Management Database

The COBI.wms management database can be installed in **on-premises or private cloud** environments where you have direct access to the database server and can create your own databases on it. The management database allows you to centrally define and manage connections to SAP Business One databases, COBI.wms users and devices, and module permissions.

Creating the database

If you are using MS SQL Server, execute the contents of the following file in SQL Server Management Studio:

cobiwms-mssql.sql

If using SAP HANA, instead execute the following in HANA Studio:

cobiwms-hana.sql

Company connections

You must define SAP Business One database connections by inserting rows into the “companies” table.

Usually, you only need to fill in the following columns:

Column	Type / Valid values	Description
CompanyID	Text	Unique identifier for this connection
SQLDB	Text	Name of the SAP Business One database
APIType	SL or IF	SL for Service Layer; IF for Integration Framework
APIURL	Text	Service Layer URL or Integration Scenario Trigger-URL
APIID	Text	For SL: same as SQLDB; For IF: Company ID in the IF SLD
APIUsername	Text or NULL	For SL: SAP Business One username; For IF: NULL
APIPassword	Text or NULL	For SL: SAP Business One password; For IF: NULL

Tip: You can use the character sequence `{host}` as part of the APIURL value, to make the app use the same host name (or IP address) as that of the management database.

Tip: (only for Service Layer) If you want to enforce warehouse employees to use individual logins, you can leave the APIUsername and APIPassword fields NULL and only fill in the APIUser and APIPass fields in the Users table instead, as explained below in the section: Separate login per user

Note: If the ApiType is SL, then the SQLDB column can be left empty, which will cause the app to use a pure Service Layer connection as if in a Cloud environment. However, this is **strongly discouraged** because a direct database connection offers significantly higher performance and stability. Therefore, the SQLDB column should always be filled for all On-Premises installations.

See also: Architecture Overview

Example for adding a productive and a test connection for an on-premises environment, using unencrypted communication with Service Layer so it doesn't require a valid SSL Certificate:

```
INSERT INTO companies (companyId, sqlDb, apiType, apiUrl, apiId,
apiUsername, apiPassword) VALUES
('01 - PROD', 'SBO_PROD', 'SL', 'http://{host}:50001/b1s/v2', 'SBO_PROD',
'manager', 'secret');

INSERT INTO companies (companyId, sqlDb, apiType, apiUrl, apiId,
apiUsername, apiPassword) VALUES
('02 - TEST', 'SBO_TEST', 'SL', 'http://{host}:50001/b1s/v2', 'SBO_TEST',
'manager', 'secret');
```

Note: The examples above use **`http:`** instead of **`https:`** and the port number **50001** instead of **50000**. This means that communication with Service Layer will be unencrypted, and the app will skip over the Load Balancer and directly access Node 1 of Service Layer.

If it's important for you to have encrypted communication with Service Layer, and/or you experience performance issues due to the Load Balancer being skipped, then you must ensure that a valid SSL Certificate is installed for Service Layer, and change the `http` above to `https` and the port number 50001 to 50000.

Optional columns

The following columns of the companies table should usually be left empty, i.e. `NULL` or an empty text:

Column	Type / Valid values	Description
DBType	MSSQL or HANA	MSSQL for SQL Server, HANA for SAP HANA
SQLHost	Text	Database server host name or IP address
SQLPort	Text	Database server port number
SQLUser	Text	Database login user (e.g. 'sa' or 'SYSTEM')
SQLPass	Text	Database login password
SQLDomain	Text	Domain SQL Server “trusted connection”
HANAProxyHost	Text	HANA Proxy host name or IP address
HANAProxyPort	Text	HANA Proxy port number
Profile	Text	Code to enable a customer-specific profile
PrintService	Text	Address of the COBI.wms Print Service

The columns `DBType`, `SQLHost`, `SQLUser`, and `SQLPass` only need to be filled if the SAP Business One company database resides on a different database server than the one on which the management database is installed. I.e. you can redirect the app to a different server/database by filling these columns. The `SQLPort` column only needs to be filled if the database server listens on a different port than the default (1433 for MS SQL Server, 30015 for SAP HANA).

The `HANAProxyHost` column only needs to be filled if the HANA Proxy is not installed on the same server as the SAP HANA database itself. The `HANAProxyPort` only needs to be filled if the HANA

Proxy is configured to use a different port than the default value of 30075.

The `Profile` column is used to enable customer-specific specializations in the app and should be left empty unless instructed.

The `PrintService` column can be used to centrally define the address of the COBI.wms Print Service. If not defined here, it has to be set on each Android device in the Print Settings screen of COBI.wms. When using the standard port of the COBI.wms Print Service, you should only enter the host name or IP address in this column. When using a non-standard port number, you can specify it by entering the value `HOST:PORT` in this column where `HOST` is the host name or IP address and `PORT` is the TCP port number.

