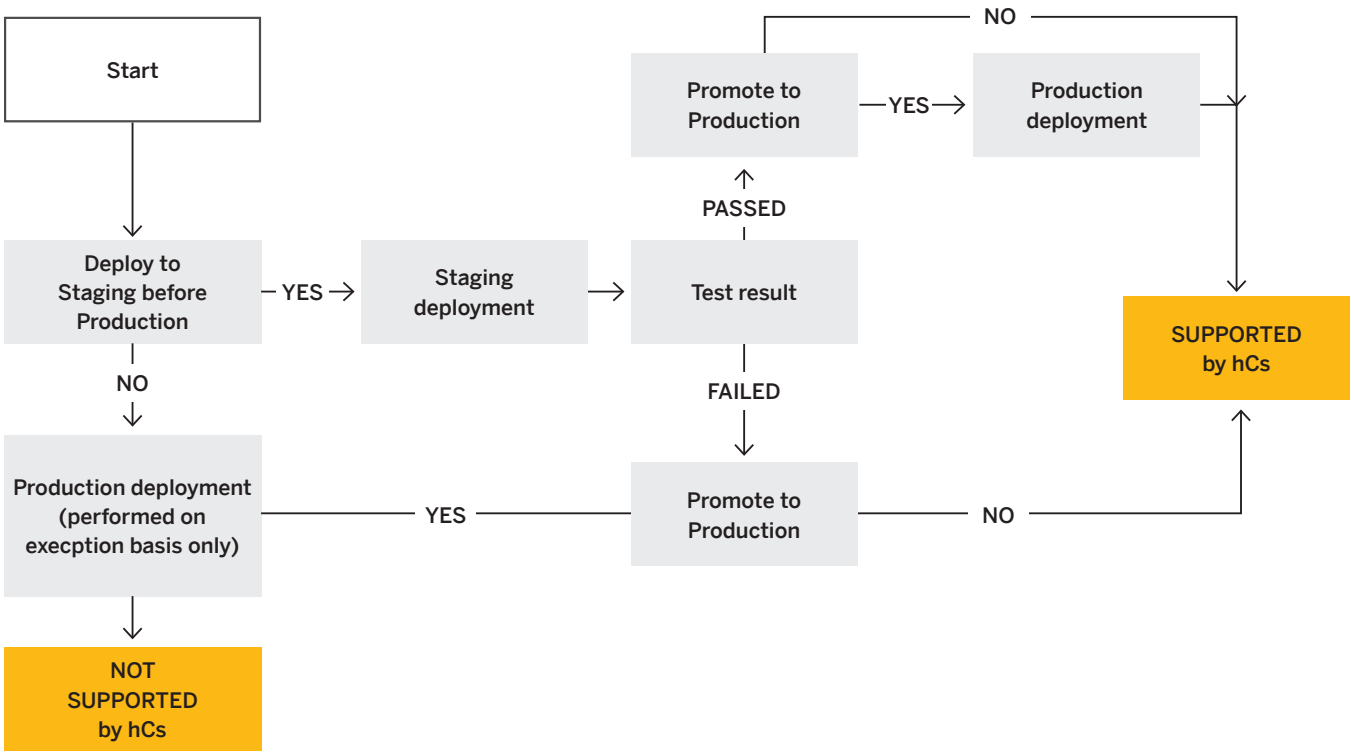


Figure 1: hCS support case flowchart for the possible deployment promotion pipelines

For this reason there is a distinction made between a package intended to be deployed in Production and a package not intended for Production use:

- A package that is NOT intended for Production may contain contents and configuration files specific to only the target environment (for example Staging).
- A package intended for Production must contain contents and configuration files for both Staging and Production.

A package that does not also contain contents and configuration files for the Staging environment cannot be deployed in Production. It is possible to submit a full Production package and deploy in Staging only, and postpone the final decision about a Production deployment until after the result of a Staging deployment.

Section 4

Package Structure

After reviewing this section, you can find examples of deployment packaging in **Section 7.2** of the **Appendix**.

ZIP-PACKAGE

For each deployment package a new zip archive folder must be placed in */NFS_DATA/deployment* on the DEV environment to contain the package files. The zip-file name should match the package naming schema outlined in **Section 2** by matching build version etc. Additionally, spaces and special characters must be replaced with "_" (underscore). For each package, an associated MD5 checksum file must also be provided (see next section).

MD5 CHECKSUM

The checksum file is used during the deployment procedure to test the integrity of the package. It should contain the MD5 hash of the zipped package. On a Linux or OSX machine the following command can be used:

```
# create MD5 file
$ md5sum acm-ts_v01.02.zip
> acm-ts_v01.02.md5
```

```
# read MD5 file
$ cat acm-ts_v01.02.md5
fa027119c1e5f9b89aecee0b-
c02955f3  acm-ts_v01.02.zip
```

```
# check MD5
$ md5sum -c acm-ts_v01.02.md5
acm-ts_v01.02.zip: OK
```

DIRECTORY TREE

The compressed package must contain a specific directory structure: The top level directory should be the name of the deployment package, which in turn will determine the name of the zip-file once compressed. A subdirectory named *hybris* should contain all application related deployment files. Future expansion of the package structure will include additional directories for other server types (apache, solr, etc.).

