Standard Scripting Guidelines

October, 2017

Table of Contents

[Document Purpose 3](#_Toc496607784)

[Scripting Strategy 3](#_Toc496607785)

[Configurable Scripts 3](#_Toc496607786)

[SOLUTION\_MAPPING 3](#_Toc496607787)

[STANDARD\_SOLUTIONS 3](#_Toc496607788)

[CONFIGURABLE\_RULESETS 4](#_Toc496607789)

[Individual Ruleset Configuration 4](#_Toc496607790)

[Setup 4](#_Toc496607791)

# Document Purpose

The purpose of this document is to outline the new scripting standards Delivery Solutions is implementing to decrease the amount of scripting and group scripting into solution related scripts. This document will also walk through how Configurable Scripts will run, which will include auto-run functionality in release 9.2.1 of Accela Civic Platform.

# Scripting Strategy

The new script strategy is to create re-useable and repeatable scripts that are easy to support and bundle with our certified and standard solutions. To do this, we have created a new methodology around configurable scripts and solution includes.

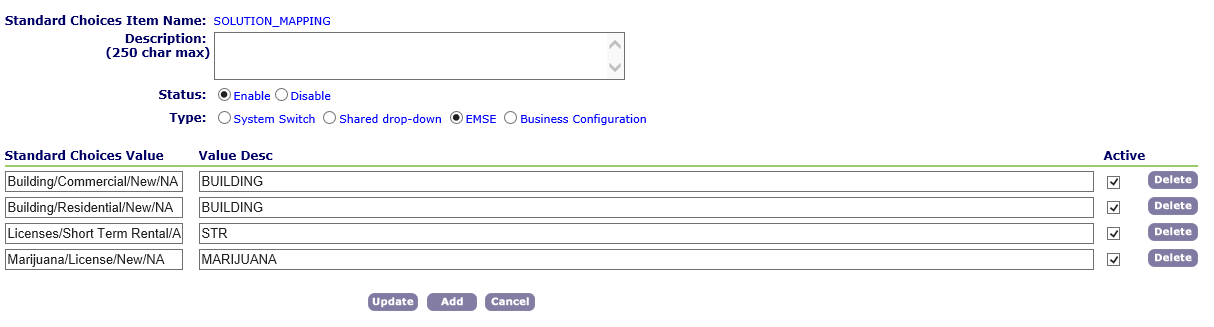
## Configurable Scripts

Configurable scripts use JSON objects to define rule sets. Each rule set will have distinct rules for that set. This will not be documented here but is available in the instructions for each configurable script. In order to utilize the configurable scripts you must have the following setup. A data manager job will include all of this and is documented in the Setup area of this guide.

* SOLUTION\_MAPPING standard choice
* STANDARD\_SOLUTIONS standard choice
* CONFIGURABLE\_RULESETS standard choice
* The individual configurable ruleset scripts
* Your project specific JSON rulesets

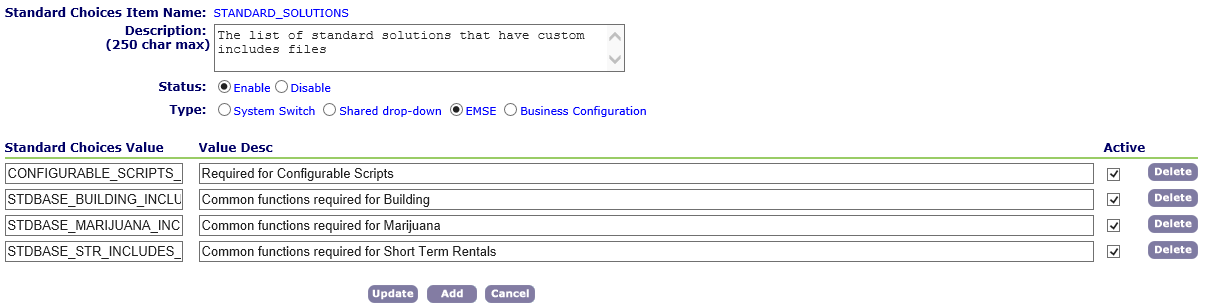
### SOLUTION\_MAPPING

Rule sets will be packaged by the solution they belong to. These are JSON definitions stored like you would any other script. This standard choice will allow you to define which record types belong to which solutions. The value will be the four-layer record type and the description will be the solution those record types belong to. This will support wild cards as well, for example Marijuana/\*/\*/\* with solution MARIJUANA.



