new solutions
business software

Checkware

Event Engine

checkware
DIGITAL CHECKLIST MANAGER

Document Informations:

Creation date:	2016/06/28
Last changes:	2016/06/28
Version:	1.0.0

new solutions
business software

Table of Contents

1. Overview....2
2. Creating an event script....3
2.1 Opening the 'New Event Script' tab....3
2.2 Overview of the 'New Event Script' tab....4
2.3 Event script example....5
3. Assign event script to template....6
4. API – Reacting to events....7
4.1 Listening to events....7
4.2 Functions....7
5. Logging in database....12

1. Overview

The Checkware Event Engine allows users to react on different events in the life cycle of a checklist, and to implement various kinds of logic when a certain stage is reached. For example, you can setup a script to send an email whenever a checklist is completed.

There are four events the Event Engine recognizes:

- **Init**: This event is triggered, when the checklist is opened in Checkware.
- **Save**: Whenever the checklist is saved, this event gets triggered.
- **Complete**: Upon completion of the checklist, this event is triggered.
- **Delete**: This event is triggered, when the checklist is deleted.

To use the Checkware Event Engine, you first create a new event script and then assign it to the checklist(s) of your choice.

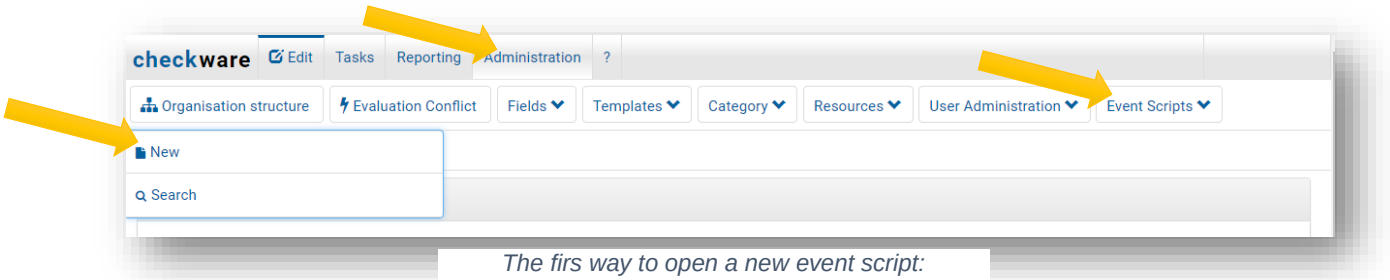
In the following chapters, we will take a closer look at how to create an event script, how to assign it to a checklist, and at the API of the Event Engine.

2. Creating an event script

2.1 Opening the ‘New Event Script’ tab

After opening Checkware Web you have two ways to get to the event script creation tab. You can either:

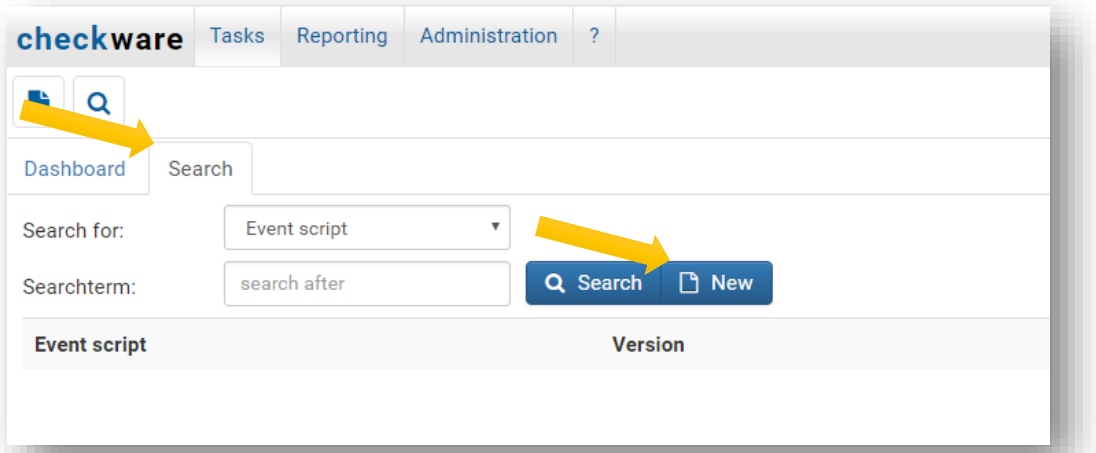
1. Use the ribbon menus: Click on ‘Administration’ → ‘Event Scripts → ‘New’.