The following diagram summarizes the directory and file structure of a standard deployment package, and indicates which of them are mandatory for a deployment.

For deployment in more than one environment, a configuration set for each environment has to be provided. Each configuration set will be grouped under the *config* subdirectory, under another directory *<ENV>* named for the target environment (more information follows).

Only one set of binary files (for example *hybrisServer-Platform.zip* and *hybrisServer-AllExtensions.zip*) under the bin parent directory should be provided for deployment packages, regardless of the number of target environments.

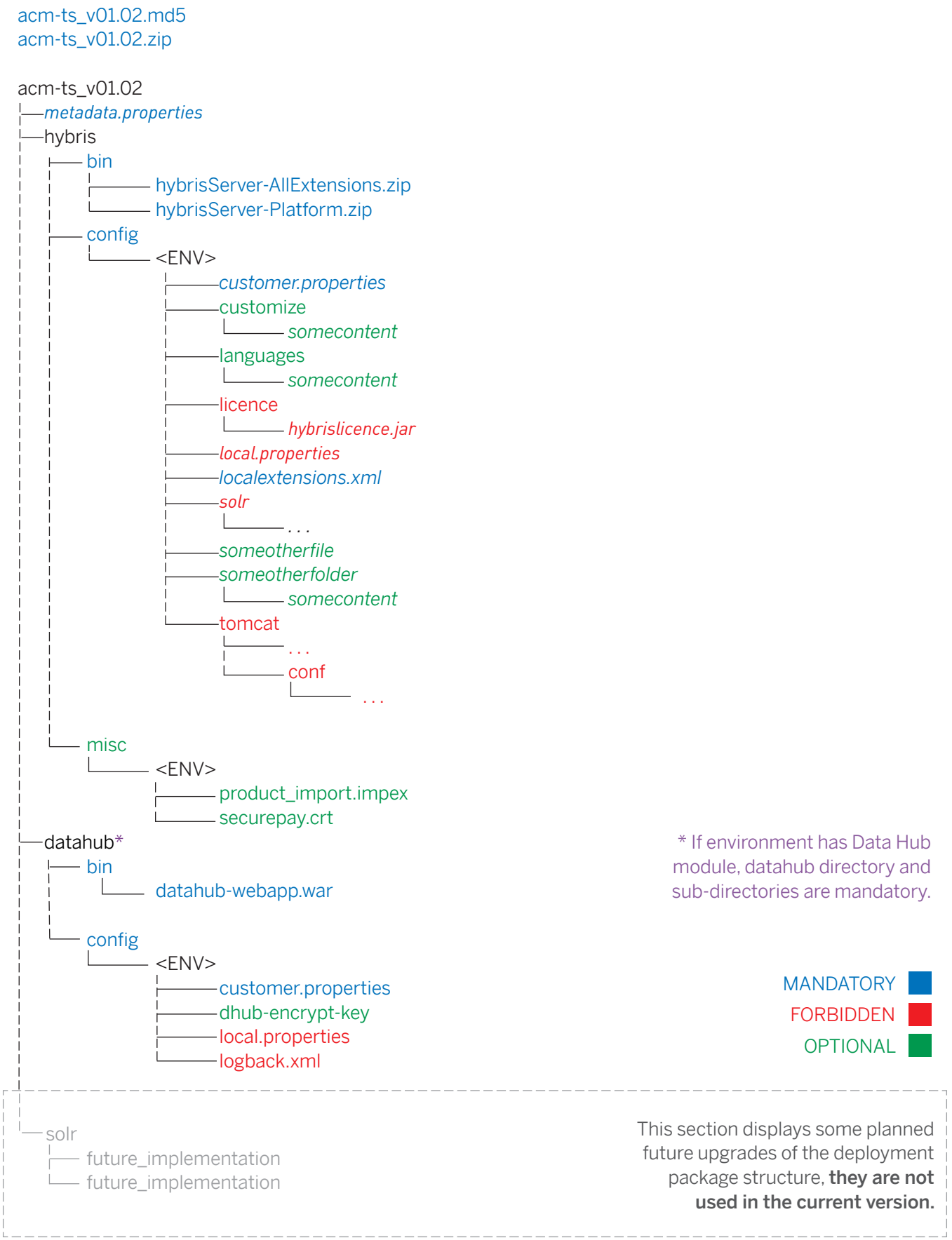


Figure 2: Deployment package structure

The target environment subdirectory **<ENV>** should be named as indicated in the following table:

ENVIRONMENT	HCS CODE	SUBDIRECTORY NAME INSIDE PACKAGE
Development	d	*dev*
QA	q	*qa*
Staging	s	*stag*
Production	p	*prod*

Table 2: Standard environment subdirectory names within deployment package

If there are multiple environments of the same type the subdirectory name of the second, third etc are followed by a number (without zero padding). For example: A second Development environment would be dev2, while the third would be dev3, and so on.

Section 5

Package Contents

5.1 METADATA (MANDATORY)

The file *metadata.properties* contains information required by the hCS automation platform. This at present should contain the version number of the hCS Deployment Packaging Specification. So, per this document it should contain the following property:

metadata.properties

```
package_version = 2.3
datahub_infra =  true
pre-production-env=stag2
```

- datahub_infra = true

  The datahub_infra property is to specify if this environment has data hub infrastructure. This is NOT to specify if a datahub package is included in the deployment package.
  If an environment has data hub infrastructure, it should always have a data hub package and the the datahub_infra property should be set to true.
  Possible values for datahub_infra property are: true & false. If the property is not defined, it will default to false.

