Getting Started

- Login into Accela Automation
  - Agency: scripta
  - User: userXX (where XX is your user number, ie. user03 or user18)
  - Password: Same as user. (all capital)

Setting up Variable Branching

- The master scripts and variable branching have already been installed for the training class so please follow along.

- Turn on the variable branching functionality.
  - Go to Admin Tools -> Events -> Scripts
  - Click "Submit" to get a list of Scripts
  - Choose any of the installed scripts
  - Locate the line:
    - var enableVariableBranching = false;
  - Change the false to true;
  - Click "Save"

- Add script controls to control string stand choices:
  - For each event that master scripts are installed for add the script controls to dynamically branch based on the event and cap type, replacing the branch prefix based on the event:

Standard Choice: ApplicationSubmitAfter

```
01    true ^ showMessage = false; showDebug = false;
02    true ^ branch("ASA:" + appTypeArray[0] + "/*/*/*")
03    true ^ branch("ASA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/*/*")
04    true ^ branch("ASA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/" +
      appTypeArray[2] + "/*")
05    true ^ branch("ASA:" + appTypeString)
```

Standard Choice: WorkflowTaskUpdateAfter

```
01      true ^ showMessage = false; showDebug = false;
02      true ^ branch("WTUA:" + appTypeArray[0] + "/*/*/*")
03      true ^ branch("WTUA:" + appTypeArray[0] + "/" + appTypeArray[1] +
        "/*/*")
04      true ^ branch("WTUA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/"
        + appTypeArray[2] + "/*")
05      true ^ branch("WTUA:" + appTypeString)
```

Standard Choice: InspectionResultSubmitAfter

```
01      true ^ showMessage = false; showDebug = false;
02      true ^ branch("IRSA:" + appTypeArray[0] + "/*/*/*")
03      true ^ branch("IRSA:" + appTypeArray[0] + "/" + appTypeArray[1] +
        "/*/*")
04      true ^ branch("IRSA:" + appTypeArray[0] + "/" + appTypeArray[1] + "/" +
        appTypeArray[2] + "/*")
05      true ^ branch("IRSA:" + appTypeString)
```

Standard Choices Setup

- For this class Cap Types have already been setup for each user.
  - Cap Type: Building/User 15/NA/NA
  - Master Script Branches:
    - ApplicationSubmitAfter -> ASA:Building/User 15/NA/NA
    - WorkflowTaskUpdateAfter -> WTUA:Building/User 15/NA/NA
    - InspectionResultSubmitAfter -> IRSA:Building/User 15/NA/NA
- All exercises in this class will be coded from the above standard choices. Be sure when creating test caps that you create the correct one based on your User number.

Exercise 1 – ApplicationSubmitAfter – Calculating Total from a Table

Objective:

- Looping through an Application Specific Info Table to obtain a total calculation.

Exercise:

- For the event Application Submit After, perform the following:
  - Loop through the Application Specific Info Table "ROOM DIMENSIONS" to obtain the sum total square footage from every room identified by multiplying "Length" * "Width" on each row.
  - Update the Application Specific Info field "Total Sq Ft" with the total calculated form looping through the "ROOM DIMENSIONS" table.

Functions:

- The following function(s) are required to complete this exercise:
  - editAppSpecific(itemName,itemValue)
    - itemName (string) – App Spec Info field to edit
    - itemValue (string) - Value that the app spec info field itemName should be changed to.

Exercise Notes:

- Standard Choice: ASA:Building/User XX/NA/NA
- Use the typeof() function to test if the ROOM DIMENSIONS table exists.
  - typeof(ROOMDIMENSIONS) == "objet" ^ *do something*
- When referring to an ASI table in script controls remove all spaces and special characters from the ASI table name, for example "ROOM DIMENSIONS" becomes "ROOMDIMENSIONS".
- When looping through a table you must use a for loop and branch to another standard choice for each row, this is done like this:
  - typeof(ROOMDIMENSIONS) == "object" ^ for(x in ROOMDIMENSIONS) branch("XXX")
- When branching in a for loop good practice is to branch to a standard choice whose name is a variant of the current one, for example:
  - for(x in ROOMDIMENSIONS) branch("ASA:Building/User XX/NA/NA:LOOP")
- In JavaScript you can use the "+=" in the loop to continue to add the row total to the grand total.
- Columns in each row are referenced as an associative array:
  - ROOMDIMENSIONS[x]["Length"]
  - ROOMDIMENSIONS[x]["Width"]

Exercise 2 – Application Submit After – Working with Proximity Alerts

Objective:

- Query Accela GIS to determine if selected parcel is certain distance from a particular object.

