Script Concepts

John Schomp
Director, Accela Professional Services

Jason Plaisted
Senior Customer Success Manager

ENGAGE 2015

Introduction

- The Basics
  - What do you need to write scripts?
  - EMSE, Events, and Scripts
  - Master Scripts and Business Scripts
  - Installing and Your First Script
- Going Deeper
  - Master Script Versions, difference
  - Include Files
  - Standard Choices
  - Scripts
  - Standard Choice Configurations
  - Global Variables
  - Event Variables
- Productized vs. User Scripts
- Debug Output
- Script Tester
- Try/Catch Blocks
- Debugging
  - Settings
  - Common Errors
- New Functions in 2.0 and 3.0

What do you need to write scripts?

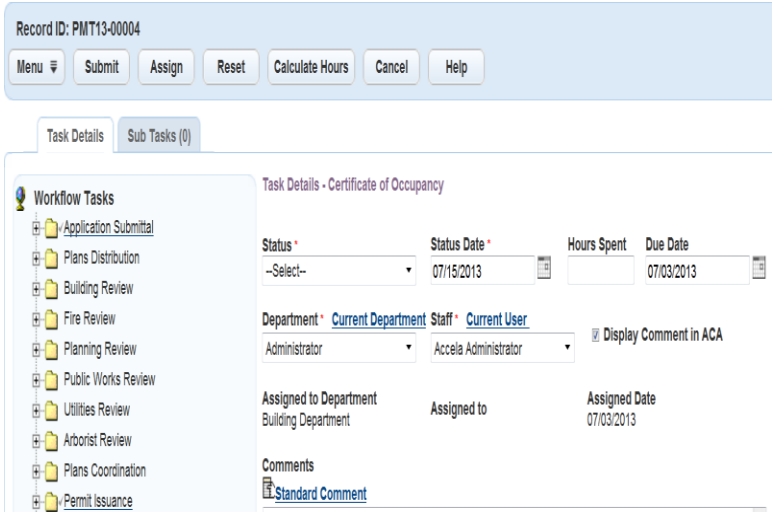
- Knowledge of JavaScript
  - Books: many to choose from
  - Online: CodeAcademy.com, Lynda.com, w3schools.com
- Enterprise Scripts release https://accela.force.com/success/06960000002BfFf
- Accela Automation 7.3 FP3 Scripting Guide https://accela.force.com/success/06960000002BJaH
- Dev Environment (IDE): Eclipse, Notepad++, Visual Studio

What is EMSE?

- EMSE = Event Manager and Script Engine
  - Core Civic Platform feature
  - Leverages Rhino implementation of JavaScript
  - Work together to allow customization to Civic Platform
  - Event Manager controls timing
  - Script Engine executes actions

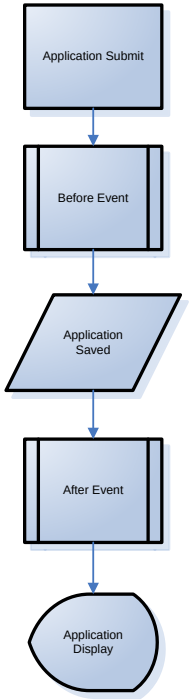
What is an Event?

- Action triggered by a user
- “Submit” button on a form
- Saving data
- 280+ unique events defined in Accela Automation



Event Types

- **Before**
  - Prior to database action
  - Database action can be cancelled via script
  - Typically used for validation
  - Example: ApplicationSubmitBefore
- **After**
  - After database action completes
  - Typically used for automation
  - Example: ApplicationSubmitAfter



What is a script?

- What is a **Script**?
  - JavaScript code
  - Stored in the database
  - In order to be executed, must be attached to:
    - Event,
    - Batch Job, or
    - Script Test
    - Set

```
function addFee(fcode,fsched,fperiod,fqty,finvoice) // Adds a single fee
        {
        assessFeeResult = aa.finance.createFeeItem(capId,fsched,fcode,fperiod,fqty);
        if (assessFeeResult.getSuccess())
                {
                feeSeq = assessFeeResult.getOutput();
                message+="Successfully added Fee " + fcode + ", Qty " + fqty + br;
                debug+="The assessed fee Sequence Number " + feeSeq + br;
                if (finvoice == "Y")
                        {
                        feeSeqList.push(feeSeq);
                        paymentPeriodList.push(fperiod);
                        }
                }
        else
                {
                debug+= "ERROR: assessing fee (" + fcode + "): " + assessFeeResult.getErrorMessage() + br;
                }
        }
```

Common Events with Available Master Scripts

- **ApplicationSubmitBefore/After**
- **ApplicationSpecificInfoUpdateBefore/After**
- **ApplicationStatusUpdateBefore/After**
- **ContactAddBefore/After**
- **DocumentUploadBefore/After**
- **FeeAssessBefore/After**
- **PaymentReceiveBefore/After**
- **VoidPaymentBefore/After**
- **InspectionMultipleScheduleBefore/After**
- **V360InspectionResultSubmitBefore/After**
- **WorkflowTaskUpdateBefore/After**

Master Scripts and Business Scripts

- *Master Scripts*
  - Framework under which Business Scripts run
  - Common across all agencies
  - One master script per event
  - Provide functions representing common business actions
  - Pre-loaded with information about the entity (i.e. record, asset, inspection, workflow, etc.) triggering the event.
- *Business Scripts*
  - Defined for each individual agency
  - Enforce their business rules (automation and validation)

Master Script Global Variables

- Script Global Variables
  - Event dependent
    - Workflow: wfTask, wfStatus, wfDate, etc.
    - Inspection Result: inspType, inspResult, InspGroup, etc.
  - Debug pop-up window shows the full list