Devices and Users

COBI.wms Devices

Devices will register themselves automatically when they connect to the management database.

Devices are assigned a numeric ID beginning from 1, which can be seen in the login screen of the app at the bottom right of the login button.

A device can be removed by using the `removeDevice` procedure:

```
-- MS SQL Server
EXEC removeDevice 1;

-- SAP HANA
CALL removeDevice(1);
```

COBI.wms Users

This section is optional; the app can also be used without adding COBI.wms users to the management database.

You may add COBI.wms users to more strictly control access to the app and monitor what was done by who. The “Username” and “Password” fields in the app will only appear if at least one COBI.wms user is defined.

To add COBI.wms users, use the `addUser` procedure:

```
-- MS SQL Server
EXEC addUser 'user1', 'password', NULL, NULL, 'Full Name';

-- SAP HANA
CALL addUser('user1', 'password', NULL, NULL, 'Full Name');
```

The first parameter is the user ID that uniquely identifies the user and it is also the username for logging into the app. It could be a name like 'alice' or 'bob' that easily identifies an employee, or it

could be a symbolic name like 'manager' or 'production1'.

The second parameter is the password. It cannot be NULL, but it can be an empty text (just `''`) to allow the user to log in without having to type anything into the Password field.

The third and fourth parameters are obsolete and should be `NULL`.

The last parameter can be the full name, or a long description of the user, or it can be `NULL`.

The password of a user can be reset with the `resetPassword` procedure:

```
-- MS SQL Server
EXEC resetPassword 'user1', 'new password';

-- SAP HANA
CALL resetPassword('user1', 'new password');
```

Users can be removed with the `removeUser` procedure:

```
-- MS SQL Server
EXEC removeUser 'user1';

-- SAP HANA
CALL removeUser('user1');
```

Separate login per user

You can specify a separate SAP Business One login for each COBI.wms user or device. This way, the Change Log in SAP Business One can correctly display which COBI.wms user or device booked or updated a document.

Specify the SAP Business One login for a COBI.wms user by executing the following SQL command on the Management Database:

```
UPDATE users
SET apiUser = 'sbo_username',
    apiPass = 'sbo_password'
WHERE userId = 'cobiwms_username';
```

If you don't use COBI.wms users, you can do the same for a device. Each device that connects to the database actually creates a special COBI.wms user called `_deviceXXXX` (where XXXX is the device ID), and you can set the login data for that special user.

For example, you could create SAP Business One users called `wms0001`, `wms0002`, etc., and then assign the login data of those to the corresponding COBI.wms device users such as `_device0001`, `_device0002`, and so on.

```
UPDATE users
SET apiUser = 'wms0001',
    apiPass = 'password'
```

```
WHERE userId = '_device0001';

UPDATE users
SET apiUser = 'wms0002',
    apiPass = 'password'
WHERE userId = '_device0002';

-- And so on...
```

Once you've updated the `users` table with these commands, just restart the COBI.wms Android app and the change will take effect. You can make a test booking with the app and check the Change Log in SAP Business One to make sure that it worked.

WARNING: When you save the SAP Business One user's password in the `apiPass` field as shown above, the password will be seen in plain text in the management database, just like the `apiPassword` column of the `companies` table. This should generally not be an issue because untrusted persons should not have access to your database server. However, if this poses an issue for you, see the next section.

Avoiding plaintext passwords in the database

Normally, the password of an SAP Business One user has to be specified either in the `apiPassword` column of the `companies` table, or in the `apiPass` column of the `users` table.

(Technical details: It makes no sense to encrypt these columns, because the key to decrypt them would need to be deployed as part of the app, such that anyone could access it by analyzing the Android app. We cannot store a hashed value either, because the app needs to forward the plaintext password to Service Layer when logging in to SAP Business One.)

To avoid this issue, the following strategy can be used:

1. Leave the `apiPassword` and `apiPass` columns in the database empty
2. Create COBI.wms users with the same username and password as SAP Business One users

When you enter a username & password in the COBI.wms login screen, the app first uses these to perform a COBI.wms user login. It then tries to find a username & password for Service Layer by checking the `apiPassword` and `apiPass` columns mentioned above. However, if these columns are empty, the app will simply take the username & password that were entered for the COBI.wms user login, and try to use these for the Service Layer login as well. So, if the username & password of the COBI.wms user is the same as an SAP Business One user, it will work.

(Technical details: The password of a COBI.wms user is *not* stored in plaintext in the database, only a secure hash value of it is stored, since it doesn't need to be forwarded anywhere.)

License management

Licensing model

Every parallel access to COBI.wms requires a license. For example, if a maximum of 3 people will use COBI.wms at the same time, you will need 3 licenses. However, whether you want to license devices or users is up to you.