The firs way to open a new event script: ‘Administration’ → ‘Event Scripts’ → ‘New’

Or you can:

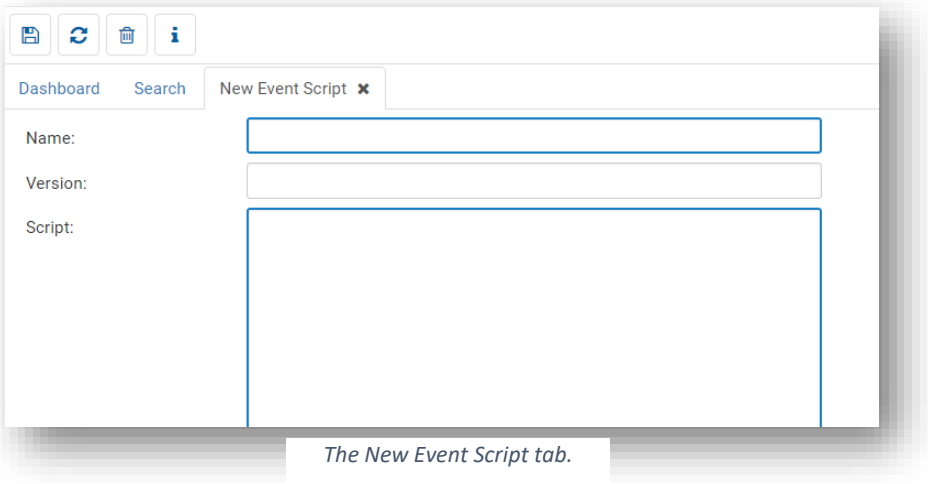
1. Click on ‘Search’.
2. Choose ‘Event script’ from the drop-down menu ‘Search for’.
3. Finally click on the button New.



The second way uses the Search tab.

2.2 Overview of the ‘New Event Script’ tab

After opening the tab, you will see the empty input screen for your event script.



The New Event Script tab.

There are three fields on the tab:

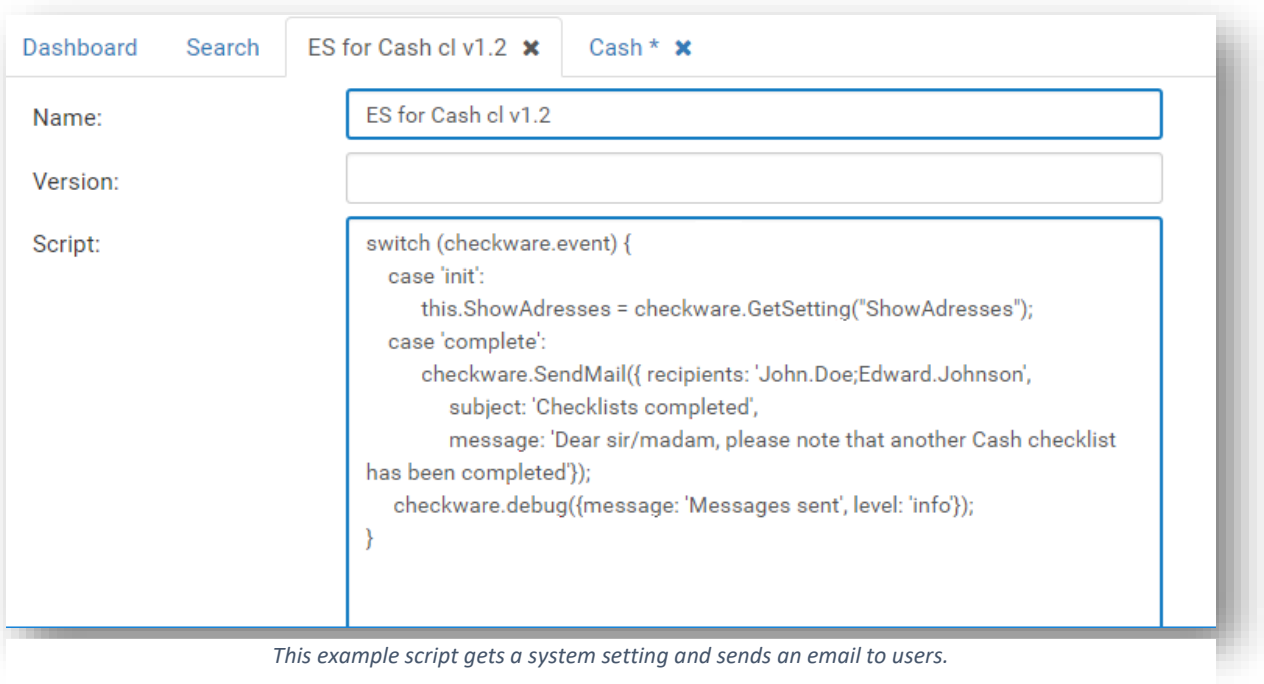
Name	The name for the event script. Mandatory field.
Version	Here you can note the version of the script if you have multiple versions of the same script.
Script	This field holds the script itself. Mandatory field. (For details on how to write an event script, see Chapter 4: API.)

