Configuration file webdesk.properties and environment variables

The file webdesk.properties is the startup configuration of Webdesk 4. It contains the connection data for the Webdesk and Shark databases, the settings for encrypting these credentials and a few switches that control how the system behaves at startup. This page is intended for system administrators and consultants who install a system, move it to containers or analyze startup problems. Functional settings (clients, jobs, authorizations, etc.) are not made here but in the application itself, see e.g. System parameters Portal & Organization[1].

Selected keys of the file can additionally be overridden by JVM parameters (-Dkey=value) or environment variables (WD_...). This is the recommended method for container operation, because then no credentials need to be stored in the file.

Location and structure

- Webdesk looks for the file on the classpath of the web application. In a standard installation it is therefore located at WEB-INF/classes/webdesk.properties of the Webdesk application in the servlet container (Windows installer: <Webdesk-Webapp>\WEB-INF\classes\webdesk.properties).
- The file is a Java properties file: one entry per line in the form key = value; comment lines start with #. The file is read at startup; changes only take effect after a restart.
- Use the file shipped with the installer or the template webdesk.properties.template of the web application as a starting point. Customer examples can be found in the source code under webdesk-webapp/src/test/resources/customerprops.
- Webdesk does not start without this file (error "Error while trying to load webdesk.properties"), unless the embedded database is enabled (see webdesk.useEmbeddedDatabase). Even when operating with environment variables, at least a small file must therefore exist.
- Webdesk itself writes to the file at startup (encrypting the credentials, the key, adjusting outdated entries, resetting webdesk.runRegistration). The user running the container therefore needs write permission, and in container operation the file must reside on persistent storage. Create a backup before making manual changes.

Minimal example

```
webdesk.runRegistration = 1

webdesk.connection.driver_class = org.mariadb.jdbc.Driver
webdesk.connection.url = jdbc:mysql://<db-host>/webdesk
webdesk.dialect = org.hibernate.dialect.MySQLDialect
webdesk.connection.username = <db-user>
webdesk.connection.password = <db-password>

shark.connection.driver_class = org.mariadb.jdbc.Driver
shark.connection.url = jdbc:mysql://<db-host>/shark
shark.connection.username = <db-user>
shark.connection.password = <db-password>

webdesk.ta.if6020impl.rmi = rmi://<ta-host>:4009/taRmiBroker
```

At the first start Webdesk encrypts the user names and passwords in the file, generates (or reads) the encryption key and itself adds webdesk.credentials.encrypted, shark.credentials.encrypted, webdesk.encryption.key and webdesk.encryption.key.source (see "Encryption of the credentials").

Automatic adjustments at startup

At startup Webdesk corrects the following outdated entries in the file and then saves it again:

- The old keys hibernate.connection.driver_class, hibernate.connection.url, hibernate.connection.username, hibernate.connection.password, hibernate.dialect are renamed to webdesk.connection.* and webdesk.dialect (provided the new key does not exist yet).
- webdesk-SessionFactory.executeIndexManager is renamed to PoStartup.executeIndexManager.
- The driver com.mysql.jdbc.Driver is replaced by org.mariadb.jdbc.Driver (the MySQL driver is not shipped).
- For JDBC URLs starting with jdbc:mysql, the parameters permitMysqlScheme and allowMultiQueries=true are added.
- For SQL Server JDBC URLs (jdbc:sqlserver) without an encrypt parameter, ;encrypt=false is appended; the dialect org.hibernate.dialect.SQLServerDialect becomes org.hibernate.dialect.SQLServer2012Dialect.
- Obsolete cache replication entries are removed or renamed (webdesk.cacheReplication.peerProvider.properties becomes webdesk.cacheReplication.RMI.peerProvider.properties).

Note: These corrections only affect the file. Values supplied by environment variables or JVM parameters are not corrected and must already use the current notation (e.g. the driver org.mariadb.jdbc.Driver).