Exercise:

- For the event Application Submit After, perform the following:
  - Check the current parcel's proximity to any water lines within 100 feet
  - If parcel is within 100 feet of water lines, display a message to the user that says, "Current parcel is within 100 feet of city water lines".

Functions:

- The following function(s) are required to complete this exercise:
  - comment(cstr)
    - cstr (string) – Message to display to user
  - proximity(service,layer,distance,[unit])
    - service (string) - GIS Service name
    - layer (string) - GIS layer, i.e., object that function is testing proximity to
    - distance (integer) - Distance of parcel on current app to the GIS object identified by layer
    - unit (string)(optional) - Unit for distance (defaults to feet if not provided)

Exercise Notes:

- Standard Choice: ASA:Building/User XX/NA/NA
- The map service we will be using is "flagstaff"
- The layer we will be checking is "Water_Lines" (This will always be the layer name from the table of contents on the AGIS map)
- The distance parameter is an integer so it does not need to be surrounded by double quotes, use the distance of 100.
- The proximity function returns a true or false
  - Only display a message if proximity returns true. showMessage must be set to true in order to pop up a message to the user.
- Test with the address 3201 N Dyer St.

Exercise 3 – Application Submit After – Retrieving Attribute Information from AGIS for current parcel

Objective:

- Query Accela GIS to retrieve attribute information for the current parcel.

Exercise:

- For the event Application Submit After, perform the following:
  - Retrieve the zoning classification for the parcel selected in the application from AGIS.
  - Edit the Application Specific Info field "Zoning Classification" with the value from AGIS.

Functions:

- The following function(s) are required to complete this exercise:
  - getGISInfo(service,layer,attribute)
    - service (string) - GIS Service name
    - layer (string) - GIS layer.
    - attribute (string) - Name of attribute to retrieve.
  - editAppSpecific(itemName,itemValue)
    - itemName (string) – App Spec Info field to edit
    - itemValue (string) - Value that the app spec info field itemName should be changed to.

Exercise Notes:

- Standard Choice: ASA:Building/User XX/NA/NA
- The map service we will be using is "flagstaff"
- The layer we will be retrieving an attribute from is "Parcels" (This will always be the layer name from the table of contents on the AGIS map)
- The attribute name we will be retrieving is "ZONING" (This will needs to match the spelling and case of the attribute from AGIS)
- Test with the address 3201 N Dyer St.

Exercise 4 – Application Submit After – Retrieving Attribute Information from AGIS within a buffer

Objective:

- Query Accela GIS to retrieve attribute information for any parcels within a buffer.

Exercise:

- For the event Application Submit After, perform the following:
  - Retrieve the PIN of all parcels located within 500 feet of the parcel selected in the application from AGIS.
  - Display a message to the end user listing out the retrieve PINs that starts with, "The following PINs are located within 500 feet".

Functions:

- The following function(s) are required to complete this exercise:
  - getGISBufferInfo(service,layer,distance,attribute$_1$,attribute$_n$,…)
    - service (string) - GIS Service name
    - layer (string) - GIS layer.
    - distance (integer) - Distance (in feet) of GIS objects on CAP to attributes in layer
    - attribute$_1$…attribute$_n$ (strings)(optional) - Additional attributes of the GIS layer to retrieve.
  - comment(cstr)
    - cstr (string) – Message to display to user

Exercise Notes:

- Standard Choice: ASA:Building/User XX/NA/NA
- The map service we will be using is "flagstaff"
- The layer we will be retrieving an attribute from is "Parcels" (This will always be the layer name from the table of contents on the AGIS map)
- The distance for this exercise will be 500
- The attribute name we will be retrieving is "PIN" (This will needs to match the spelling and case of the attribute from AGIS)
- Test with the address 3201 N Dyer St.
- Use a for loop to display the results to the end user like this:
  - showMessage = true; for(z in bufferArray) comment(bufferArray[z]["PIN"]);
  - Because there is only one action within the body of the for loop there is no need to branch to another standard choice as { } are not required.

Exercise 5 – WorkflowTaskUpdateAfter – Assess/Invoice Complex Fee

Objective:

- Assess/Invoice a fee after a workflow task update after event that cannot be calculated by built in Accela Automation fee formulas.

Exercise:

- For the event Workflow Task Update After, perform the following:
  - Upon the task "Initial Review" being update to the status of "Detailed Review Required":
    - Calculate, assess and invoice the "Detailed Review" fee.
    - The "Detailed Review" fee is calculated as, (1 % of Total Sq Ft)^2 rounded up to the nearest integer.