- package version

  Please note the package_version property represents the packaging guidelines version and **NOT** the build number of the submitted deployment package.

- pre-production-env

  The pre-production-env property is to specify a pre-production environment code if it is other than "stag". It is only used for a production deployment package. When checking a production package, it also checks the pre-production environment (e.g. stag) according to the environment code defined.

  Possible values for pre-production-env property property are stag[0-9]? If the property is not defined, the default is "stag".

5.2 PLATFORM AND EXTENSIONS ZIP-FILES (MANDATORY)

The Platform and Extensions zip-files should be created on the DEV environment by running the *ant production* command, which will produce the following two files in the folder */opt/hybris/temp/hybris/hybrisServer*:

- *hybrisServer-Platform.zip*
  - */hybris/bin/platform/tomcat(-[0-9])?/ directory must present in the zip file*
    - *Example: /hybris/bin/platform/tomcat/,/hybris/bin/platform/tomcat-1/,etc*
- *hybrisServer-AllExtensions.zip*

These two files must be copied inside the *bin* folder within the deployment package.

CONFIGURATION FILES (MANDATORY)

The following two configuration files are mandatory for every hCS deployment package:

- *localextensions.xml*
- *customer.properties*

The *localextensions.xml* file is the standard configuration file for SAP Hybris extensions, and is required for any customer specific SAP Hybris application.
The *customer.properties* file is used to construct the main SAP Hybris configuration file *local.properties* for the target environment (Staging or Production) during the deployment process. The *local.properties* file is created by combining the content of the *customer.properties*, provided by the partner, with the information from another configuration file, *hybris.properties*, which is managed by hCS.
The *hybris.properties* file contains server configuration settings which are managed by hCS and must not be changed by the hCS customer or partner.

The *hybris.properties* file must NOT be included in the configuration. This file contains all node-specific settings that should not be replaced during the deployment and is already resident on each target node.

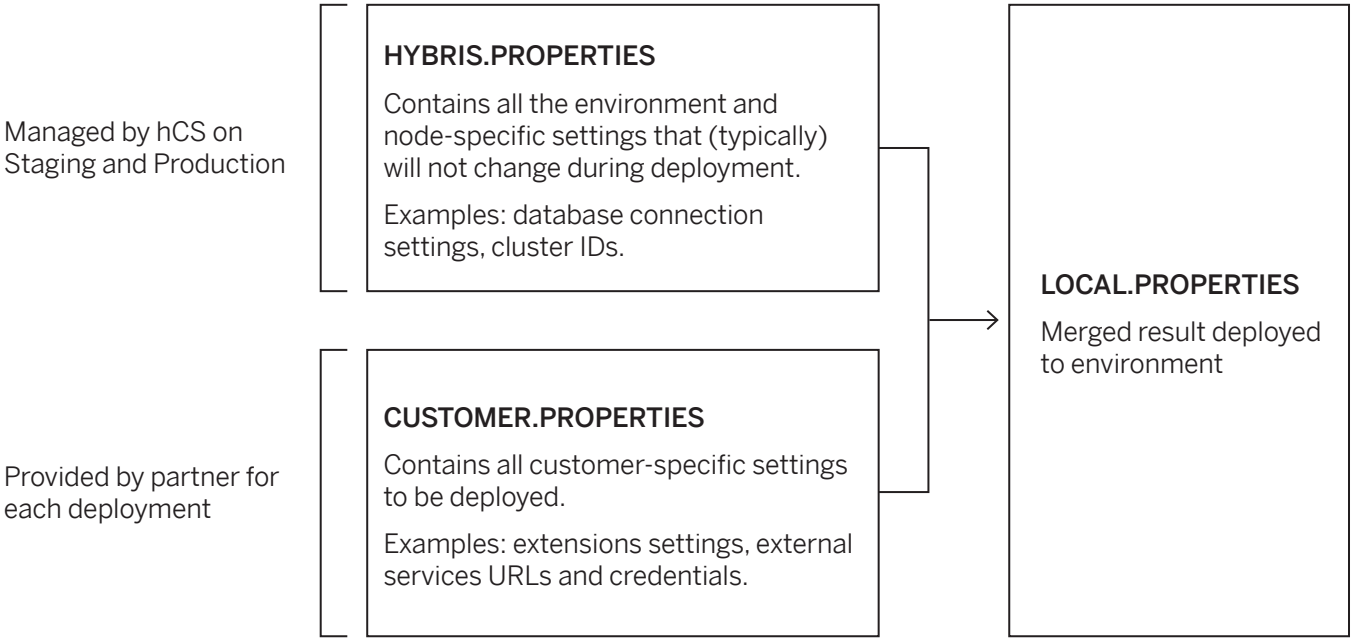


Figure 3: hybris.properties and customer.properties are merged to create final local.properties file

To properly create the customer.properties file, remove all Development environment specific settings and replace them with settings required by the target environments (Staging or Production). Settings must not include node and environment settings that are already included in hybris.properties. In case such a setting is included by error, it will be overwritten during the deployment process and will not take into effect.
A complete list of settings maintained by hCS in the hybris.properties can be found in the Appendix section of this document.