Database connection

Property	Description	Possible value(s)
webdesk.connection.driver_class	JDBC driver class of the Webdesk database.	e.g. org.mariadb.jdbc.Driver (MySQL/MariaDB), com.microsoft.sqlserver.jdbc.SQLServerDriver (SQL Server), oracle.jdbc.OracleDriver (Oracle) (template: com.mysql.jdbc.Driver, replaced at startup)
webdesk.connection.url	JDBC URL of the Webdesk database.	e.g. jdbc:mysql://<db-host>/webdesk or jdbc:sqlserver://<db-host>:1433;databaseName=webdesk (template: jdbc:mysql://localhost/webdesk)
webdesk.dialect	Hibernate dialect matching the database.	e.g. org.hibernate.dialect.MySQLDialect, org.hibernate.dialect.SQLServer2012Dialect, org.hibernate.dialect.OracleDialect (template: org.hibernate.dialect.MySQLDialect)
webdesk.connection.username webdesk.connection.password	Credentials of the Webdesk database. If they are encrypted, Base64 texts are stored here (see below).	text (template: plain-text sample values)
webdesk.credentials.encrypted	Specifies whether user name and password of the Webdesk database are stored encrypted in the file. If the entry is missing, Webdesk encrypts the values at startup and sets the entry to true. This entry cannot be overridden by an environment variable.	true, false (default if the entry is missing: Webdesk encrypts and sets true)

shark.connection.driver_class shark.connection.url	Driver class and JDBC URL of the Shark database (workflow engine). The same notes as above apply.	(template: com.mysql.jdbc.Driver, jdbc:mysql://localhost/shark)
shark.connection.username shark.connection.password	Credentials of the Shark database. If the entries are missing (system without workflow module), they are assumed to be empty.	text
shark.credentials.encrypted	Like webdesk.credentials.encrypted, for the Shark database. If the entry is missing and a Shark user name exists in the file, Webdesk encrypts at startup; if the user name is missing too, false applies.	true, false
webdesk-sessionFactory.schemaUpdate	Specifies whether the database schema is adjusted automatically at startup when the data model contains new columns or tables. With false the schema is not checked (shorter startup, but not suitable for production systems while schema changes are pending after an update).	true, false (default: true)
webdesk.exitJvmOnConnectionError	With true the JVM is terminated (exit code 1) if no database connection can be established at startup and the error message contains "Access denied" or "Communications link failure". Useful so that a container orchestrator restarts the container. The value is read from the file webdesk.properties; please set it there.	true, false (default: false)
webdesk.useEmbeddedDatabase	With true, Webdesk starts with an embedded HSQLDB instead of the configured databases; the connection data in the file is ignored. The database is located in WEB-INF/work/db (if present), otherwise in the JVM temp directory under db, and is preserved across restarts. Intended for demo and development purposes. A fixed test key is used instead of a secret key; do not use it for production systems.	true, false (default: false)

Encryption of the credentials

Webdesk encrypts the credentials in webdesk.properties with a key (AES-256). The same key is also used for other passwords stored in the application, for example in connectors (database, REST and Notes connectors), the document management (repositories) and the groupware access (Graph/Exchange). For more about the change of the encryption see New Features in 4.56[2].

Warning: If the key is lost, the encrypted credentials in webdesk.properties and the encrypted passwords stored in the database can no longer be decrypted. Webdesk then can no longer be started or the connections stop working. The key (enc.key or the entry webdesk.encryption.key) must therefore be backed up separately, independently of the database backup. A key that exists only in a container instance without persistent storage is lost when the container is replaced.

