Scripting: Beyond the Basics

Seth Axthelm, Consulting Manager
Accela, Inc.

Overview

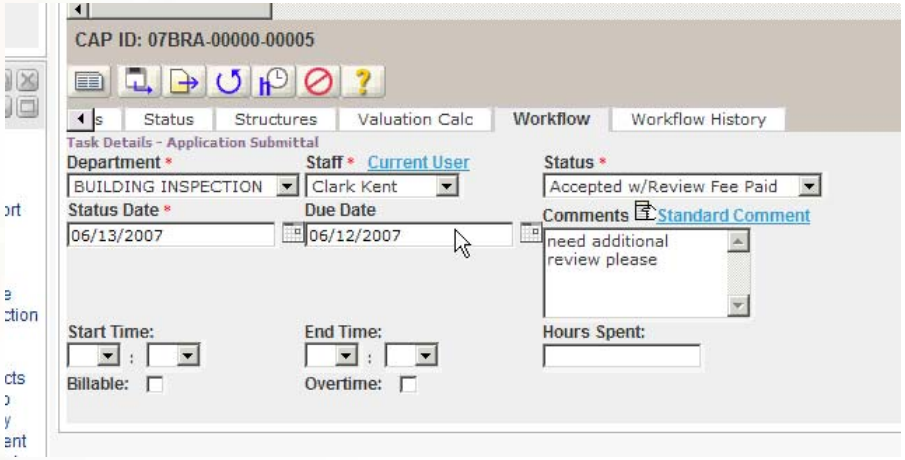
- Quick EMSE Overview
- Variable Branching
- Exercises
  - Setting Up Variable Branching
  - GIS scripting
  - ASI Tables
  - Complex Fees

What is EMSE?

- EMSE = Event Manager and Script Engine
  - Core Accela Automation® feature
  - Work together to allow customization to Accela Automation
  - Event Manager controls timing
  - Script Engine executes actions

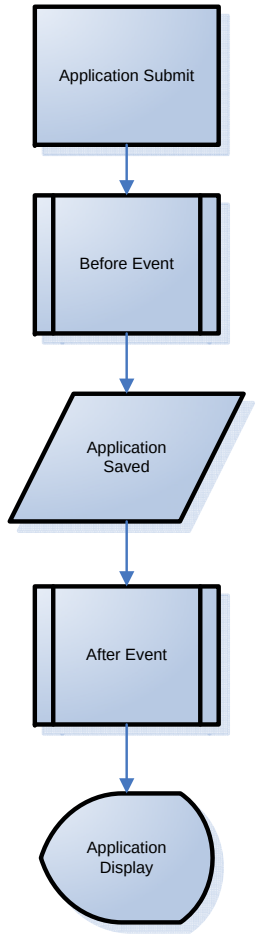
What is an Event?

- Action triggered by a user
- “Submit” button on a form
- Saving data
- 76 unique events defined in Accela Automation (20 added from last year)



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

```
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;
                }
```

Master Scripts

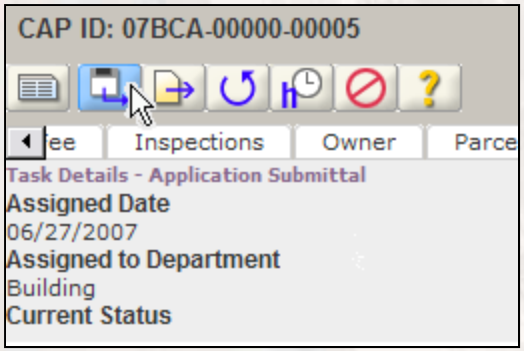
- Standardized scripts
- One script per event
- *Standard Choice* entries control the script
- Modular, plug-in functions
- Minimal JavaScript knowledge required