To demonstrate the process of creating the customer.properties file for customers and partners, hCS can provide examples of hybris.properties and customer.properties and the resulting local.properties file for the Development environment.

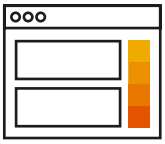
Customers and partners must not simply add the entire *config* folder of the Development environment to the **hCS deployment package**.

Configurations in the Staging and Production environments contain many settings that are critical for the proper functioning of the SAP Hybris application in the whole environment (connector ports, proxy, logging, JMX, clustering, etc.). Replacing the entire *config* folder is likely to render the application not working as designed.

If settings other than the ones described in this section are required, they must be added and properly described in the „Additional Files" section of the **SAP Hybris Deployment Request Form** (see **Appendix** for the link to this documentation).

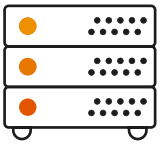
CONFIGURATION PER SERVER ROLE

If the project has a specific requirement to maintain distinct configurations between application servers depending on their role –
for example frontend versus backend/admin servers – this is supported to a limited extent. The partner will need to submit different sets of configurations which will be applied to servers based on their assigned role.



app

for frontend application servers



adm

for backend/admin application server

Role assignment is embedded in the configuration filenames using a specific schema, as follows.

For frontend nodes:

- *customer.app.properties*
- *localextensions.app.xml*

For backend/admin nodes:

- *customer.adm.properties*
- *localextensions.adm.xml*

NB: You must choose between using the **generic names** (e.g. *customer.properties* and *localextensions.xml*) or role-specific ones (e.g. *customer.app.properties* and *localextensions.app.xml*).
The schemes are mutually exclusive. Mixing the two strategies (e.g. *localextensions.adm.xml* with *localextensions.xml*, or localextensions.xml with local.adm.properties) is not allowed.

The resulting structure would thus be like the following:

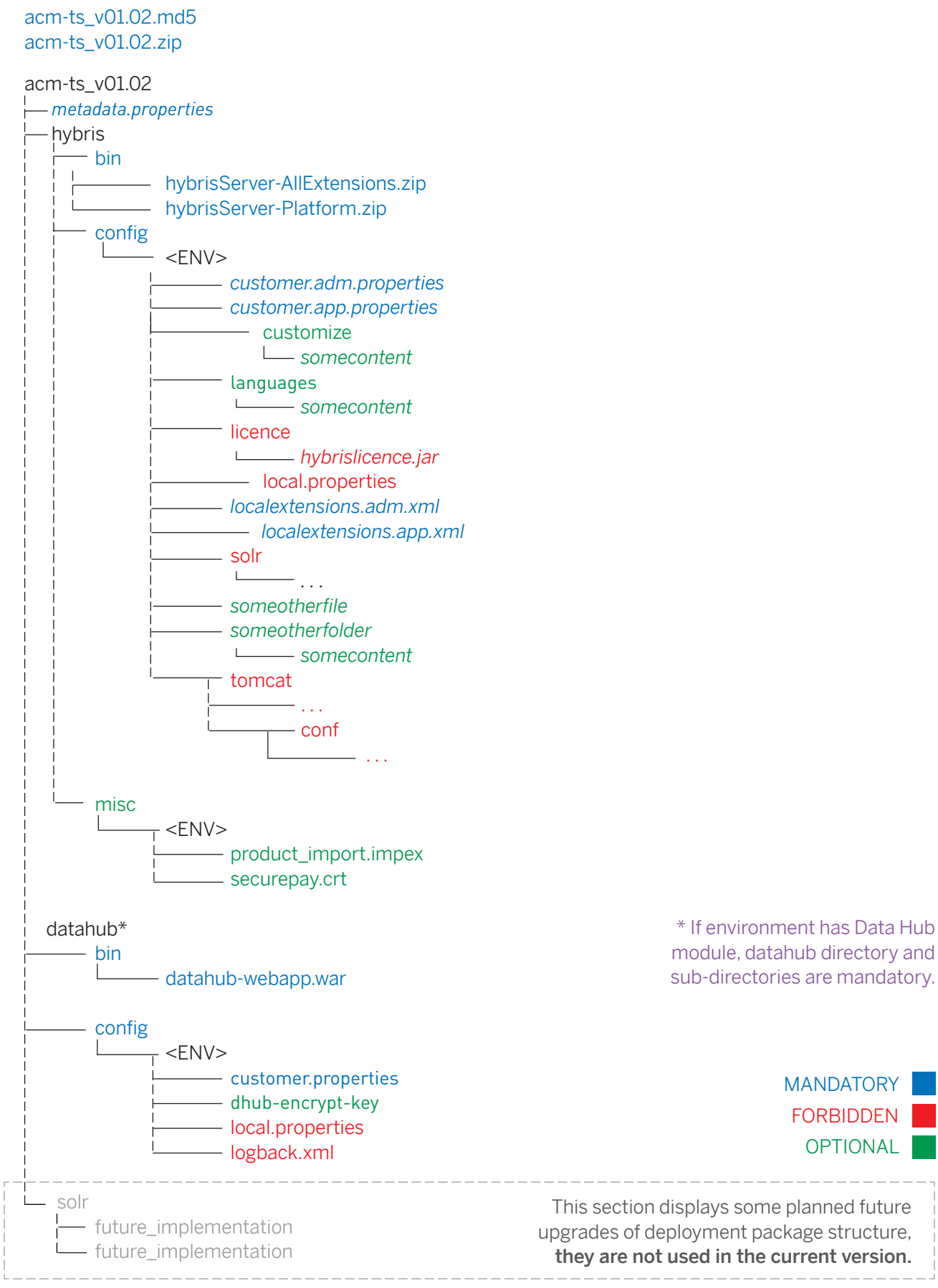


Figure 4: Deployment package structure for a package using role-specific configuration files

CUSTOMIZE AND LANGUAGES FOLDERS (OPTIONAL)

The *customize* folder is used in some installations to add additional custom settings. During the deployment process this folder will be copied inside the *config* folder of the target environment. In this case the build process will be executed using the *ant customize all* command in order to apply the changes.

NB: Please specify in the **SAP Hybris Deployment Request Form** if a *customize* directory exists for any given release. Note that the *customize* directory section only exists in the SAP Hybris Deployment Request Form version 2.9 or higher. Note also that if the *customize* directory is delivered then the *languages* directory also becomes mandatory.

Details about the usage of the *customize* directory can be found on the SAP Hybris platform **Build Framework** documentation, in the Section "Replacing Files in the ${HYBRIS_BIN_DIR} Directory" (see **Appendix** for the link).

ADDITIONAL FILES AND CONFIGURATION FOLDERS (OPTIONAL)

Additional files relevant for a deployment can be included in the hCS deployment package. Examples may be ImpEx files, certificates and keys for external services, or other files to be used during the deployment. It is important to note that such files are not part of the standard deployment process. Therefore clear implementation instructions detailing how and when they should be used must be fully documented in the Section "**Additional Instructions**" of the **SAP Hybris Deployment Request Form**.

Additional files should be saved in the *misc* directory, separated into subdirectories for Staging and Production environments (see examples below as well as the example in **Section 4**).

EXAMPLE: IMPEX FILE

An ImpEx file is provided to import data. In the **SAP Hybris Deployment Request Form**, specify how and when to use the provided file. Instructions provided in the SAP Hybris Deployment Request Form:

Special instructions: After system update import the provided file *product_import.impex* by using the import function of the hAC console.

EXAMPLE: EXTERNAL SERVICE CERTIFICATE

An SSL certificate is provided to allow secure connection to an external payment provider. In the **SAP Hybris Deployment Request Form**, it is clearly specified where the certificate must be placed (use environment variables instead of absolute paths):

Special instructions: Copy the provided certificate file securepay.crt to *${HYBRIS_CONFIG_DIR}/security/certs/securepay.crt*. Then restart the application.

DATA HUB WEBAPP

The Data Hub web app should be provided as a standard Web application ARchive (WAR) file. The web app should be self-contained, ready to work with no external dependencies.

NB: The use of a *context.xml* file to customize the location and context path of the Data Hub WAR file is forbidden.

hCS will organize the web app directory structure in a standard way for all installations. This is because the use of a specific location and context for each customer would make it very difficult to automate, manage and maintain.

DATA HUB WAR FILE (MANDATORY FOR DATA HUB)

The Data Hub WAR file should be created from the DEV environment: *datahub-webapp.war*. The WAR file must be copied inside the *datahub/bin* folder within the deployment package.

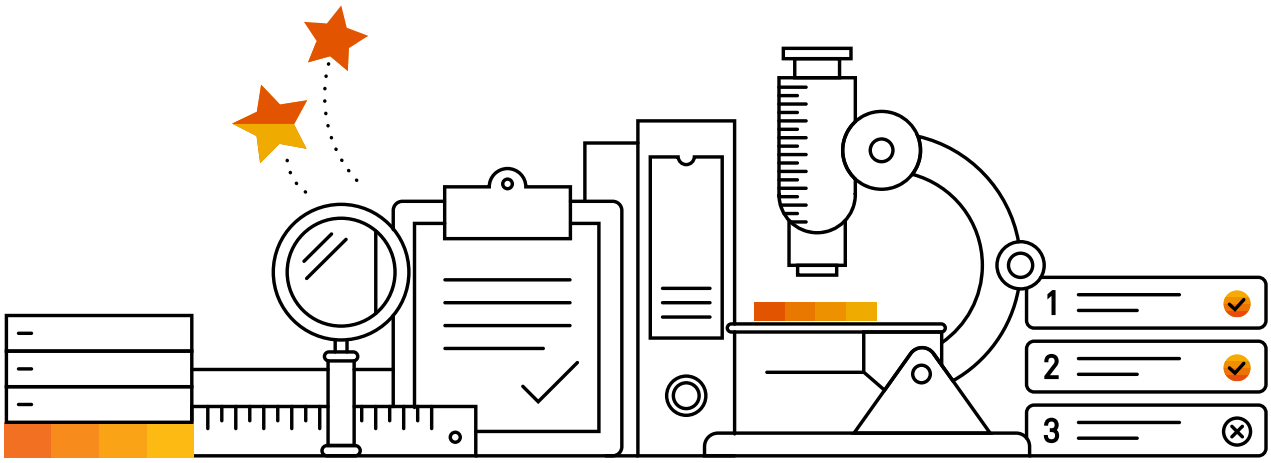
CONFIGURATION FILES (MANDATORY FOR DATA HUB)