You could assign your licenses to three devices, so anyone can use those devices with or without a COBI.wms user. Or you could assign your licenses to three COBI.wms users, so those users can use COBI.wms on any number of Android devices by using their login. You can also mix the two models. For example, you could assign licenses to two devices so anyone can use them, and assign a third license to a user so that user can use any Android device to log in.

Importing licenses

Import licenses simply by executing INSERT statements:

```
-- Change LICENSE_1, LICENSE_2 etc. to the actual license, keep the
apostrophes.
INSERT INTO licenses (license) VALUES ('LICENSE_1');
INSERT INTO licenses (license) VALUES ('LICENSE_2');
INSERT INTO licenses (license) VALUES ('LICENSE_3');
```

The licenses table also has an optional `notes` column which you can use for notes about the license. For example, if you have both regular COBI.wms licenses as well as COBI.ppc licenses in the same database, you can differentiate them through these notes. Or, when importing test licenses that are only valid for a limited time, you can enter this as a note. Examples:

```
INSERT INTO licenses (license, 'notes') VALUES ('LICENSE_1', 'WMS');
INSERT INTO licenses (license, 'notes') VALUES ('LICENSE_2', 'PPC');
INSERT INTO licenses (license, 'notes') VALUES ('LICENSE_3', 'PPC, valid
until November 2023');
```

Assigning licenses

Bulk-editing the LICENSES table

The information of which user or device a license is assigned to is found directly in the LICENSES table of the management database. If you want to make a large number of changes, it might be easiest to directly modify this table.

For example, in MS SQL Server Management Studio, you can right-click on the Devices table and select “Edit top 200 rows” and directly edit the “UserID” or “DeviceID” column of each license. (For each license, only one of the columns can be filled, the other must be NULL.)

Using stored procedures

To assign licenses to devices and/or users, you can also use the `assignDeviceLicense` and `assignUserLicense` procedures. These will automatically check whether you have any free

(unassigned) licenses and use one of them:

```
-- MS SQL Server
EXEC assignDeviceLicense 1; -- Give a free license to Device 1
-- or
EXEC assignUserLicense 'user1'; -- Give a free license to User 'user1'

-- SAP HANA
CALL assignDeviceLicense(1); -- Give a free license to Device 1
-- or
CALL assignUserLicense('user1'); -- Give a free license to User 'user1'
```

For revoking licenses, you can use the `revokeDeviceLicense` and `revokeUserLicense` procedures. This will free up the license that is currently used by a given user or device, so you can then assign it to another user or device:

```
-- MS SQL Server
EXEC revokeDeviceLicense 1; -- Take away the license of Device 1
-- or
EXEC revokeUserLicense 'user1'; -- Take away the license of User 'user1'

-- SAP HANA
CALL revokeDeviceLicense(1); -- Take away the license of Device 1
-- or
CALL revokeUserLicense('user1'); -- Take away the license of User 'user1'
```

Permission management

All modules of the app are enabled by default for all devices and users, and have to be blocked explicitly if you want to change this.

The settings for users take precedence over the settings for devices. For example, you could block a number of modules for a certain device, but if a user logs in on that device who is explicitly given permissions for those modules, then the user can use those modules anyway. Conversely, if a number of modules are explicitly blocked for a user, then no matter what device the user logs in to, those modules will not be available to that user.

You can effectively block usage of the app without COBI.wms user login, by blocking all modules for all devices, and then giving permissions to the users. (This can also be done by leaving the `apiUsername` and `apiPassword` fields in the Companies table empty, and only setting them for individual users.)

To allow/block modules for devices/users, use the `setDevicePermission` and `setUserPermission` procedures:

```
-- MS SQL Server
EXEC setDevicePermission 1, 'MODULE_ID', 0; -- Device 1 has MODULE_ID
disabled
EXEC setUserPermission 'user1', 'MODULE_ID', 1; -- But user1 has it enabled
```

```
so she/he can use it anyway

-- SAP HANA
CALL setDevicePermission(1, 'MODULE_ID', 0); -- Device 1 has MODULE_ID
disabled
CALL setUserPermission('user1', 'MODULE_ID', 1); -- But user1 has it enabled
so she/he can use it anyway
```

The first parameter is the device or user ID, the second is a module ID, and the third parameter is the allow/block status. 1 means allow, 0 means block. So in the above example, the module “MODULE_ID” is blocked for device 1, but explicitly allowed for the user 'user1'.