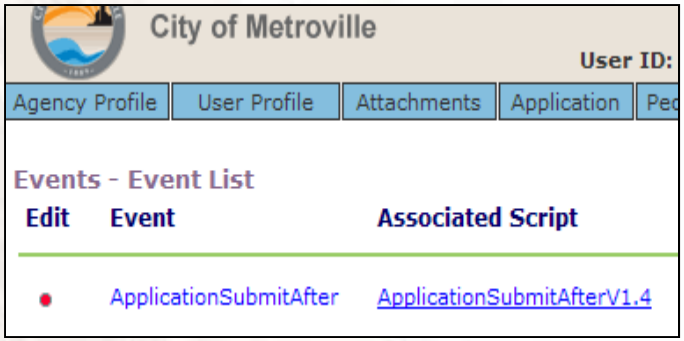
Events with Available Master Scripts

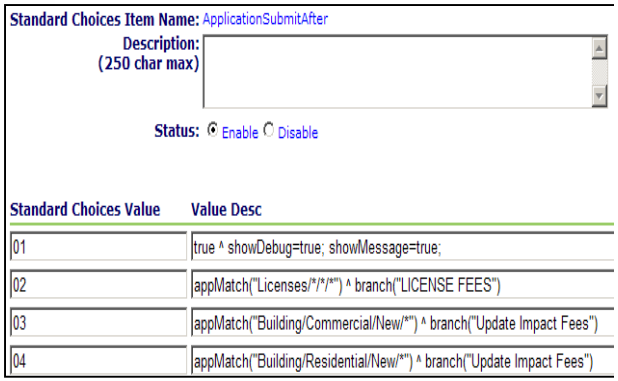
ApplicationConditionAddAfter	InspectionScheduleAfter
ApplicationSpecificInfoUpdateAfter	InspectionScheduleBefore
ApplicationSpecificInfoUpdateBefore	InvoiceFeeAfter
ApplicationStatusUpdateAfter	LicProfUpdateBefore
ApplicationSubmitAfter	ParcelAddAfter
ApplicationSubmitBefore	ParcelUpdateAfter
ContactAddAfter	PaymentReceiveAfter
ContactAddBefore	PaymentReceiveBefore
ContactEditAfter	RenewalInfoUpdateAfter
ContactRemoveAfter	**V360InspectionResultSubmitAfter**
FeeAssessAfter	**WorkflowTaskUpdateAfter**
FeeAssessBefore	**WorkflowTaskUpdateBefore**
InspectionResultSubmitAfter	