The following configuration file is mandatory for every hCS deployment package: *customer.properties.*

This file is used to construct the main SAP Hybris configuration file *local.properties* for the target environment (Staging or Production) during the deployment process. The *local.properties* file is created by combining the content of the customer.properties, provided by the partner, with the information from another configuration file, *hybris.properties*, which is managed by hCS. The *hybris.properties* file contains server configuration settings which are managed by hCS and must not be changed by the hCS customer or partner.

To properly create the *customer.properties* file, all Development environment specific settings need to be removed and replaced with settings required by the target environments (Staging or Production). Settings must not include node and environment settings that are already included in *hybris.properties*.

Special instructions: The *hybris.properties* file must NOT be included in the configuration. This file contains settings that should not be replaced during the deployment.



EXAMPLE: EXPLODED VIEW OF A DATAHUB-WEBAPP.WAR

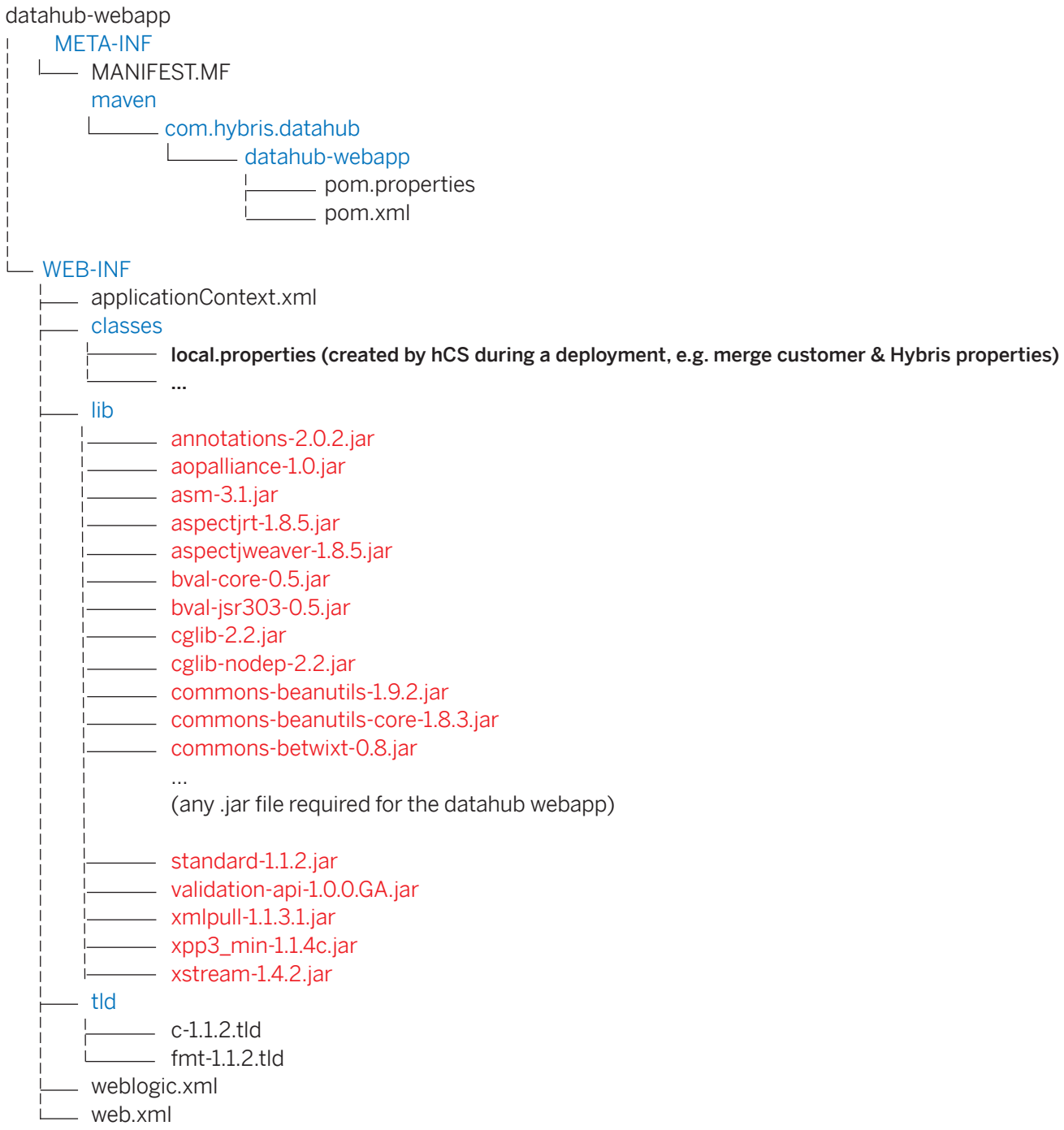
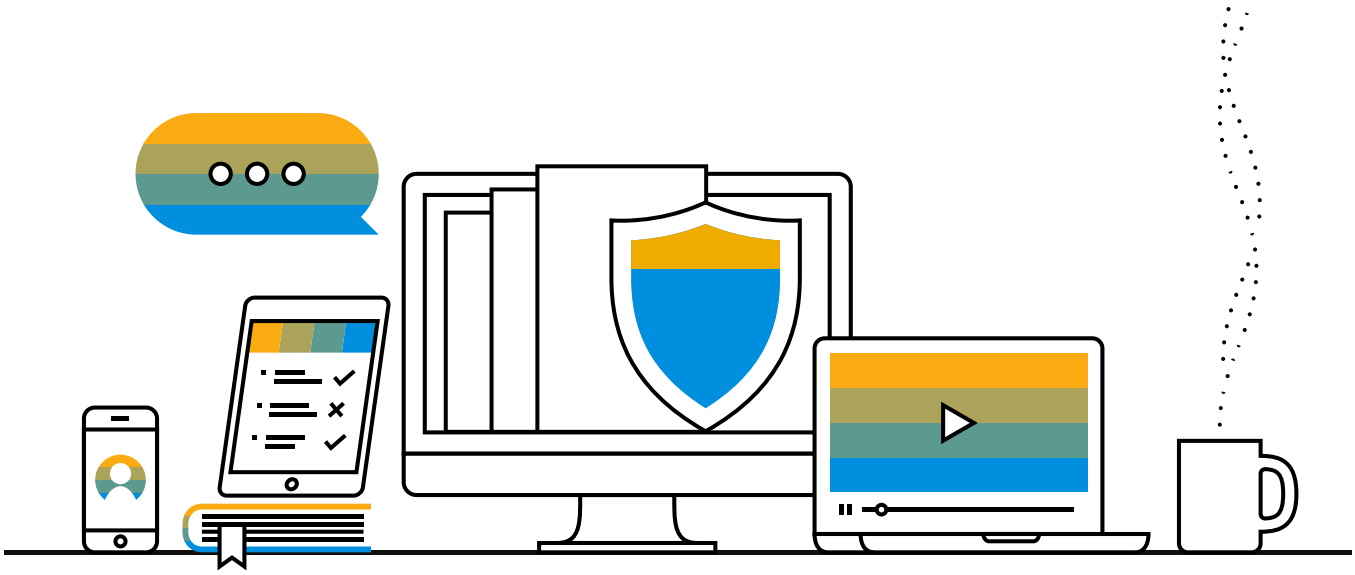