Message

EMSE Script Framework Versions
EVENT TRIGGERED: WorkflowTaskUpdateBefore
SCRIPT EXECUTED: WORKFLOWTASKUPDATEBEFOREV3.0
INCLUDE VERSION: 3
SCRIPT VERSION : 3
GLOBAL VERSION : 3
ASI Table Array : SECONDARYBUSINESSTYPE (1 Rows)
ASI Table Array : EMPLOYEEINFO (0 Rows)
ASI Table Array : LICENSEREVIEWBOARD (0 Rows)
ASI Table Array : MONTHLYTAXFEES (0 Rows)
**WARNING: getting project parents: Project Parents Not Found: record ID: MOOSEJAW-12000-00000
**WARNING: Could not find parent license Cap for child CAP(12000-00000-00006):
EMSE Script Results for ALC000002
capId = class com.accela.aa.aamain.cap.CapIDModel
cap = class com.accela.aa.emse.dom.CapScriptModel
currentUserID = ADMIN
currentUserGroup = LicensesAdmin
systemUserObj = class com.accela.aa.aamain.people.SysUserModel
appTypeString = Licenses/Business License/Alcohol/NA
capName = null
capStatus = Pending
fileDate = 7/24/2012
fileDateYYYYMMDD = 2012-07-24
sysDate = class com.accela.aa.emse.util.ScriptDateTime
parcelArea = 0
estValue = 0
calcValue = 0
feeFactor = CONT
houseCount = 0

Getting Started

Server URL	https://av.training.accela.com
Agency	script2 through script36
User Name	admin
Password	admin

Loading Scripts

- Option 1: Manually (zip file)
- Option 2: Import scripts and config std choices automatically via Data Manager
- Link events to new master scripts

Exercise

- Exercise 1, Installing the Master Scripts 3.0

Configurations (1 of 2)

Standard Choices Item Name: EMSE_EXECUTE_OPTIONS

Description: (250 char max)

Status: Enable Disable

Type: System Switch Shared drop-down EMSE Business Configuration

Standard Choices Value	Value Desc	Active
SCRIPT		☑
STD_CHOICE		☐

Standard Choices Item Name: MULTI_SERVICE_SETTINGS

Description: (250 char max)

Status: Enable Disable

Type: System Switch Shared drop-dowr

Standard Choices Value	Value Desc
AGENCY_LOGO_TYPE	USERINFOLOGO
ALLOW_SERVICE_LOCK_SE	No
IS_SUPER_AGENCY	No
SUPER_AGENCY_FOR_EMSE	NYELS
SUPER_AGENCY_INCLUDE_S	INCLUDES_CUSTOM_ENTERPRISE

Standard Choices Item Name: EMSE_VARIABLE_BRANCH_PREFIX

Description: (250 char max)

Status: Enable Disable

Type: System Switch Shared drop-dov

Standard Choices Value	Value Desc
ApplicationConditionAddAfter	ACAA
ApplicationConditionDeleteAfter	ACDA
ApplicationConditionOfApproval	ACUA
ApplicationConditionUpdateAfte	ACUA
ApplicationDetailUpdateAfter	ADUA

Configurations (2 of 2)

Script Code: INCLUDES_CUSTOM_GLOBALS

Script Title: INCLUDES_CUSTOM_GLOBALS

Script Initializer:

Script Text:

```
showDebug = false;

if (currentUserID == "SAXTHELM") showDebug = 3;
if (publicUserID == "PUBLICUSER51") showDebug = 3;

var envName = "DEV";
var sysFromEmail = "dos_noreply@eLicensing.ny.gov";
var acaUrl = lookup("ACA_CONFIGS","OFFICIAL_WEBSITE_URL");
var schoolVerifEmail = false;
var LICENSESTATE = "NY";
```

Events - Event List

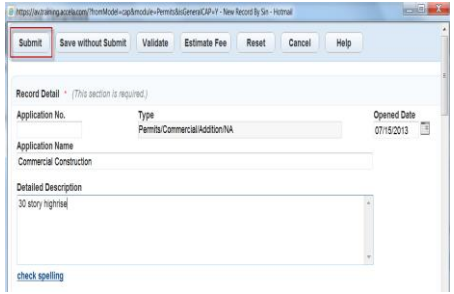
Edit	Event	Associated Script
•	ApplicationConditionAddAfter	ApplicationConditionAddAfterV3.0
•	ApplicationConditionDeleteAfter	UniversalMasterScriptV3.0
•	ApplicationConditionOfApprovalUpdateAfter	ApplicationConditionUpdateAfterV3.0
•	ApplicationConditionUpdateAfter	ApplicationConditionUpdateAfterV3.0
•	ApplicationDetailUpdateAfter	UniversalMasterScriptV3.0
•	ApplicationSpecificInfoUpdateAfter	ApplicationSpecificInfoUpdateAfterV3.0
•	ApplicationStatusUpdateAfter	ApplicationStatusUpdateAfterV3.0
•	ApplicationSubmitAfter	ApplicationSubmitAfterV3.0
•	ApplicationSubmitBefore	ApplicationSubmitBeforeV3.0

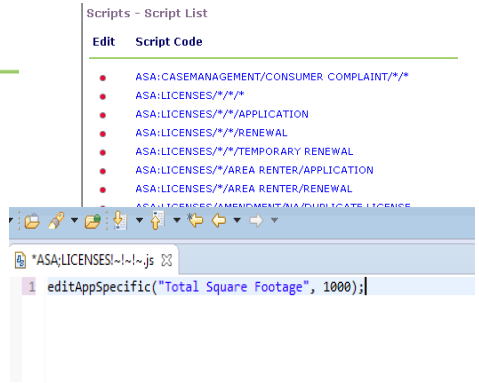
Exercise

- Exercise 2 – Configure Master Scripts
  - **ApplicationSpecificInfoUpdateAfter**
  - **ApplicationSubmitAfter**
  - **WorkflowTaskUpdateAfter**
  - **Clear Cache!!**