Property	Description	Possible value(s)
webdesk.encryption.key.source	Specifies where the key is read from. webdesk_properties: the key is stored Base64-encoded in the entry webdesk.encryption.key of the same file. file: the key is stored in the file enc.key (Base64 text only, first line), located on the classpath in the same directory as webdesk.properties. webservice is not implemented in the code. If the entry is missing, Webdesk looks for the key in the database (table PoFile, FILE_UID = 'encryptionKey'), creates a new one if necessary and writes it into the file with the source webdesk_properties.	file, webdesk_properties (default: not set, see description)
webdesk.encryption.key	The key (Base64) if the source is webdesk_properties. Generated and entered by Webdesk itself. Do not change it manually: any other value makes the existing encrypted data unreadable.	Base64 text (generated by the system)

Moving the key to a separate file

We recommend moving the key out of the configuration file into a separate file enc.key after the first successful start and backing it up separately:

1. Create a backup of webdesk.properties and of the databases.
2. Write the value of the entry webdesk.encryption.key as the only line into a new file enc.key and place it in the same directory as webdesk.properties.
3. In webdesk.properties set the entry webdesk.encryption.key.source = file and remove the entry webdesk.encryption.key.
4. Restart Webdesk and verify that the system starts.
5. Back up enc.key separately from the application.

Behavior in a cluster

All nodes must work with the same key. If the entry webdesk.encryption.key.source is missing on a node, it reads the key from the database (PoFile, FILE_UID = 'encryptionKey'); when creating it, the table is locked so that only one key is created. If the key is moved to enc.key, the file must exist with identical content on every node. The database entry is then no longer used and can (according to the note in New Features in 4.56[3]) be removed with DELETE FROM PoFile WHERE FILE_UID='encryptionKey';, but only after all nodes have started successfully with enc.key. For further cluster settings see the page "Cluster- und Cache-Betrieb".

Command line tool CredentialsEncryptor

The tool webdesk-encryptor can encrypt, display and reset credentials in a webdesk.properties without starting the system. On Windows installations the script manageWebdeskDBProps.bat in the Tomcat bin directory is provided for this; it finds the file and the program file webdesk-encryptor-*.jar itself and lets you choose between "Basic" and "Expert" mode. Direct call:

```
java -cp webdesk-encryptor-<version>.jar at.workflow.webdesk.tools.CredentialsEncryptor
 path=<path>/webdesk.properties [webdeskLogin=<user>] [webdeskPassword=<pw>]
 [sharkLogin=<user>] [sharkPassword=<pw>]
```

- Without path, the webdesk.properties in the current working directory is used.
- The parameters webdeskLogin, webdeskPassword, sharkLogin, sharkPassword set the respective user name or password anew (given in plain text, the tool encrypts them).
- If the credentials are not encrypted yet, they are encrypted and webdesk.credentials.encrypted / shark.credentials.encrypted are set; values using the old encryption scheme are re-encrypted.
- Afterwards the tool shows a menu: display values decrypted, display encrypted values, re-enter all four values, quit.
- If the file does not yet contain a key source, the tool generates a new key and enters it with the source webdesk_properties. The tool only supports this source; with webdesk.encryption.key.source = file it aborts with an error message.

Warning: If the tool is applied to a file that belongs to an already running system but contains no key, a new key is created that differs from the one used by the system. Use the tool only with the original file of the system including its key.

Startup behavior

Property	Description	Possible value(s)
webdesk.runRegistration	Specifies whether actions, texts (text modules) and other artifacts of the modules are registered in the database at startup. 0: never. 1: once, afterwards Webdesk resets the entry in the file to 0. 2: completely at every start. 3: at every start, but without text modules. 4: like 2, afterwards Webdesk exits (only for automated deployments and updates with Kubernetes). An invalid value acts like 1. If the database is still empty, the registration is forced	0, 1, 2, 3, 4 (default if the entry is missing: 1; template: 1)