Following is the list of module IDs:

- IGN: Plus booking
- IGE: Minus booking
- WTR: Inventory transfer
- PDN: Receipt
- PKL: Picking
- DLN: Delivery
- RPD: Purchase return
- RDN: Sales return
- IPE: Issue for production
- IPN: Receipt from production
- PRQ: Purchase request
- POR: Purchase order
- ITM: Wares list
- INC: Inventory counting
- WTQ: Inventory transfer request
- PRINT: Label printing

For your convenience, here is a template for calling the `setUserPermission` procedure once for every module, which you can copy & paste into SQL Server Management Studio or HANA Studio:

```
-- MS SQL Server
EXEC setUserPermission 'username', 'IGN', 1;
EXEC setUserPermission 'username', 'IGE', 1;
EXEC setUserPermission 'username', 'WTR', 1;
EXEC setUserPermission 'username', 'PDN', 1;
EXEC setUserPermission 'username', 'PKL', 1;
EXEC setUserPermission 'username', 'DLN', 1;
EXEC setUserPermission 'username', 'RPD', 1;
EXEC setUserPermission 'username', 'RDN', 1;
EXEC setUserPermission 'username', 'IPE', 1;
EXEC setUserPermission 'username', 'IPN', 1;
EXEC setUserPermission 'username', 'PRQ', 1;
EXEC setUserPermission 'username', 'POR', 1;
EXEC setUserPermission 'username', 'ITM', 1;
EXEC setUserPermission 'username', 'INC', 1;
EXEC setUserPermission 'username', 'WTQ', 1;
EXEC setUserPermission 'username', 'PRINT', 1;
```

```
-- SAP HANA
CALL setUserPermission('username', 'IGN', 1);
CALL setUserPermission('username', 'IGE', 1);
CALL setUserPermission('username', 'WTR', 1);
CALL setUserPermission('username', 'PDN', 1);
CALL setUserPermission('username', 'PKL', 1);
CALL setUserPermission('username', 'DLN', 1);
CALL setUserPermission('username', 'RPD', 1);
CALL setUserPermission('username', 'RDN', 1);
CALL setUserPermission('username', 'IPE', 1);
CALL setUserPermission('username', 'IPN', 1);
CALL setUserPermission('username', 'PRQ', 1);
CALL setUserPermission('username', 'POR', 1);
CALL setUserPermission('username', 'ITM', 1);
CALL setUserPermission('username', 'INC', 1);
CALL setUserPermission('username', 'WTQ', 1);
CALL setUserPermission('username', 'PRINT', 1);
```

Just change `username` to the actual username via search & replace in Notepad or the like, and switch the 1 to a 0 for the modules to disable.

Here's the same for devices:

```
-- MS SQL Server
EXEC setDevicePermission deviceID, 'IGN', 1;
EXEC setDevicePermission deviceID, 'IGE', 1;
EXEC setDevicePermission deviceID, 'WTR', 1;
EXEC setDevicePermission deviceID, 'PDN', 1;
EXEC setDevicePermission deviceID, 'PKL', 1;
EXEC setDevicePermission deviceID, 'DLN', 1;
EXEC setDevicePermission deviceID, 'RPD', 1;
EXEC setDevicePermission deviceID, 'RDN', 1;
EXEC setDevicePermission deviceID, 'IPE', 1;
EXEC setDevicePermission deviceID, 'IPN', 1;
EXEC setDevicePermission deviceID, 'PRQ', 1;
EXEC setDevicePermission deviceID, 'POR', 1;
EXEC setDevicePermission deviceID, 'ITM', 1;
EXEC setDevicePermission deviceID, 'INC', 1;
EXEC setDevicePermission deviceID, 'WTQ', 1;
EXEC setDevicePermission deviceID, 'PRINT', 1;

-- SAP HANA
CALL setDevicePermission(deviceID, 'IGN', 1);
CALL setDevicePermission(deviceID, 'IGE', 1);
CALL setDevicePermission(deviceID, 'WTR', 1);
CALL setDevicePermission(deviceID, 'PDN', 1);
CALL setDevicePermission(deviceID, 'PKL', 1);
CALL setDevicePermission(deviceID, 'DLN', 1);
CALL setDevicePermission(deviceID, 'RPD', 1);
CALL setDevicePermission(deviceID, 'RDN', 1);
```

```
CALL setDevicePermission(deviceID, 'IPE', 1);
CALL setDevicePermission(deviceID, 'IPN', 1);
CALL setDevicePermission(deviceID, 'PRQ', 1);
CALL setDevicePermission(deviceID, 'POR', 1);
CALL setDevicePermission(deviceID, 'ITM', 1);
CALL setDevicePermission(deviceID, 'INC', 1);
CALL setDevicePermission(deviceID, 'WTQ', 1);
CALL setDevicePermission(deviceID, 'PRINT', 1);
```

Change `deviceID` to the correct device ID number via search & replace, and switch the 1 at the end to a 0 for the modules to disable.