Master Script Flow Execution

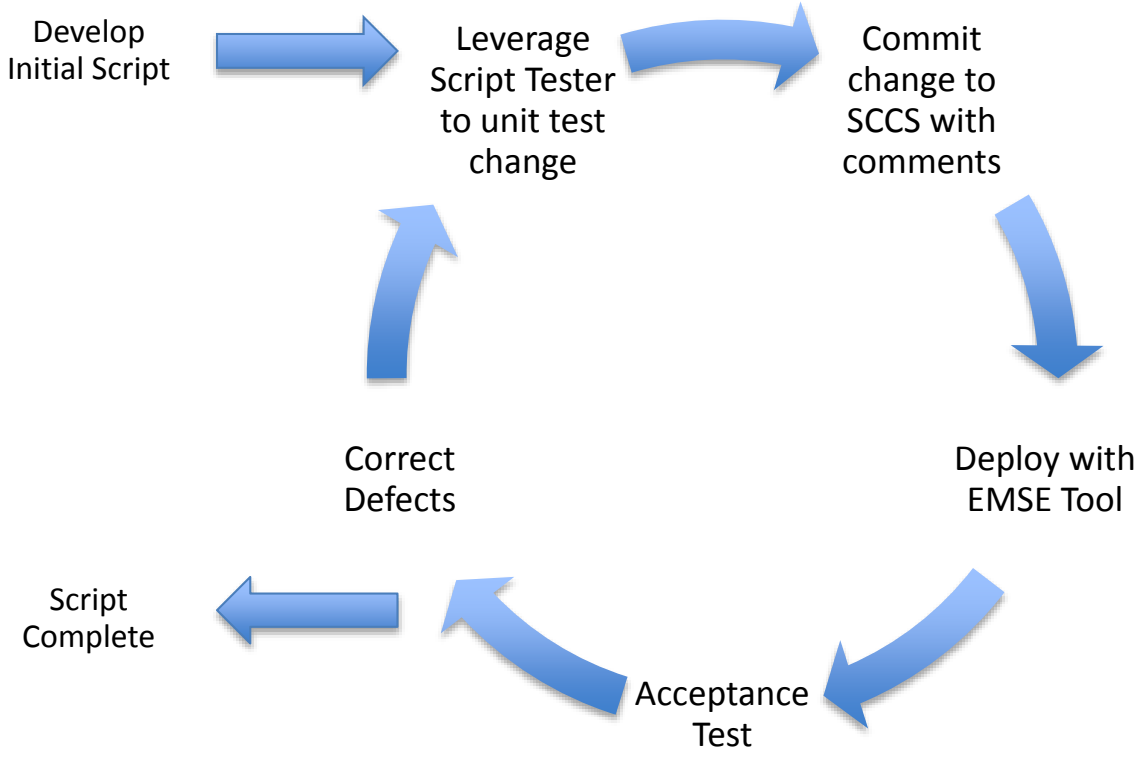
Event → **Master Script** → **Business Script**





Script Development Best Practices

Event Script Development Concept



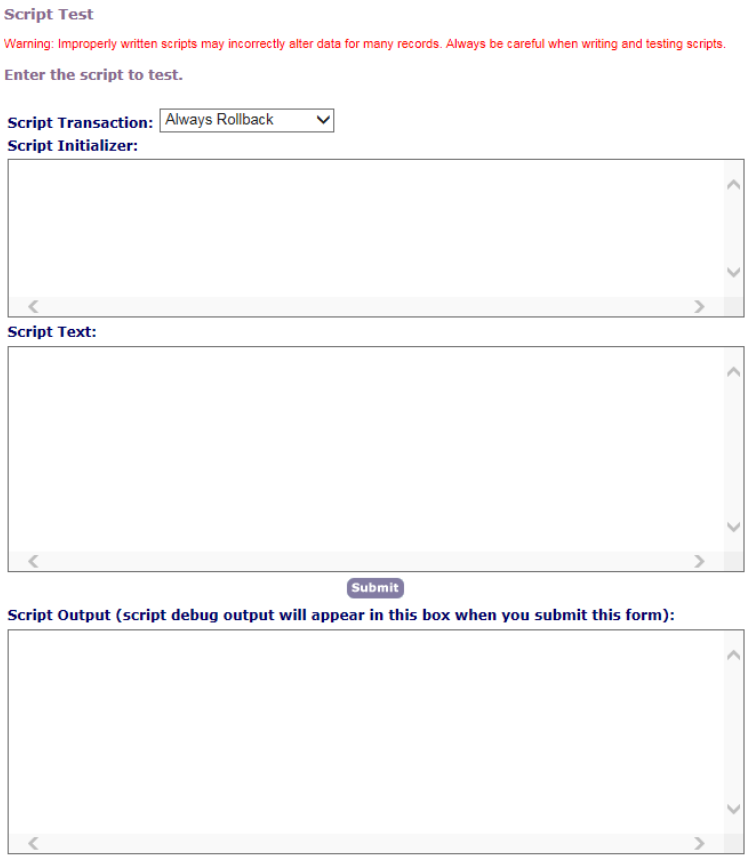
Overview

- What is Script Test?
- What is “Scriptester.js”
- Coding using Script Test
- The development lifecycle
- Testing batch jobs
- Debugging using Script Test

What is Script Test?

- Civic Platform Product feature for immediately executing JavaScript code
- Full access to the EMSE API
- Rolls back database transaction by default

What is Script Test



What is ScripTester.js?

- JavaScript file included in the Master Script 2.0 and 3.0 distribution
- Acts as a mini-Master Script
  - Loads the script environment for a record of your choosing
  - Can simulate the event and run all existing scripts, or use the framework to test code you are developing
- Version 3.0 wraps all user code in try/catch blocks in order to avoid runtime errors

Using Script Tester

```
1  var myCapId = "1";
2  var myUserId = "ADMIN";
3
4  /* ASA  */  var eventName = "ApplicationSubmitAfter";
5  /* WTUA */  //var eventName = "WorkflowTaskUpdateAfter";  wfTask = "Application S
6  /* IRSA */  //var eventName = "InspectionResultSubmitAfter" ; inspResult = "Faile
7  /* ISA  */  //var eventName = "InspectionScheduleAfter"; inspType = "Roofing"
8  /* PRA  */  //var eventName = "PaymentReceiveAfter";
9
10 var useProductScript = false;  // set to true to use the "productized" master scr
11 var runEvent = true; // set to true to simulate the event and run all std choices
12
13 /* master script code don't touch */ aa.env.setValue("EventName",eventName); var
14
15 //
16 // User code goes here
17 //
18
19 try {
20     showDebug = true;
21
22
23     }
24 catch (err) {
25     logDebug("A JavaScript Error occured: " + err.message);
26     }
27 // end user code
28 aa.env.setValue("ScriptReturnCode", "1");   aa.env.setValue("ScriptReturnMessage", debug)
29
30
```

Record ID (AltID) to use for testing

User ID to use when simulating

Event to simulate. Uncomment or input manually.