Master Script Flow Execution

Event → **Master Script** → **Script Controls**

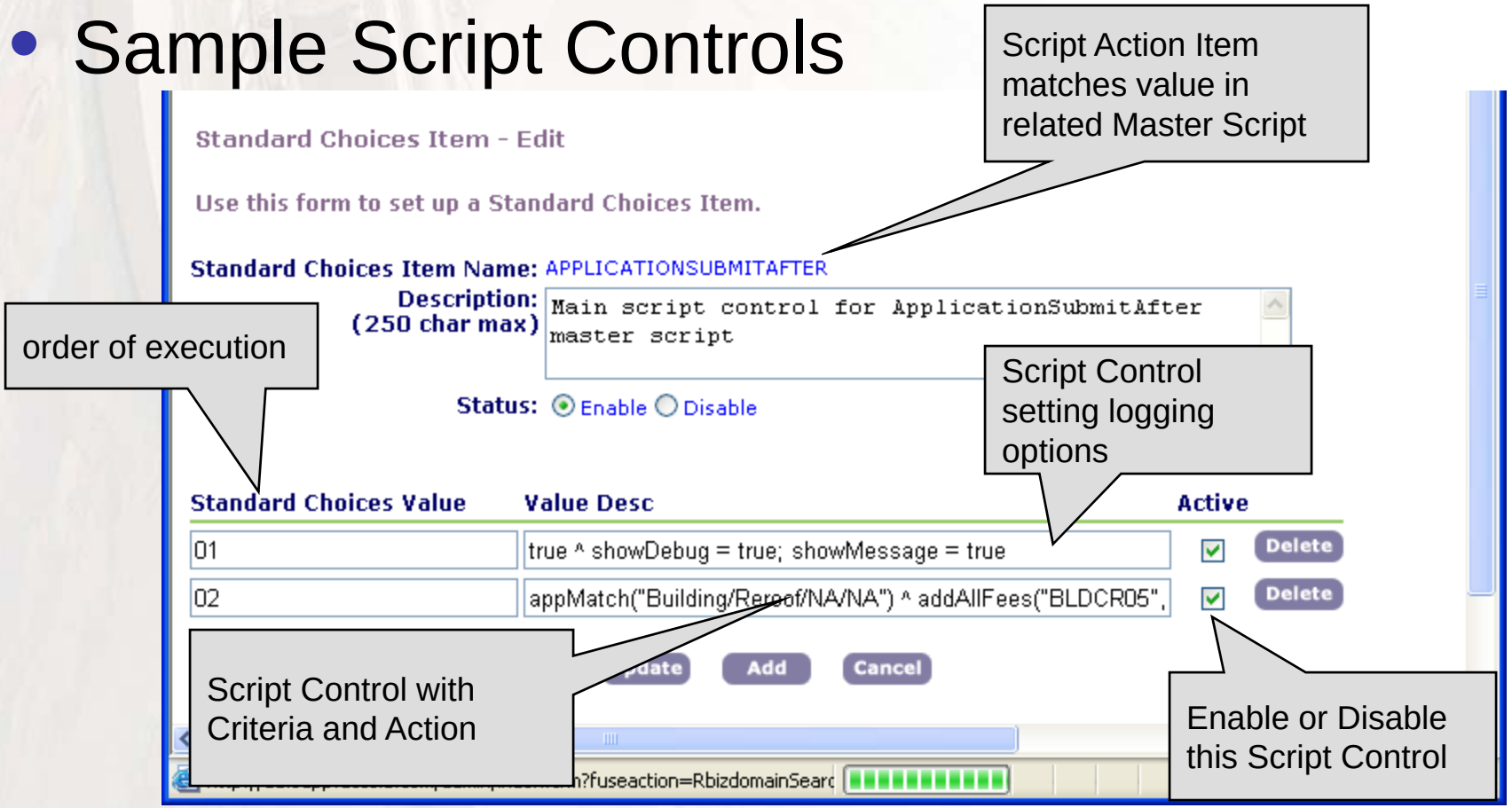






Script Control Design

- Sample Script Controls



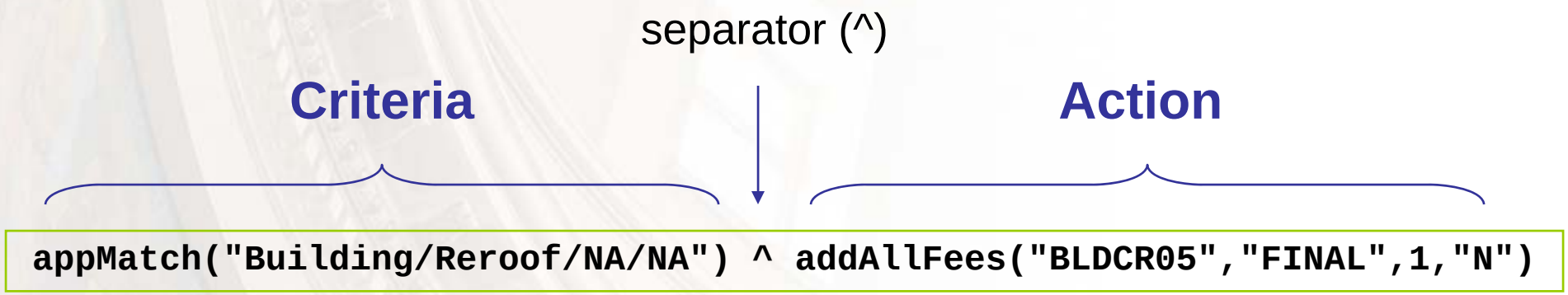
Script Control Basics

- Script Controls
  - Are case-sensitive!
  - Do all the work. Criteria tells when, Action tells what
  - Criteria and Actions separated by a caret (^)
  - Must be valid JavaScript

Script Control Basics

- Use curly braces "{" and "}" to reference
  - Application Specific Info fields
  - Task Specific Info fields
  - Parcel custom attributes
- User “” around strings
- Standard Choices must be numbered sequentially

Script Control Structure



IF this is true,

THEN do this

Script Control Structure - Else

IF this is true,

THEN do this

Else do this

Master Scripts Criteria

- **Criteria**
  - Must evaluate to only "True" or "False“
  - May contain multiple statements, separated by logical operators

Master Scripts Criteria Example

- true
- appMatch(“Building/*/*/*”)
- inspType == “Final Inspection”
- {Government Funded} == "Yes"
- {Building Type} != “Historical"
- {ParcelAttribute.Neighborhood} == "Downtown Area”
- feeExists(“LICFEE”) || balanceDue == 0
- inspType == “Final” && !isScheduled(“Electrical”)
- proximity(“GIS”,”Schools”,parseInt({Number of feet}))
- taskStatus(“License Issuance”,”Issued”)