Figure 5: View of a datahub-webapp.war

Section 6

Removed Content

Compared to previous versions of the Deployment Packaging Guidelines, in this version some contents that were previously required have been removed and should not be included anymore in the package. The below describes such contents.



LOCAL.PROPERTIES (FORBIDDEN)

The file *local.properties* must not be part of the deployment package. Instead all project-specific settings are to be included in file *customer.properties* as described in **Section 5**"Configuration Files (Mandatory)".

TOMCAT CONFIGURATION (FORBIDDEN)

The Tomcat configuration is now stored separately on the server and is managed by hCS. The deployment automation does not require that is included in the package and will explicitly ignore it.

Modification to the Tomcat configuration can be requested by raising a separate Change Request.

SAP HYBRIS LICENCE (FORBIDDEN)

The SAP Hybris licence file is now stored separately and managed by hCS, and is not required to be included in the package.

Section 7

Appendix

7.1 LOCAL.PROPERTIES SETTINGS MANAGED BY HCS (HYBRIS.PROPERTIES)

7.1.1 LOCAL.PROPERTIES SETTINGS FOR APP/ADM SERVERS

FUNCTIONAL AREA	PROPERTIES	PARTNER CAN REQUEST ADDITIONS OR MODIFICATIONS?
Generic SAP Hybris Cloud Services Settings	yms* For example: yms.hostname	NO
Database connection	db.url db.driver db.username db.password db.tableprefix - (*defaults*)	NO
Media folder	media.read.dir media.replication.dirs	NO
Clustering	clustermode cluster.id cluster.maxid cluster.broadcast.methods cluster.broadcast.method.udp. multicastaddress cluster.broadcast.method.udp.port	NO
Mail	mail.smtp.server mail.smtp.port mail.pop3.beforesmtp - (*defaults*) mail.use.tls - (*defaults*)	NO
Dynatrace	dynatrace.enabled dynatrace.agentlib dynatrace.name dynatrace.server tomcat.generaloptions.dynatrace	NO
JMX	tomcat.generaloptions.jmxsettings tomcat.jmx.port - (*defaults*) tomcat.jmx.server.port - (*defaults*)	NO
Connector ports	tomcat.http.port tomcat.ssl.port tomcat.ajp.port tomcat.ajp.secureport proxy.http.port proxy.ssl.port	NO
Log level	log4j.threshold	NO
JVM	tomcat.generaloptions java.mem This will include: tomcat.generaloptions.jmxsettings tomcat.generaloptions.jmxsettings tomcat.generaloptions.dynatrace tomcat.generaloptions.GC	Parameters can be added/modified by raising a ServiceNow Change Request
Multi Tenant	installed.tenants	Parameters can be added/modified by raising a ServiceNow Change Request