Set to true to use the productized master scripts.

Set to true to simulate the event and run all std choices / scripts…

…or add your own code to test here

Exercise

- Exercise 3 – Script Tester Intro
- Exercise 4 – Add a Fee
- Exercise 5 – Schedule an Inspection
- Exercise 6 – Create a Related Record!

Exercise

- Exercise 7 – Creating a custom function

Business Script Examples

When a Contract Application workflow task of License Issuance is resulted add a fee:

- Result the “License Issuance” workflow task with a status of “Issued”
- Add the Contractor fee

*WTUA;Licenses!Contractor!~!Application.js

```
1
2 if (wfTask == "License Issuance" && wfStatus == "Issued") {
3     addFee("ContractorFee","ContractorSched","Final",1,"Y");
4 }
5
6
```

Business Script Examples cont.

If there is a balance owed on a License Application do not allow Permit to be issued, and inform the user.

- Result the “Permit Issuance” workflow task with a status of Issue
- Cancel the update if the balance is greater than 0
- Show a message

*WTUB;LICENSE!BUSINESS!APPLICATION!NA.js

```
if(balanceDue > 0 && wfTask == "Permit Issuance" && wfStatus == "Issue"){
    cancel = true;
    showMessage = true;
    comment("You cannot issue this license until all fees are paid.");
}
```

Exercise

- Exercise 2 – Configure a Business Script

Custom Functions

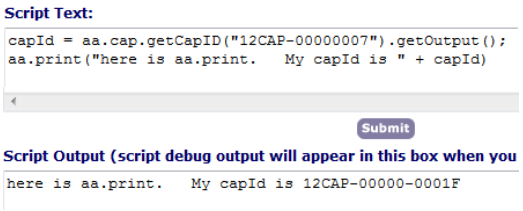
Two Different Master Script Distribution Methods

Non-Productized	Productized
Maintained by Accela Services	Maintained by Accela Engineering
Download from Accela Community (search for “3.0 script distribution”	Installed with the product. 3.0 Scripts will be available soon
Events - Event List Edit Event Associated Script ApplicationSpecificInfoUpdateAfter ApplicationSpecificInfoUpdateAfterV3.0	Events - Event List Edit Event Associated Script ApplicationSubmitBefore ApplicationSubmitBefore (Master Script - 7.2.0)
Prior to 3.0, INCLUDES_CUSTOM script is stored in Events->Scripts 3.0 Scripts have a “useCustomScriptFile” variable that can be set in INCLUDES_CUSTOM_GLOBALS. If true, INCLUDES_CUSTOM script is stored in Events->Custom Script	INCLUDES_CUSTOM script is stored in Events->Custom Script
INCLUDES_ACCELA_FUNCTION, INCLUDES_ACCELA_GLOBALS stored in…	INCLUDES_ACCELA_FUNCTION, INCLUDES_ACCELA_GLOBALS stored in Events->Master Scripts

Debugging Scripts

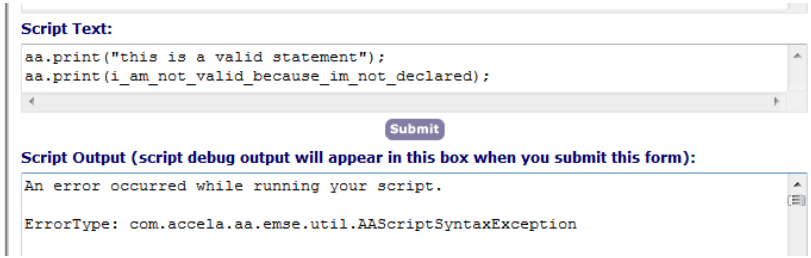
Debugging Options

- aa.print(string)
  - Will display in script test:



  - As well as the bottom of master script debug output

  - In order to see it, the script must successfully complete!



Debugging Options

- aa.debug(string,string)
  - Will only display in the BIZ log file

```
// test the aa.debug statement

aa.debug("here is a debug statement","my capId is " + capId);
```

  - Output

```
2012-04-10 14:21:47,936 INFO [STDOUT] =====EMSE Debug Out===== here is a debug statement : my capId is 12CAP-00000-0001F
```

  - Debug output will appear in the log file, even if the script aborts

Debugging Options

- showDebug(debugLevel)

Let the master script handle debugging for you. Most master script functions will output some debugging messages. The flow of the master script will be logged, including timing.

Levels:	debugLevel = 0 (or false)	// no output
	debugLevel = 1 (or true)	// only screen output
	debugLevel = 2	// output only to biz server log
	debugLevel = 3	// output to screen and biz log

- Default to false for all scripts starting with Master Script release 2.0. Default stored in INCLUDES_ACCELA_GLOBALS
- Starting with Master Scripts 3.0, INCLUDES_CUSTOM_GLOBALS should be used to set debugging options.

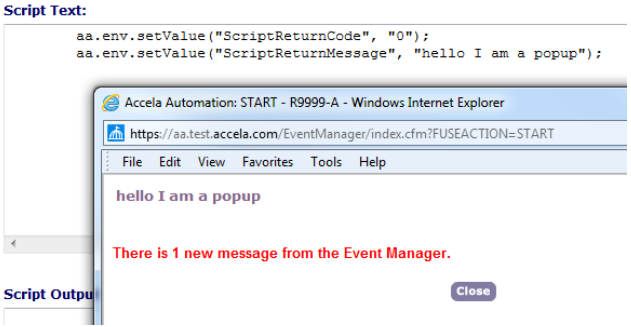
INCLUDES_CUSTOM_GLOBALS.js

```
if (currentUserID == "ADMIN") {
    showDebug = true;
}
```

- Can be set dynamically during script execution, however must be > 0 when the script ends in order to receive a popup.
- Looks for **ERROR** to send abort ScriptReturnCode of 1 back to event.