Master Scripts Actions

- Action
  - Usually call functions developed for common actions
  - May contain multiple statements (separated by semi-colon)
    - addFee(); scheduleInspection(); etc…
  - Can span multiple lines
    - true ^ addFee()
    - ^ scheduleInspection();
    - ^ UpdateAppStatus();
  - May call subroutines using the branch() function

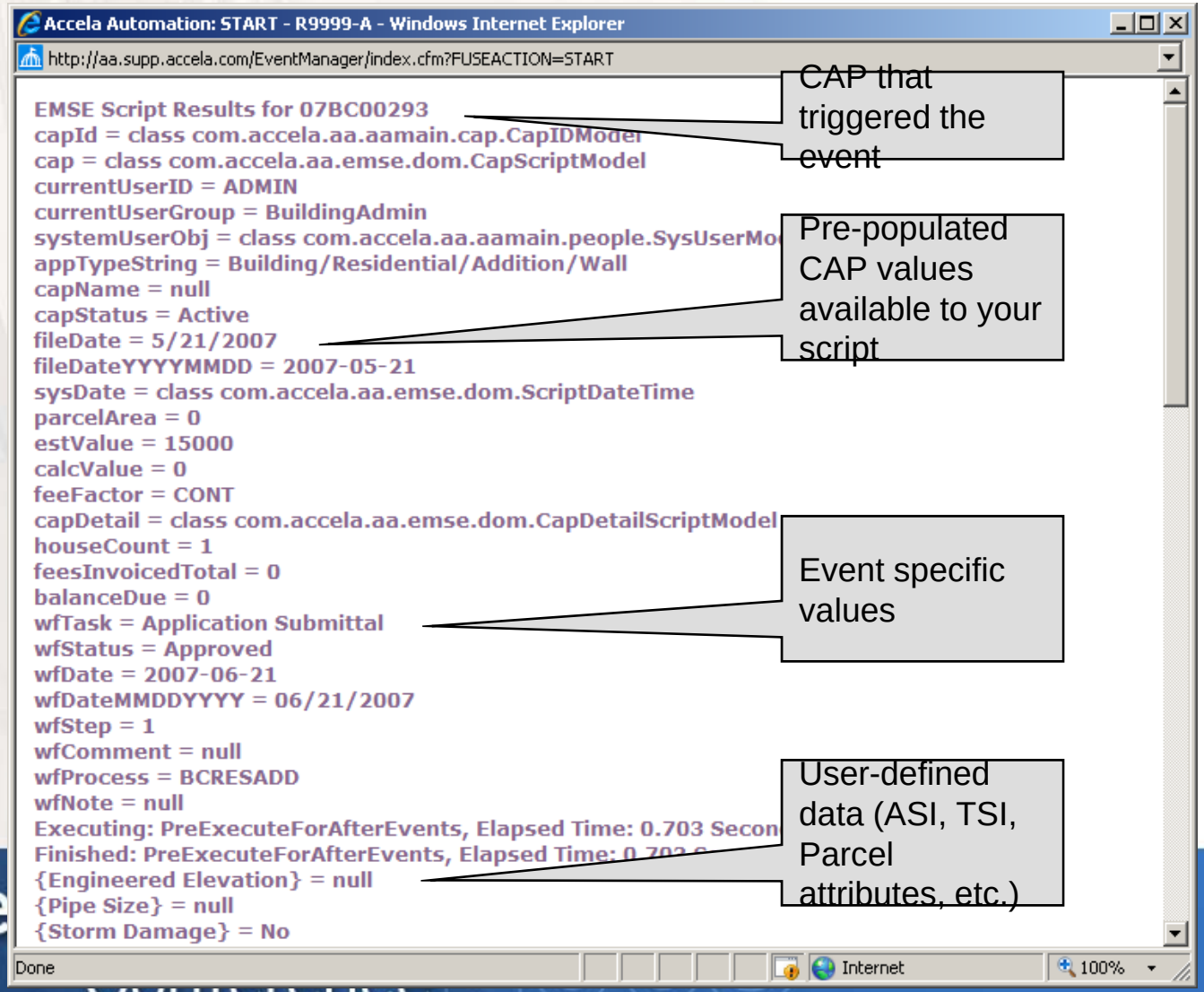
Master Scripts Action Examples

- activateTask(“Plan Review”)
- addFees(“RENEW”,“LICENSE”,”PER2005”,1,”Y”,childApp)
- addAppCondition("Permit","Applied","Reinspection Fee","Re-Inspection Fee","Hold")
- closeTask(“Peer Review”,”NA”,”Closed by Script”,”Closed by Script”)
- comment(“Square footage is “ + {Sq Ft})
- childApp = createChild(“Building”,”Commercial”,”Plumb”,”NA”,””)
- editAppSpecific(“Total Value”,{Sq Ft} * {Price per Sq Ft})
- branch(“CALCVALUES”)