7.1.2 LOCAL.PROPERTIES SETTINGS FOR DATAHUB

PROPERTY NAME	FUNCTIONALITY	ALLOWED TO OVERWRITE BY PARTNER (DURING EMPLOYMENT)
kernel.autoInitMode	can modify some functionality e.g. do initialization	NO*
dataSource.className		NO*
dataSource.jdbcUrl		NO*
dataSource.username	database connection user name	NO*
dataSource.password	database connection password	NO*
media.dataSource.className		NO*
media.dataSource.jdbcUrl		NO*
media.dataSource.username		NO*
media.dataSource.password		NO*
datahub.server.url		NO*
datahub.extension.exportURL		NO*
datahub.extension.userName		YES*
datahub.extension.password		YES*
targetsystem.hybriscore.url		NO*
targetsystem.hybriscore.username		YES*
targetsystem.hybriscore.password		YES*
datahub.security.basic.admin.user		NO*
datahub.security.basic.admin.password		NO*
datahub.security.basic.read_only.user		NO*
datahub.security.basic.read_only.password		NO*

* YES: Allowed to overwrite by Partner in deployment package
* NO: Not allowed to overwrite by Partner in deployment package, values in customer.properties will be commented out.

EXAMPLE DEPLOYMENT PACKAGES

The following examples illustrate deployment packaging compliant with the above guidelines. The first example is a basic package which provides the minimum, mandatory files required to successfully perform a deployment. The second package is a more complicated package that takes advantage of support for server roles, customize folder, additional environments (stag2) and optionally provided files.

A Basic Package (without data hub infrastructure)

- Note two files for the deployment: A zip-file of the package, and a checksum file.

```
hybris@acm-d-fr-app-001:/NFS_DATA/deployment# ls -l
-rw-r--r-- 1 hybris hybris 52 Aug 21 04:44 acm-ts_v01.02.md5
-rw-r--r-- 1 hybris hybris 12988322 Aug 21 04:40 acm-ts_v01.02.zip
```

- Directory and file structure of acm-ts_v01.02.zip

```
acm-ts_v01.02.zip
└── acm-ts_v01.02
    ├── hybris
    │   ├── bin
    │   │   ├── hybrisServer-AllExtensions.zip
    │   │   └── hybrisServer-Platform.zip
    │   └── config
    │       ├── prod
    │       │   ├── customer.properties
    │       │   └── localextensions.xml
    │       └── stag
    │           ├── customer.properties
    │           └── localextensions.xml
    └── metadata.properties
```

A More Complex Package

- Note two files for the deployment: A zip-file of the package, and a checksum file.

```
hybris@acm-d-fr-app-001:/NFS_DATA/deployment# ls -l
-rw-r--r-- 1 hybris hybris 73 Aug 21 04:44 acm-ts_v02.00.md5
-rw-r--r-- 1 hybris hybris 22988322 Aug 21 04:40 acm-ts_v02.00.zip
```

- Directory and file structure of acm-ts_v02.00.zip

```
acm-ts_v02.00.zip
└── acm-ts_v02.00
    ├── hybris
    │   ├── bin
    │   │   ├── hybrisServer-AllExtensions.zip
    │   │   └── hybrisServer-Platform.zip
    │   └── config
    │       ├── prod
    │       │   ├── acme_extraconf
    │       │   │   └── acme_sum.conf
    │       │   ├── acme_paygway.xml
    │       │   ├── customer.adm.properties
    │       │   ├── customer.app.properties
    │       │   └── customize
    │       │       └── platform
    │       │           └── ext
    │       │               └── core
    │       │                   └── resources
    │       │                       └── ehcache.xml
    │       │   ├── languages
    │       │   ├── localextensions.adm.xml
    │       │   └── localextensions.app.xml
    │       ├── stag
    │       │   ├── acme_extraconf
    │       │   │   └── acme_sum.conf
    │       │   ├── acme_paygway.xml
    │       │   ├── customer.adm.properties
    │       │   ├── customer.app.properties
    │       │   └── customize
    │       │       └── platform
    │       │           └── ext
    │       │               └── core
    │       │                   └── resources
    │       │                       └── ehcache.xml
    │       │   ├── languages
    │       │   ├── localextensions.adm.xml
    │       │   └── localextensions.app.xml
    │       └── stag2
    │           ├── acme_extraconf
    │           │   └── acme_sum.conf
    │           ├── acme_paygway.xml
    │           ├── customer.adm.properties
    │           ├── customer.app.properties
    │           └── customize
    │               └── platform
    │                   └── ext
    │                       └── core
    │                           └── resources
    │                               └── ehcache.xml
    │           ├── languages
    │           ├── localextensions.adm.xml
    │           └── localextensions.app.xml
    └── metadata.properties
```

Links

Here are links to resources mentioned throughout the document.

SAP Hybris Deployment Request Form

https://wiki.hybris.com/display/SUP/Deployment+Request+Form+v2.11_Frankfurt+and+Boston+Datacenters

SAP Hybris Platform Build Framework documentation

https://wiki.hybris.com/display/release5/Build+Framework

(see Section “Replacing Files in the ${HYBRIS_BIN_DIR} Directory”)

SAP Hybris

www.hybris.com

SAP Run Simple