	once and completed before the system is used.	
webdesk.runRegistrationInThread	With false the registration runs in the foreground, i.e. startup takes longer until it is finished. By default it runs in a background thread.	true, false (default: true)
PoStartup.executeIndexManager	Whether at startup all indexes of the Webdesk database are compared with the expected ones and missing ones are created. Can lengthen startup considerably. For the Shark database WfStartup.executeIndexManager applies analogously (default: true). The template still uses the old name webdesk-SessionFactory.executeIndexManager, which is renamed at startup.	true, false (default: true; template: false)
webdesk.disableTimeLineChecks	With true, the consistency check of the persons' timelines (historization) at startup is skipped. The check can take more than 30 seconds; disabling it forgoes the safety check for inconsistencies in the master data.	true, false (default: false; template: true)
webdesk.luceneIndexing	Whether the Lucene search index is refreshed at startup. The index is stored in the cache directory of the application under lucene/<subdirectory>. The subdirectory is a random number by default, but can be fixed with the JVM parameter -Dwebdesk.lucene.index.subdir=<name> (e.g. when several instances use the same cache path).	true, false (default: true)
webdesk.localLogConfig	With true, the node ignores requests to reload the log configuration (log4j2.xml) from the database. Changes made in the action po_editLogSettings ("Log-Settings") on this node are then neither saved to the database nor distributed to other cluster nodes. Typical use: a debug node.	true, false (default: false)
webdesk.collectorHostname.regex	Regular expression for the host names allowed to use the JavaMelody collector. If set and the Tomcat filter RemoteHostFilter is available,	regular expression (default: not set)

	Webdesk configures the filter with this expression at startup.	

TA connector (IF6020)

Property	Description	Possible value(s)
webdesk.ta.if6020impl.rmi	RMI address of the TA connector (broker) for the IF6020 time and attendance integration. If several connectors are available, separate the addresses with a semicolon. If the entry is missing, it is assumed to be empty.	rmi://<host>:<port>/taRmiBroker, several separated by ";" (template: rmi://localhost:4009/taRmiBroker)
webdesk.ta.if6020impl.rmi.timeoutMillis	Timeout in milliseconds for the connection handshake with the connector.	number (default: 2000)
webdesk.ta.if6020impl.rmi.connectTimeoutMillis	Timeout in milliseconds for establishing the TCP connection to the connector.	number (default: 3000)

Example with two connectors:

```
webdesk.ta.if6020impl.rmi = rmi://<ta-host1>:4009/taRmiBroker;rmi://<ta-host2>:4012/
taRmiBroker
```

Overriding bean properties

Every entry of the form <beanId>.<property> = value sets, at startup, the property of the Spring bean with that id (bean names and properties correspond to the names in the "Systemparameter" mask, see System parameters Portal & Organization[4]). This is a Spring "PropertyOverrideConfigurer" that reads webdesk.properties.

Examples:

```
PoOptions.clusterNode = node1
PoHelpOptions.serviceDeskEmail = <service-desk-address>
PoHelpOptions.errorReportEnabled = true
```

Warning: Entries with an unknown bean id or an unknown property are ignored without an error message. A typo therefore goes unnoticed; after the restart, check in the application whether the value is effective. Only the keys listed in the table below can be overridden by environment variable or JVM parameter, including PoOptions.clusterNode, WfStartup.engineName and SecurityHeadersOptions.forceSameSiteNoneCookieAttribute; all other bean entries must be in the file.

JVM parameters and environment variables

Certain keys can be overridden independently of the file. For these keys the following precedence applies (highest first):

1. Environment variable WD_...
2. JVM parameter -Dkey=value (e.g. in CATALINA_OPTS / JAVA_OPTS)
3. Value in webdesk.properties

Naming rule

The name of the environment variable is derived from the key name as follows: convert all letters to upper case, replace every dot with an underscore and prepend WD_. Other characters (e.g. underscores) remain unchanged. Examples:

- webdesk.connection.driver_class becomes WD_WEBDESK_CONNECTION_DRIVER_CLASS
- webdesk.ta.if6020impl.rmi becomes WD_WEBDESK_TA_IF6020IMPL_RMI
- PoOptions.clusterNode becomes WD_POOPTIONS_CLUSTERNODE

An empty environment variable or one consisting only of whitespace is ignored (the value from the file then continues to apply). Keys that are not in the list cannot be set by environment variable, even if a suitably named variable exists.