Script Return Codes

- ScriptReturnCode
- ScriptReturnMessage
  - Variables passed from the script back to the event
  - ScriptReturnMessage controls pop up results. No Text = no Popup
  - ScriptReturnCode controls how AA should react to the completed script. Only 0 and 1 are used in practice.



0	Proceed as normal.
1	Request Accela Automation to stop the current user action and go back to the previous page.
2	Request Accela Automation to stop the current user action and go back to the main menu.
3	Request Accela Automation to stop the current user action and proceed to the page designated by the ScriptReturnRedirection value.
4	Request Accela Automation to stop the current user action and log user out.

Using Try/Catch/Throw – new in 7.3

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors.

- When the JavaScript engine is executing JavaScript code, different errors can occur:
  - It can be syntax errors, typically coding errors or typos made by the programmer.
  - It can be errors due to wrong input

- JavaScript Throws Errors
  - When an error occurs, when something goes wrong, the JavaScript engine will normally stop, and generate an error message.
  - The technical term for this is: JavaScript will **throw** an error.

- JavaScript try and catch
  - The **try** statement allows you to define a block of code to be tested for errors while it is being executed.
  - The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block.
  - The JavaScript statements **try** and **catch** come in pairs.

What’s in your toolkit?

- showDebug(debugLevel)
  - Let the master script handle the toolkit for you
  - Most master script functions will output some debugging messages
  - The flow of the master script will be logged, including timing
  - Levels: debugLevel = 0 (or false) // no output
    debugLevel = 1 (or true) // only screen output
    debugLevel = 2 // output only to biz server log
    debugLevel = 3 // output to screen and biz log
  - Default to false for all scripts in Master Script release 2.0. Default stored in INCLUDES_ACCELA_GLOBALS
  - Can be set dynamically during script execution, However must be > 0 when the script ends in order to receive a popup.
  - Looks for **ERROR** to send abort ScriptReturnCode of 1 back to event.

Reading the Output – Debug Window

EMSE Message List - Windows Internet Explorer

https://avsql.migration.accela.com/portlets/emse/EMSEMessage.do?module=Building

Cancel

Time	Event	Message
03:38:34	ApplicationSpecificInfoUpdateAfter	EMSE Script Framework Versions EVENT TRIGGERED: ApplicationSpecificInfoUpdateAfter SCRIPT EXECUTED: APPSPECIFICINFOUPDATEAFTER2.0 INCLUDE VERSION: 2 SCRIPT VERSION : 2 GLOBAL VERSION : 2

Event that was called, Script that was executed, info about the script

EMSE Script Results for C03-000854

```
capId = class com.accela.aa.aamain.cap.CapIDModel
cap = class com.accela.aa.emse.dom.CapScriptModel
currentUserID = ADMIN
currentUserGroup = BuildingAdmin
systemUserObj = class com.accela.aa.aamain.people.SysUserModel
appTypeString = Building/Combo/Residential/Addition
capName =
capStatus = FINAL
fileDate = 5/30/2030
fileDateYYYYMMDD = 2030-05-30
sysDate = class com.accela.aa.emse.dom.ScriptDateTime
parcelArea = 014500.0
estValue = 0
calcValue = 0
feeFactor = CALC
houseCount = 0
```

Record ID that the event occurred on. Standard variables that are declared by the script, along with their values

Reading the Output – Debug Window

```
{Absorption System (BTU)} = null
{Main Electrical Service to 600V(amps)} = 0
{Green Building Required} = No
{Nursing Home / School} = null
{Soil Bearing Pressure} = 0
{ParcelAttribute.LegalDesc} =
{ParcelAttribute.Block} =
{State Division of Industry Safety Permit Required} = null
{Residential} = null
{Planning Zoning} = null
{Occupancy} = null
```

ASI Fields, TSI Fields, and parcel attributes that are pre-populated by the script

```
ASI Table Array : ENTITLEMENTS (0 Rows)
ASI Table Array : PERMITTEDUSE (1 Rows)
ASI Table Array : CONTRACTINFORMATION (0 Rows)
```

ASI Table variables that are pre-populated by the script

Reading the Output – Debug Window

```
Executing: ApplicationSpecificInfoUpdateAfter, Elapsed Time: 1 Seconds
ADMIN : ApplicationSpecificInfoUpdateAfter : #02 : Criteria : true
ADMIN : ApplicationSpecificInfoUpdateAfter : #02 : Action : branch("ASIA:" + appTypeArray[0] + "/*/*/*")
Executing: ASIA:Planning/*/*/*, Elapsed Time: 1 Seconds
Finished: ASIA:Planning/*/*/*, Elapsed Time: 1 Seconds
```

Stepping through the script controls (standard choice entries)

```
ADMIN : assessMechanicalFees : #22 : Criteria : {Heating Appliance} && {Heating Appliance}!= "" &&
parseInt({Heating Appliance}) > 0
ADMIN : assessMechanicalFees : #22 : Action : updateFee("M_HEATAPP", "BLDG_MECH", "STANDARD",
parseInt({Heating Appliance}), invYN);
Updated Qty on Existing Fee Item: M_HEATAPP to Qty: 1
```

"Action" entries only appear if the criteria evaluates to *true.* Debug messages are also sprinkled in

Common Errors and the Mistakes that Cause Them

Message
(Script Engine) com.accela.aa.emse.util.AAScriptSyntaxException:undefined illegal character (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