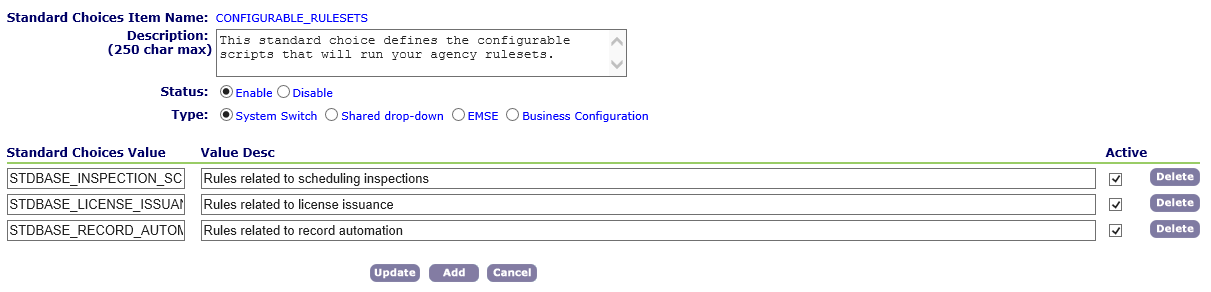
### STANDARD\_SOLUTIONS

Each standard solution may require specific functions related to that standard solution. Alternatively, agencies may require custom functions created for their solutions. This standard choice allows you to achieve that by defining a specific common file for each solution. Naming convention should follow the convention of STDBASE\_{SOLUTION\_NAME}\_INCLUDES\_CUSTOM.



### CONFIGURABLE\_RULESETS

This standard choice defines the configurable scripts that will run your agency rulesets. You can also add new rulesets if you develop a new configurable rules function. Naming convention should follow the STDBASE\_{RuleSet\_Function}. Please do not abbreviate the rule set function name, verbose is required.



### Configurable Ruleset Scripts

The configurable ruleset scripts will auto run as of 9.2.1. Early versions can manually call these as discussed in the Setup section. These will handle the same naming convention above, STDBASE\_{RuleSet\_Function}. They will be saved much like you would an event script in the past.

Configurable Ruleset Scripts should be developed in a very specific manner. They should utilize the CONFIGURABLE\_SCRIPTS\_COMMON library for re-useable functions. This will automatically be pulled in and does not need to be referenced in the script.

Script should follow a try catch method with a function built to run the logic. The below is an example of the code structure that should be followed:

var scriptSuffix = "VERBOSE\_NAME"; // CONF\_{SOLUTION}\_LICENSE\_ISSUANCE

// {SOLUTION} = AS DEFINED IN THE "SOLUTION MAPPING" STANDARD CHOICE

try {

var settingsArray = [];

if (isConfigurableScript(settingsArray, scriptSuffix)) {

for (s in settingsArray) {

var rules = settingsArray[s];

//Execute PreScript

var preScript = rules.preScript;

if (!matches(preScript, null, "")) {

eval(getScriptText(preScript));

}

//Execute function here

verboseName(param1, param2);

//Execute Post Script

var postScript = rules.postScript;

if (!matches(postScript, null, "")) {

eval(getScriptText(postScript));

}

}

}

} catch (ex) {

logDebug("\*\*ERROR: Exception while verification the rules for " + scriptSuffix + ". Error: " + ex);

}

/\*\*

\* Standard base script for Verbose Name

\*

\* @param1 param description

\* @param2 param description

\*/

function verboseName(param1, param2) {

### Project Ruleset Configuration

When building your ruleset JSON, you should follow the specifications for that given ruleset. These should be saved as a script much like you would an event script in the past. Naming convention should follow the CONF\_{SOLUTION\_NAME}\_{RULESET\_NAME}. It is critical that the solution name is the same solution you have mapped in the SOLUTION\_MAPPING standard choice. The Ruleset name should follow the CONFIGURABLE\_RULESETS name without the STDBASE\_ in front. For example, CONF\_BUILDING\_INSPECTION\_SCHEDULING.

NOTE: in the JSON files, you can only have the JSON. Do not have any comments or other text

# Setup

As of 9.2.1 you do not need to define when the includes and configurable scripts will run, they will run for you. A Data Manager import of all standard choices and scripts can be found at TO BE ENTERED. Once you import that, you can simply create your JSON files for your rules logic. This does not hold true for pageflow, which will need to be manually set up.

If you are in an environment prior to 9.2.1 you will need to manually run the necessary script import using the below code. This would need to be run on any master event such as WTUA:\*/\*/\*/\* or ASA:\*/\*/\*/\*.

//Add Standard Solution Includes

solutionInc = aa.bizDomain.getBizDomain("STANDARD\_SOLUTIONS").getOutput().toArray();

for (sol in solutionInc) {

if (solutionInc[sol].getAuditStatus() != "I") eval(getScriptText(solutionInc[sol].getBizdomainValue(),null));

}

//Add Configurable RuleSets

configRules = aa.bizDomain.getBizDomain("CONFIGURABLE\_RULESETS").getOutput().toArray();

for (rule in configRules) {

if (configRules[rule].getAuditStatus() != "I") eval(getScriptText(configRules[rule].getBizdomainValue(),null));

}