Also, the tab has four buttons in the top left corner:

	Use this button to save the event script.
	With this button you can update the view, so you can see if another user made any changes.
	Delete the event script with this button.
	Shows information about this entry: When and by whom it was created and last updated.

2.3 Event script example

The following example shows a small event script that calls a system setting and saves it in a variable so that it can later be used to show/hide certain parts of the checklist, and that sends an email to two users to inform them when the checklist is completed.



This example script gets a system setting and sends an email to users.

For further information on how to write the event script, see Chapter 4: API.

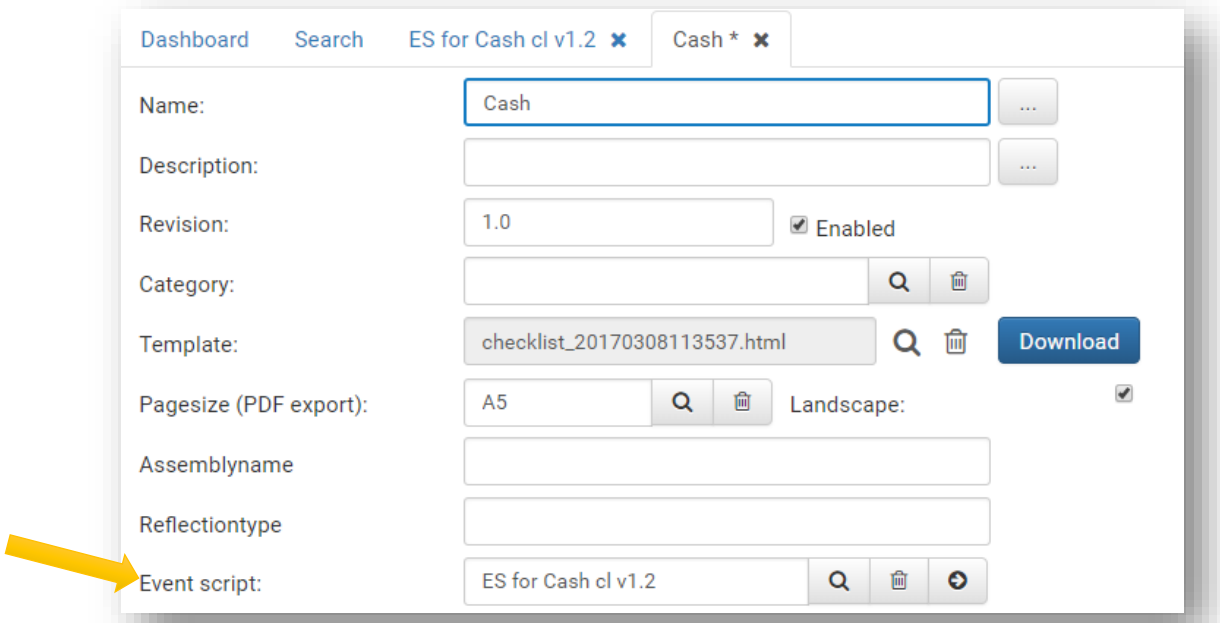
3. Assign event script to template

After you have entered and saved your event script, you now have to assign it to the template of the checklist that you have created it for.

Navigate to the template by:

1. *Either using the ribbon menu*: Click on 'Administration' → 'Templates' → 'Search'. *Or by:* Clicking on the 'Search' tab and choosing 'Templates' from the Search for dropdown.
2. Click on the Search button.
3. Click on the desired checklist.

The template tab for that checklist opens. There you can find the field 'Event script' where you can search for the script you want. Click on the button and select your script from the new dialog. Save the template, and then the script is assigned to this checklist.



The template for a checklist called Cash. At the bottom is the field where you can set the event script.

4. API – Reacting to events

4.1 Listening to events

We can check for events using the `checkware` variable. It holds the `event` property with the four events `init`, `save`, `complete`, and `delete`.

The easiest way to react to any event is to use a simple `switch` statement:

```
switch (checkware.event) {
    case 'init':
        // ...
        break;
    case 'save':
        // ...
        break;
    case 'complete':
        // ...
        break;
    case 'delete':
        // ...
        break;
    default:
        // ...
}
```

With this code, when an event in the checklist occurs, everything in the corresponding `case` section of the switch statement will be executed.

4.2 Functions

In addition to the possibility to listen for events, the Event Engine has a couple of helpful functions you can use in the event scripts.