Functions:

- The following function(s) are required to complete this exercise:
  - addFee(fcode,fsched,fperiod,fqty,finvoice)
    - fcode (string) – Fee code to be added
    - fsched (string) – Fee schedule of the fee to be added
    - fperiod (string) – Fee period to be used
    - fqty (integer) – Quantity to be added
    - finvoice (string) – Flag for invoicing ("Y" or "N")

Exercise Notes:

- Standard Choice: WTUA:Building/User XX/NA/NA
- Application Fee Details
  - fcode = BLD_REV
  - fsched = BLD
  - fperiod = FINAL
  - fqty = calculated fee
  - finvoice = Y
- When using calling a function remember that when hard coding a string in the call to surround the string with double quotes. For example, "BLD_APP". Double quotes are not needed around integers or variables.
- The following are workflow specific variables available when defining your criteria:
  - Workflow Task – wfTask
  - Workflow Status – wfStatus
- The JavaScript syntax to calculate the above fee is:
  - Math.ceil(Math.pow(({Total Sq Ft} * .01),2));

Exercise 6 – WorkflowTaskUpdateAfter – Issuance Automation

Objective:

- Schedule an Inspection for X amount of days in the future.

Exercise:

- For the event Workflow Task Update After, perform the following:
  - When the task "Permit Issuance" is set to the status of "Issued":
    - Schedule a "Final Inspection" for 1 day in the future.

Functions:

- The following function(s) are required to complete this exercise:
  - scheduleInspection(inspDesc,daysAhead)
    - inspDesc (string) – inspection description
    - daysAhead (integer) - Number of days in the future to schedule the inspection for.

Exercise Notes:

- Standard Choice: WTUA:Building/User XX/NA/NA
- The following are workflow specific variables available when defining your criteria:
  - Workflow Task – wfTask
  - Workflow Status – wfStatus
- Final Inspection details:
- inspDesc = "Final Inspection"

Exercise 7 – InspectionResultSubmitAfter – Synchronizing Results with the Workflow – Creating a Child

Objective:

- Upon submittal of inspection results synchronize with the workflow and create Violation Cap if needed.

Exercise:

- For the event Inspection Result Submit After, perform the following:
  - When the inspection "Final Inspection" result is "Denied - Violation":
    - Update the workflow task "Inspection" with a status of "Denied", with a workflow comment that matches the result comment.
    - Schedule another "Final Inspection" for 15 days in the future.
    - Create a child cap of type "Building/Violation/NA/NA" with an application name = "InspectionViolation: " + Permit #. (ie. Inspection Violation: BLD08-00293).
    - Update ASI field "Violation Details" on violation cap with Inspection Result Comments from permit inspection, "Final Inspection".

Functions:

- The following function(s) are required to complete this exercise:
  - updateTask(wfTask, wfStatus, wfComment, wfNote)
    - wfTask (string) – workflow task name
    - wfStatus (string) – status to update
    - wfComment (string) – comment to add
    - wfNote (string) – note to add to workflow task
  - scheduleInspection(inspDesc,daysAhead)
    - inspDesc (string) – inspection description
    - daysAhead (integer) - Number of days in the future to schedule the inspection for.
  - createChild(grp, typ, stype, cat, appName)
    - grp (string) – App Group. Top classification of the application
    - typ (string) – App Type. Second classification of the application
    - stype (string) – App SubType: 3rd Classification of the application
    - cat (string) – App Category: 4th Classification of the application
    - appName (string) - Application name
  - editAppSpecific(itemName,itemValue,capId)
    - itemName (string) – App Spec Info field to edit
    - itemValue (string) - Value that the app spec info field itemName should be changed to.
    - capId (capIDModel) - CapID object for application whose app spec info field *itemName* is to be changed to *itemValue*.