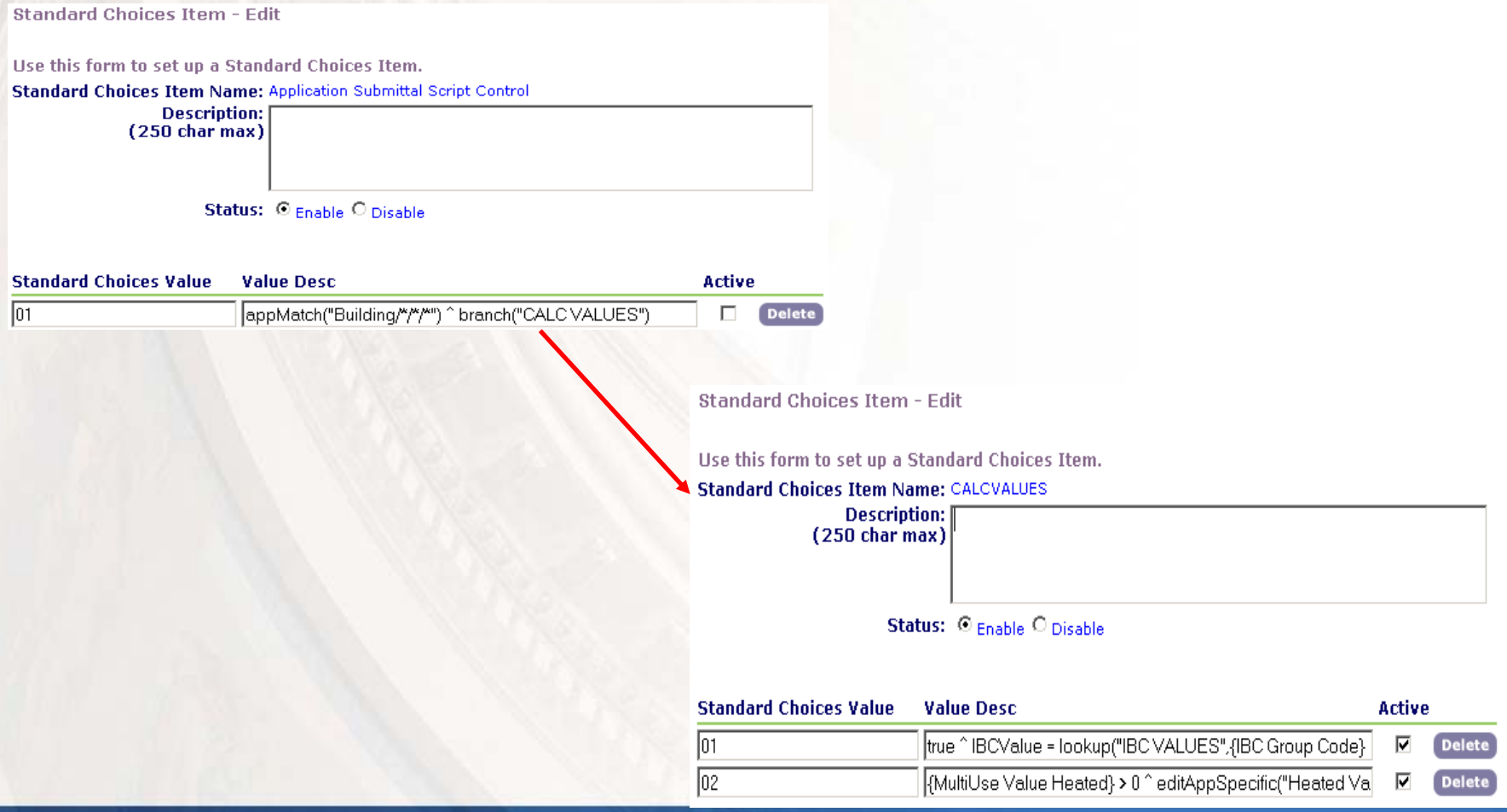
Master Scripts Global Variables

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

Master Scripts Debug Window



Master Scripts Branching



Variable Branching Overview

- In Master Script versions prior to 1.4, branches needed to be hard coded:

  true ^ branch(“Assess Fees”)