4.2.1 Main functions

These are the functions you'll normally use in your day-to-day checklists.

sendMail()

Can be used to send an email to someone.

Parameter	{
	recipients: string, // recipients separated with ';'
	subject: string,
	message: string
	}

Return	void
Example	`checkware.sendMail({ recipients: 'User1;User2', subject: 'ExampleSubject', message: 'ExampleMessage' });`

getValue()

Can be used to get a value from the checklist (in terms of the database: an EvaluationValue).

Parameter	matchcode: string
Return	string || number || boolean
Example	`var evaluationValue = checkware.getValue('ExampleMatchcode');`

setValue()

Can be used to set a value from the checklist (in terms of the database: an EvaluationValue).

Parameter	{
	matchcode: string,
	value: string || number || boolean
	}

Return	void
Example	`checkware.setValue({ matchcode: 'ExampleMatchcode', value: 'ExampleValue' });`

createEvaluation()

Can be used to create a new evaluation.

Parameter	{
	folderStructureChecklistId: string,
	parameters: Dictionary<string, string> // Matchcode-Value
	}

Return	void
Example	`checkware.createEvaluation({ folderStructureChecklistId: '11111111-2222-3333-4444-555555555555', parameters: [{ key: 'ExampleMatchcode', value: 'ExampleValue' }] });`

deleteEvaluation()

Can be used to delete the evaluation that called this event script.

Parameter	none
Return	void

Example	`checkware.deleteEvaluation();`

deleteEvaluation()

Can be used to delete the evaluation that called this event script.

Parameter	none
Return	void
Example	`checkware.deleteEvaluation();`

completeEvaluation()

Can be used to complete the evaluation that called this event script.

Parameter	none
Return	void
Example	`checkware.completeEvaluation();`

assignTo()

Can be used to assign the evaluation that called this event script to a different user.

Parameter	Username: string
Return	void
Example	`checkware.assignTo('ExampleUser');`

createTask()

Creates a new task.

Parameter	{ folderstructurechecklistid: string, // optional taskType: number, // 0: all users must return; 1: one user must return startDate: string, // e.g. 20101220101233 - yyyyMMddHHmmss endDate: string, // e.g. 20101220101233 - yyyyMMddHHmmss remark: string,

	recipients: List<string> // array of usernames }
Return	void
Example	`checkware.createTask({ folderStructureChecklistId: '11111111-2222-3333-4444-555555555555', taskType: 1, startDate: '20180601120000', endDate: '20180610120000', description: 'Example description', recipients: ['ExampleUser1', 'ExampleUser2'] });`

deleteTask()

Deletes the task that is assigned to this evaluation.

Parameter	none
Return	void
Example	`checkware.deleteTask();`

callConnector()

Calls a method on a connector

Parameter	{
	connectorKey: string,
	method: string,
	parameter: string // JSON object
	}

Return	ConnectorResult = {
	Success: boolean,
	Message: string,
	Result: object
	}

Example	`var result = checkware.callConnector({ connectorKey: 'Demo1', method: 'Add', parameter: '{ Number1: 10, Number2: 5 }' });`

4.2.2 Utility functions

These functions help you with the programmatical aspects of working with checklists.

debug()

Creates a log entry for the event execution in the database (table *EventEngineLog*). This is useful for debugging purposes, e.g. when a user wants to know if a specific line of code was executed.

Parameter	{
	message: string,
	level: string
	}

Return	void
Example	`checkware.debug({ message: 'ExampleMessage', level: 'warning' });`

getSetting()

Gets a system setting from the database.

Parameter	Setting: string
Return	string
Example	`var setting = checkware.getSetting('Foo');`

setSetting()

Sets a system setting from the database.

Parameter	{
	setting: string,
	value: string
	}

Return	void
Example	`checkware.setSetting({ setting: 'Foo', value: 'ExampleValue' });`

5. Logging

With the `debug` command (see last chapter) you can write log entries in the database for debugging. The table ***EventEngineLog*** holds all entries this command creates. This table has the following columns:

EventEngineLogIg	The unique identifier for this entry
EventScriptId	The unique identifier of the script that has made this entry
ChecklistId	The unique identifier of the checklist template the script belongs to
EvaluationId	The unique identifier of the evaluation from which the script was called
Message	The message that was set in the `debug()` method that created this entry
Level	The level of this log entry: info/warning
InsertBy/InsertDate	The date this entry was created and who created it
UpdateBy/ UpdateDate	The date this entry was last updated and who updated it
DeleteBy/ DeleteDate	The date this entry was deleted and who deleted it

In a future release of Checkware, you will even be able to browse these event logs from within Checkware Web itself.