Overridable keys

Key	Environment variable	Remark
webdesk.connection.driver_class	WD_WEBDESK_CONNECTION_DRIVER_CLASS	Webdesk database
webdesk.connection.url	WD_WEBDESK_CONNECTION_URL	Webdesk database
webdesk.connection.username	WD_WEBDESK_CONNECTION_USERNAME	Webdesk database
webdesk.connection.password	WD_WEBDESK_CONNECTION_PASSWORD	Webdesk database
webdesk.dialect	WD_WEBDESK_DIALECT	Hibernate dialect
shark.connection.driver_class	WD_SHARK_CONNECTION_DRIVER_CLASS	Shark database
shark.connection.url	WD_SHARK_CONNECTION_URL	Shark database
shark.connection.username	WD_SHARK_CONNECTION_USERNAME	Shark database
shark.connection.password	WD_SHARK_CONNECTION_PASSWORD	Shark database
webdesk.ta.if6020impl.rmi	WD_WEBDESK_TA_IF6020IMPL_RMI	TA connector(s), several separated by ";"
webdesk.runRegistration	WD_WEBDESK_RUNREGISTRATION	registration at startup
webdesk.disableTimeLineChecks	WD_WEBDESK_DISABLETIMELINECHECKS	timeline check
webdesk.useEmbeddedDatabase	WD_WEBDESK_USEEMBEDDEDDATABASE	embedded database
webdesk.exitJvmOnConnectionError	WD_WEBDESK_EXITJVMONCONNECTIONERROR	connection error check reads the value from webdesk.properties; set it there
PoOptions.clusterNode	WD_POOPTIONS_CLUSTERNODE	node name (cluster)
WfStartup.engineName	WD_WFSTARTUP_ENGINENAME	name of the workflow engine (default: webdeskshark)
SecurityHeadersOptions.forceSameSiteNoneCookieAttribute	WD_SECURITYHEADERSOPTIONS_FORCESAMESITENONECOOKIEATTRIBUTE	force cookie attribute SameSite=None
isDistributed	WD_ISDISTRIBUTED	cluster, see below
useJmsReplication	WD_USEJMSREPLICATION	cluster, see below
webdesk.cacheReplication.RMI.peerProviderProperties	WD_WEBDESK_CACHEREPLICATION_RMI_PEERPROVIDER_PROPERTIES	cluster, see below
webdesk.cacheReplication.JMS.peerProviderProperties	WD_WEBDESK_CACHEREPLICATION_JMS_PEERPROVIDER_PROPERTIES	cluster, see below
redis.useAsSessionStore redis.connection.url	WD_REDIS_USEASSESSIONSTORE WD_REDIS_CONNECTION_URL	Redis as session store, see below. The key redis.credentials.encrpted

redis.connection.password	WD_REDIS_CONNECTION_PASSWORD	Yearly spelled this way (without the "y").
redis.clusterNodes	WD_REDIS_CLUSTERNODES	
redis.session.cookie.jvmRoute	WD_REDIS_SESSION_COOKIE_JVMROUTE	
redis.credentials.encrpted	WD_REDIS_CREDENTIALS_ENCRPTED	

Example for container operation

In the file webdesk.properties inside the container, one entry is enough that states that the credentials supplied via the environment are not encrypted (this entry cannot be set by environment variable). Without it, Webdesk tries at startup to encrypt the (missing) values of the file and aborts with the message that webdesk.connection.username in webdesk.properties has no value:

```
webdesk.credentials.encrypted = false
webdesk.runRegistration = 1
```