- As of Master Script version 1.5, new feature Variable Branching allows us to create the branch destination using variables:

  true ^ branch(variableName1 +variableName2)

Turning on Variable Branching

- To make use of this functionality the following flag needs to be set to true in each script you wish to use it in:

var enableVariableBranching = true;

- Note: when using variable branching, the documentation functionality of the scripts no longer works

So What Does This Mean?

- Many uses for variable branching
- Use the CAP type as a part of the variable branching.
- Allows us to put the following script controls for an event (ie. ApplicationSubmitAfter)

```
true ^ branch("ASA:" + appTypeArray[0] + "/*/*/*")
true ^ branch("ASA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/*/*")
true ^ branch("ASA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/" +
appTypeArray[2] + "/*")
true ^ branch("ASA:" + appTypeString)
```

So What Does This Mean?

- Which for CAP Type, “Permits/Drainage/Non-Residential/NA”, would, based on the code above, branch to the following Standard Choices:

  ASA:Permits/*/*/*

  ASA:Permits/Drainage/*/*

  ASA:Permits/Drainage/Non-Residential/*

  ASA:Permits/Drainage/Non-Residential/NA

So What Does That Mean?

- Put your scripts based on where they need to be called from

ASA:Permits/*/*/*

(Any script controls put in this Standard Choice would be executed for all CAP types in the Permits module)

ASA:Permits/Drainage/*/*

(Any script controls put in this Standard Choice would be executed for any CAP types that fall under the Permits/Drainage/ type)

ASA:Permits/Drainage/Non-Residential/*

(Any script controls put in this Standard Choice would be executed for any CAP types that fall under the Permits/Drainage/Residential/ sub-type)

ASA:Permits/Drainage/Non-Residential/NA

(Any script controls put in this Standard Choice would be executed for any CAP types that fall under the Permits/Drainage/Non-Residential/NA cap type)

What About Other Events?

- We have seen that the Standard Choice before was prefixed by “ASA” standing for ApplicationSubmitAfter. We hard coded that in the Standard Choice for that event.
- In other events you will want to hard code a different prefix for that event’s Standard Choice.

What About Other Events?

- For example, for the WorkflowTaskUpdateAfter event you would have the following entries:

  true ^ branch(“WTUA:" + appTypeArray[0] + "/*/*/*")
  true ^ branch(“WTUA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/*/*")
  true ^ branch(“WTUA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/" + appTypeArray[2] + "/*")
  true ^ branch(“WTUA:" + appTypeString)

- It is the same except for the prefix that is entered.

What About Other Events?

- Abbreviations used as prefixes for each of the events:

Event	Abbrev	Event	Abbrev
ApplicationConditionAddAfter	ACAA	InspectionScheduleAfter	ISA
ApplicationSpecificInfoUpdateAfter	ASIUA	InspectionScheduleBefore	ISB
ApplicationSpecificInfoUpdateBefore	ASIUB	InvoiceFeeAfter	IFA
ApplicationStatusUpdateAfter	ASUA	LicProfUpdateBefore	LPUB
ApplicationSubmitAfter	ASA	ParcelAddAfter	PAA
ApplicationSubmitBefore	ASB	ParcelUpdateAfter	PUA
ContactAddAfter	CAA	PaymentReceiveAfter	PRA
ContactAddBefore	CAB	PaymentReceiveBefore	PRB
ContactEditAfter	CEA	RenewalInfoUpdateAfter	RIUA
ContactRemoveAfter	CRA	V360InspectionResultSubmitAfter	VIRSA
FeeAssessAfter	FAA	WorkflowTaskUpdateAfter	WTUA
FeeAssessBefore	FAB	WorkflowTaskUpdateBefore	WTUB
InspectionResultSubmitAfter	IRSA		

Exercises

- Handouts
- Exercises
  - Setting up variable branching
  - GIS scripting
  - ASI Table interactions
  - Inter-cap scripting
  - Complex fees

Questions?

Seth Axthelm

saxthelm@accela.com