Common Errors

The CopyPasta: Pasting text from HTML emails into standard choices

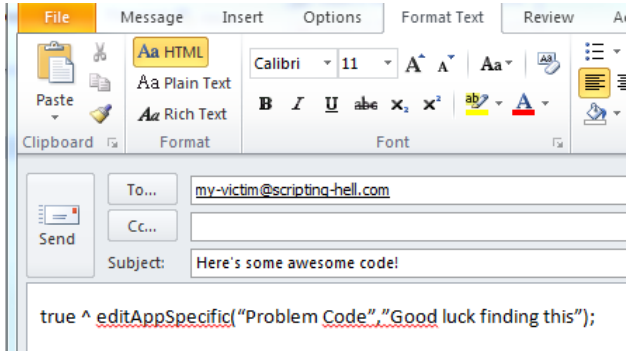
aa.print(“take a look at my nifty quotes”);

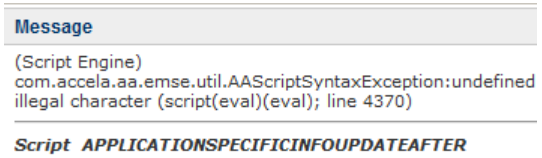
-- vs --

aa.print("these quotes are not so nifty");

Solution(s):

- Use Plain Text when sending code (OK)
- Send code in an attached text file (Better)
- Send code as a data manager extract (Best)





Common Errors

Message

(Script Engine)
com.accela.aa.emse.util.AAScriptSyntaxException:undefined:
The undefined value has no properties. (script(eval)(eval); line 4367)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

Message

(Script Engine)
com.accela.aa.emse.util.AAScriptSyntaxException:undefined:
Cannot convert null to an object. (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

Common Errors and the Mistakes that Cause Them

the undefined value has no properties:

- Attempting to use a property or method “dot” (.) on a variable/object that isn’t defined.
- In most cases, the real question is “why isn’t it defined?”
- Usually because the code that defined the value didn’t work as intended.

Solution:

- Update your code to test variables. Do not assume they populated correctly.

Common Errors and the Mistakes that Cause Them

```
2012-04-12 11:51:52,946 ERROR [com.accela.aa.emse.emse.ScriptRootBusiness] HostSignon3-co
org.mozilla.javascript.JavaScriptException: java.lang.NullPointerException
        at org.mozilla.javascript.JavaScriptException.wrapException(Unknown Source)
        at org.mozilla.javascript.NativeJavaMethod.call(Unknown Source)
```

Common Errors

Null Pointer Exception

- Java from the Biz Server (not Javascript) is attempting to operate on a variable/object that is null.
- Nasty! Your script may abort with no popup, appearing like it will complete successfully. Depends on whether the exception is trapped in the biz server code.
- In most cases, the real question is “why is it null”
- Usually because the code that defined the value didn’t work as intended.

Solution:

- Update your code to test method outputs.

Common Errors

Message

(Script Engine)
com.accela.aa.emse.util.AAScriptSyntaxException:undefined:
"myName" is not defined. (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

Common Errors

Variable is not defined:

Message

(Script Engine)
com.accela.aa.emse.util.AAScriptSyntaxException:undefined: "myName" is not defined. (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

- But I declared it right here!

```
true ^ var myName = "John"; branch("Tell Me What My Name Is");
```

Solution:

- This is a “scope” issue. The “var” keyword tells JavaScript that this is a local variable. From a master script standpoint, it will only be visible within the standard choice.
- If you omit the “var” keyword, the variables you declare will persist for the entire script. Be careful with your naming!
- Custom functions should declare every variable with “var”, unless you are assuming a global variable (such as capId).

Common Errors

Message

(Script Engine)
com.accela.aa.emse.util.AAScriptSyntaxException:org.mozilla.javascript.EvaluatorException:
Can't find method com.accela.aa.emse.dom.CapScript.getCapID(string,string).

Script APPLICATIONSPECIFICINFOUPDATEAFTER

Common Errors

Wrong method name or mismatched parameters

Message

(Script Engine)
com.accela.aa.emse.util.AAScriptSyntaxException:org.mozilla.javascript.EvaluatorException:
Can't find method com.accela.aa.emse.dom.CapScript.getCapID(string,string).

Script APPLICATIONSPECIFICINFOUPDATEAFTER

- Attempting to call a function or method incorrectly

Solution(s)

- Check the function/method name, make sure that it is spelled correctly
- Compare the number of parameters that you are passing, compare to method definition. (e.g., you are passing two and it expects 3)
- Check the data types that you are passing. Compare the error message to method definition. (e.g., It may be expecting a capIdModel, not a string)

Other Common Errors

Message

(Script Engine) com.accela.aa.emse.util.AAScriptSyntaxException:undefined: missing) after argument list (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

- Missing a parenthesis

Message

(Script Engine) com.accela.aa.emse.util.AAScriptSyntaxException:undefined: missing ; before statement (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

- Too many parenthesis

Message

(Script Engine) com.accela.aa.emse.util.AAScriptSyntaxException:undefined: unterminated string literal (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

- Forgot a quote

Message

(Script Engine) com.accela.aa.emse.util.AAScriptSyntaxException:undefined: "capID" is not defined. (script(eval)(eval); line 4370)

Script APPLICATIONSPECIFICINFOUPDATEAFTER

- Misspelled a variable. Watch your case!

Exercise

- Exercise 3 – Debugging Options

Questions?

John Schomp
Director, Accela Professional Services

Unused Slides

Loading Scripts

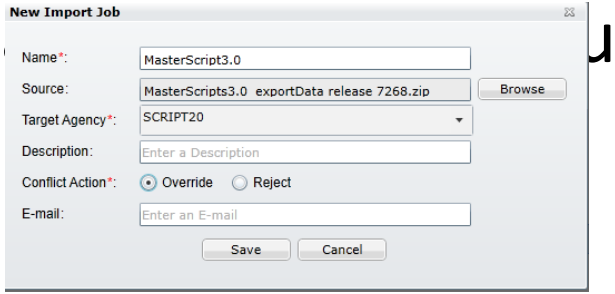
- Option 1: Import scripts and config std choices automatically via Data Manager (preferred!)
- Option 2: Manually (zip file)
- Link events to new master scripts