The environment variables (shown as a list, as in a Docker Compose file):

```
WD_WEBDESK_CONNECTION_DRIVER_CLASS=org.mariadb.jdbc.Driver
WD_WEBDESK_CONNECTION_URL=jdbc:mysql://<db-host>/webdesk?
permitMysqlScheme&allowMultiQueries=true
WD_WEBDESK_CONNECTION_USERNAME=<db-user>
WD_WEBDESK_CONNECTION_PASSWORD=<db-password>
WD_WEBDESK_DIALECT=org.hibernate.dialect.MySQLDialect
WD_SHARK_CONNECTION_DRIVER_CLASS=org.mariadb.jdbc.Driver
WD_SHARK_CONNECTION_URL=jdbc:mysql://<db-host>/shark?
permitMysqlScheme&allowMultiQueries=true
WD_SHARK_CONNECTION_USERNAME=<db-user>
WD_SHARK_CONNECTION_PASSWORD=<db-password>
WD_WEBDESK_TA_IF6020IMPL_RMI=rmi://<ta-host>:4009/taRmiBroker
WD_WEBDESK_EXITJVMONCONNECTIONERROR=true
```

Note: If you want to use encrypted credentials in the container, set webdesk.credentials.encrypted = true in the file and supply in the variables the Base64 values encrypted with the system key. Because WD_WEBDESK_EXITJVMONCONNECTIONERROR does not reach the check (see table), enter webdesk.exitJvmOnConnectionError = true in the file if the container is to exit on connection errors.

Cluster and Redis (overview)

For operation with several nodes, webdesk.properties knows, among others, the following keys; for their meaning and interplay see the page "Cluster- und Cache-Betrieb".

Property	Description	Possible value(s)
isDistributed	Enables cluster operation (cache replication).	true, false (default: false)
useJmsReplication	Cache replication via JMS instead of RMI.	true, false
PoOptions.clusterNode	Name of the current node. If isDistributed is set to true via environment/JVM and no node name is given, Webdesk uses the value of the environment variable HOSTNAME. In a cluster Webdesk appends the node name to the name of the workflow engine (webdeskshark_<node>) as long as WfStartup.engineName has not been changed.	text, e.g. node1

webdesk.cacheReplication.RMI.peerProvider.properties webdesk.cacheReplication.peerListener.properties webdesk.cacheReplication.JMS.peerProvider.properties	Parameters of the cache replication (RMI or JMS).	examples: see template
redis.*	Redis as session store (redis.useAsSessionStore, redis.connection.url, redis.connection.password, redis.clusterNodes, redis.session.cookie.jvmRoute).	see page "Cluster- und Cache-Betrieb"

Troubleshooting

- **Webdesk does not start, "Error while trying to load webdesk.properties":** the file is missing on the classpath (WEB-INF/classes).
- **Startup aborts with "... in webdesk.properties has no value":** the credentials are not in the file, but webdesk.credentials.encrypted is missing (see the container example).
- **Decryption fails or the system does not start after replacing the file or the server:** the key (enc.key or webdesk.encryption.key) does not match the key used by the system.
- **An environment variable has no effect:** check the key in the table "Overridable keys" and the naming rule (dots become underscores, upper case); the variable must not be empty.
- **A change in the file has no effect or has disappeared after startup:** the file is only read at startup; Webdesk partly rewrites it (see "Automatic adjustments at startup") and sets webdesk.runRegistration = 1 to 0 after the registration.
- **Startup is very slow:** PoStartup.executeIndexManager, webdesk.disableTimeLineChecks, webdesk-sessionFactory.schemaUpdate, webdesk.runRegistration and webdesk.luceneIndexing determine which checks run at startup.

1. /daisy/personalwolke-admin/5655-dsy.html?language=4
2. /daisy/personalwolke-admin/10823-dsy.html?language=4
3. /daisy/personalwolke-admin/10823-dsy.html?language=4
4. /daisy/personalwolke-admin/5655-dsy.html?language=4