Two Different Master Script Distribution Methods

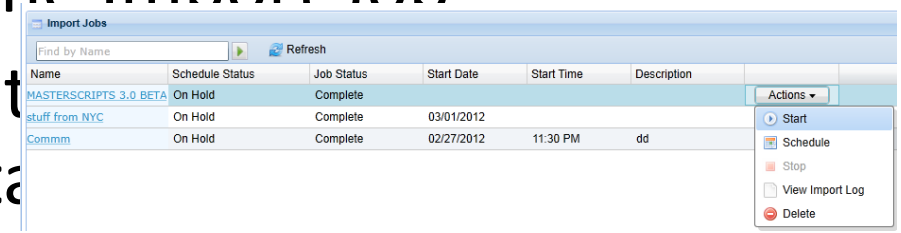
Non-Productized	Productized
Maintained by Accela Services	Maintained by Accela Engineering
Download from Accela Community (search for “3.0 script distribution”	Installed with the product. 3.0 Scripts will be available soon
Events - Event List Edit Event Associated Script ApplicationSpecificInfoUpdateAfter ApplicationSpecificInfoUpdateAfterV3.0	Events - Event List Edit Event Associated Script ApplicationSubmitBefore ApplicationSubmitBefore (Master Script - 7.2.0)
Prior to 3.0, INCLUDES_CUSTOM script is stored in Events->Scripts 3.0 Scripts have a “useCustomScriptFile” variable that can be set in INCLUDES_CUSTOM_GLOBALS. If true, INCLUDES_CUSTOM script is stored in Events->Custom Script	INCLUDES_CUSTOM script is stored in Events->Custom Script
INCLUDES_ACCELA_FUNCTION, INCLUDES_ACCELA_GLOBALS stored in…	INCLUDES_ACCELA_FUNCTION, INCLUDES_ACCELA_GLOBALS stored in Events->Master Scripts

Deploy with Data Manager

- Go to AA Admin
- Create a new Import Job and Select the MasterScripts3.0 zip file loc ute MISC folder



- Ensure override is selected
- Save the Import Job
- Go to t and sta



Configurations (1 of 2)

Activates scripts, standard choices, or both

Standard Choices Item Name: EMSE_EXECUTE_OPTIONS
Description: (250 char max)

Status: Enable Disable
Type: System Switch Shared drop-down EMSE Business Configuration

Standard Choices Value	Value Desc	Active
SCRIPT		☑
STD_CHOICE		☐

Standard Choices Item Name: MULTI_SERVICE_SETTINGS
Description: (250 char max)

Status: Enable Disable
Type: System Switch Shared drop-dowr

Standard Choices Value	Value Desc
AGENCY_LOGO_TYPE	USERINFOLOGO
ALLOW_SERVICE_LOCK_SE	No
IS_SUPER_AGENCY	No
SUPER_AGENCY_FOR_EMSE	NYELS
SUPER_AGENCY_INCLUDE_S	INCLUDES_CUSTOM_ENTERPRISE

Optional custom include file for all agencies within a superagency

Standard Choices Item Name: EMSE_VARIABLE_BRANCH_PREFIX
Description: (250 char max)

Status: Enable Disable
Type: System Switch Shared drop-dov

Standard Choices Value	Value Desc
ApplicationConditionAddAfter	ACAA
ApplicationConditionDeleteAfter	ACDA
ApplicationConditionOfApproval	ACUA
ApplicationConditionUpdateAfte	ACUA
ApplicationDetailUpdateAfter	ADUA

Event to prefix mapping, used to locate scripts for each event

Configurations (2 of 2)

Script Code: INCLUDES_CUSTOM_GLOBALS

Script Title: INCLUDES_CUSTOM_GLOBALS

Script Initializer:

Script Text:

```
showDebug = false;

if (currentUserID == "SAXTHELM") showDebug = 3;
if (publicUserID == "PUBLICUSER51") showDebug = 3;

var envName = "DEV";
var sysFromEmail = "dos_noreply@eLicensing.ny.gov";
var acaUrl = lookup("ACA_CONFIGS","OFFICIAL_WEBSITE_URL");
var schoolVerifEmail = false;
var LICENSESTATE = "NY";
```

Store custom global declarations in this optional script

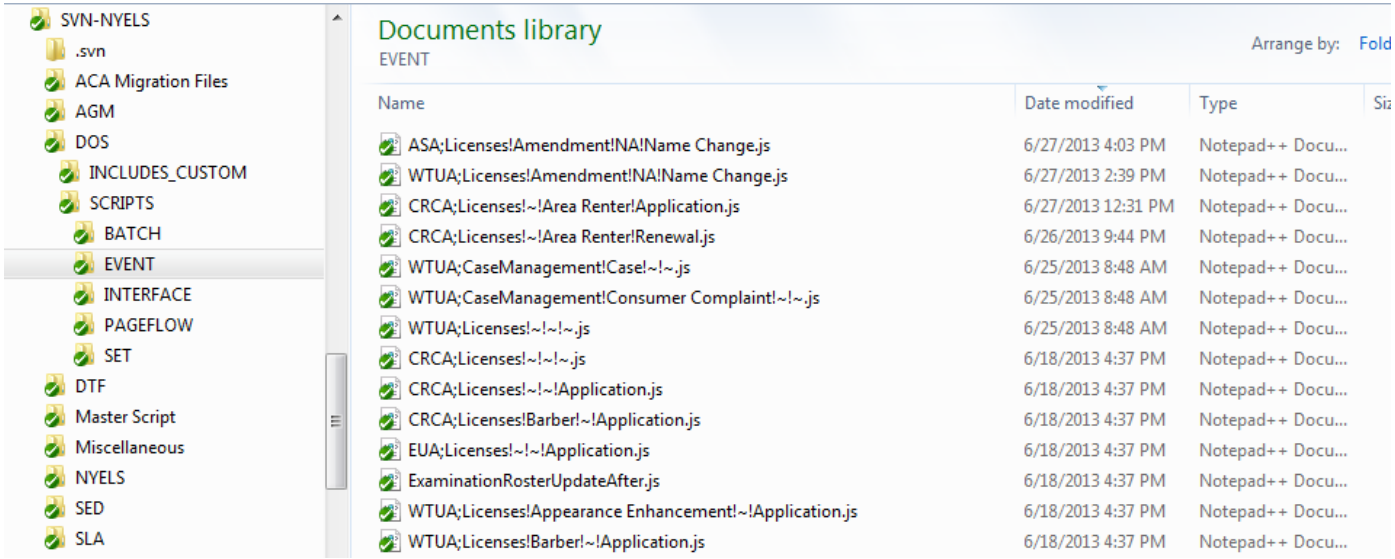
Events - Event List

Edit	Event	Associated Script
•	ApplicationConditionAddAfter	ApplicationConditionAddAfterV3.0
•	ApplicationConditionDeleteAfter	UniversalMasterScriptV3.0
•	ApplicationConditionOfApprovalUpdateAfter	ApplicationConditionUpdateAfterV3.0
•	ApplicationConditionUpdateAfter	ApplicationConditionUpdateAfterV3.0
•	ApplicationDetailUpdateAfter	UniversalMasterScriptV3.0
•	ApplicationSpecificInfoUpdateAfter	ApplicationSpecificInfoUpdateAfterV3.0
•	ApplicationStatusUpdateAfter	ApplicationStatusUpdateAfterV3.0
•	ApplicationSubmitAfter	ApplicationSubmitAfterV3.0
•	ApplicationSubmitBefore	ApplicationSubmitBeforeV3.0

Install:

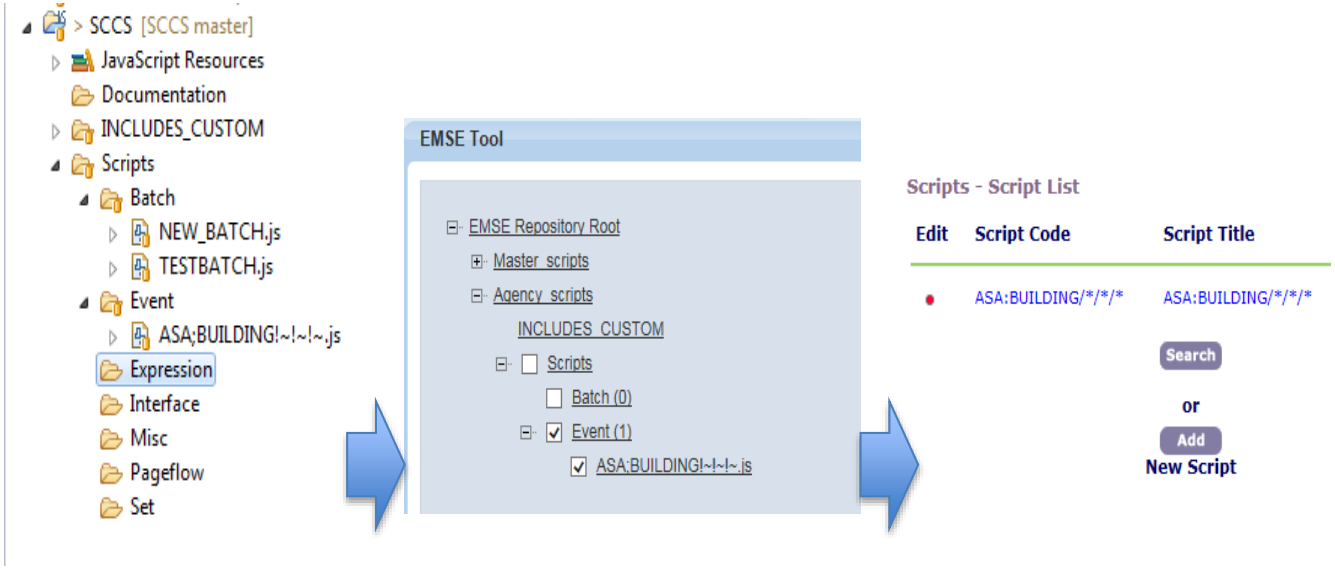
- INCLUDES_CUSTOM
- INCLUDES_ACCELA_FUNCTIONS
- INCLUDES_ACCELA_FUNCTIONS_ASB
- INCLUDES_ACCELA_GLOBALS
- Master scripts as needed, assigned to matching event

Screen – Local Script Directory



Use the sample folder structure for script and function storage

Deploying Code to Civic Platform



Store scripts in the file system using naming convention

Load to Events->Scripts, named exactly as variable branching

Sample logDebug Override

```
function logDebug(dstr) {
    vLevel = 1
    if (arguments.length > 1)
        vLevel = arguments[1];
    if ((showDebug & vLevel) == vLevel || vLevel == 1)
        debug += dstr + br;
    if ((showDebug & vLevel) == vLevel)
        aa.debug(aa.getServiceProviderCode() + " : " + aa.env.getValue("CurrentUserID"), dstr);

    try {
    if ((String(dstr).indexOf("**ERROR") > -1) || (String(dstr).indexOf("**INFODEBUG") > -1)) {
        var errorEmail = lookup("DEC_CONFIG","SCRIPT_ERROR_EMAILS");
        if (errorEmail) {
            if (typeof(capId) == "object") {
                var subject = "Script Error on Record:" + capId.getCustomID() + " Site:" + lookup("ACA_CONFIGS","ACA_SITE");
                }
            else
                var subject = "Script Error Site:" + lookup("ACA_CONFIGS","ACA_SITE");
                }

            var msg = "Event Name :" + aa.env.getValue("EventName") + "<br>";
            msg+= "Script Code:" + aa.env.getValue("ScriptCode") + "<br>";
            msg+= "User ID    :" + aa.env.getValue("CurrentUserID") + "<br>";
            msg+= "Error      :" + dstr + "<br>";
            msg+= "Stack Trace<br><br>";
            msg+=debug;

            aa.sendMail("noreply@accela.com", errorEmail, "", subject, msg);
        }
    }
    catch(err) { aa.debug("emse","couldn't send email for script error : " + err.message) }
```