Exercise Notes:

- Standard Choice: IRSA:Building/User XX/NA/NA
- The following are inspection specific variables available when defining your criteria or actions:
  - Inspection Type – inspType
  - Inspection Result – inspResult
  - Inspection Result Comment - inspComment
- When updating the workflow via EMSE must use the correct workflow function based on the desired action: updateTask (No Change), closeTask (Go to Next Task), branchTask (Go to Branch Task), loopTask(Go to Loop Task)
- Script controls can be continued onto a second line by starting the next line with: ^
- The capId can be captured when calling the createChild() function for use with other function calls to update the new child cap:
  - childId = createChild("Building","Violation","NA","NA","Application Name");
  - editAppSpecific("Field Name","Field Value",childId);

Bonus Exercises

BONUS 1

- For the event Workflow Task Update After, perform the following:
  - Upon the task "Application Acceptance" being update to the status of "Accepted":
    - Assess and invoice the "Application Fee".
- For the event Inspection Result Submit After, perform the following:
  - When the inspection "Final Inspection" result is "Approved":
    - Close the workflow task "Inspection" with a status of "Approved", with a workflow comment that matches the result comment.
    - Close the workflow task "Closure" with a status of "Closed"
  - When the inspection "Final Inspection" result is "Denied":
    - Update the workflow task "Inspection" with a status of "Denied", with a workflow comment that matches the result comment.
    - Schedule another "Final Inspection" for 15 days in the future.

Functions:

- The following function(s) are required to complete this exercise:
  - addFee(fcode,fsched,fperiod,fqty,finvoice)
    - fcode (string) – Fee code to be added
    - fsched (string) – Fee schedule of the fee to be added
    - fperiod (string) – Fee period to be used
    - fqty (integer) – Quantity to be added
    - finvoice (string) – Flag for invoicing ("Y" or "N")
  - closeTask(wfTask, wfStatus, wfComment, wfNote) & updateTask(wfTask, wfStatus, wfComment, wfNote)
    - wfTask (string) – workflow task name
    - wfStatus (string) – status to update
    - wfComment (string) – comment to add
    - wfNote (string) – note to add to workflow task
  - scheduleInspection(inspDesc,daysAhead)
    - inspDesc (string) – inspection description
    - daysAhead (integer) - Number of days in the future to schedule the inspection for.

Exercise Notes:

- Application Fee Details
  - fcode = BLD_SCRIPT, fsched = BLD, fperiod = FINAL, fqty = 1, finvoice = Y

BONUS 2

Objective:

- Retrieve the email address from the applicant and display to end user.

Exercise:

- For the event Workflow Task Update After, perform the following:
  - When the task "Application Acceptance" is set to the status of "Received":
    - Retrieve the email address of the applicant and display to the user.

Functions:

- The following function(s) are required to complete this exercise:
  - getContactArray([capId])
    - capId (capIDModel) – optional (if null will pull from current cap)
  - comment(cstr)
    - cstr (string) – Message to display to user

Exercise Notes:

- Standard Choice: WTUA:Building/User XX/NA/NA
- getcontactArray()Retrieves field values and custom attribute values for all contacts and returns them as an array of associative arrays. Each element in the outer array contains an associative array of values for one contact. Each element in each inner associative array is a different field. All custom attributes are also added to the associative array, where the element name is the attribute name (in upper-case). Note that the attribute name may not be the same as the attribute label.

Contact Field	Element Name
First Name	firstName
Last Name	lastName
Business Name	businessName
Phone 1	phone1
Phone 2	phone2
Email Address	email
Contact Type	contactType
Contact Relationship	relation
Sequence Number	contactSeqNumber

- You reference a value like this: contactArray[0]["email"]
- We will need to use a for loop to examine the array looking for an applicant, if found display the email address.

BONUS 3

Objective:

- Apply a condition to the current application

Exercise:

- For the event Workflow Task Update After, perform the following:
  - When the task "Application Acceptance" is set to the status of "Received":
    - Apply a condition with a severity of notice, with a
      - Description of, "Training condition"
      - Comment of, "I am automatically applying this condition just because".

Functions:

- The following function(s) are required to complete this exercise:
  - addAppCondition(cType,cStatus,cDesc,cComment,cImpact)
    - cType (string) – Type of condition
    - cStatus (string) – Status
    - cDesc (string) – Description of the condition (free text)
    - cComment (string) – Condition comment (free text)
    - cImpact (string) - must be "Lock", "Hold", "Notice", "Required", or ""

Exercise Notes:

- Standard Choice: WTUA:Building/User XX/NA/NA
- The type of condition will be: "Training"
- The status of the condition will be: "Applied"
- The impact of the condition will be: "Notice"

Solutions

Exercise 1:

Standard Choice: ASA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  true ^ totalSqFt = 0;
03  typeof(ROOMDIMENSIONS) == "object" ^ for(x in ROOMDIMENSIONS)
    branch("ASA:Building/User 0/NA/NA:LOOP");
04  totalSqFt > 0 ^ editAppSpecific("Total Sq Ft",totalSqFt);
05  proximity("flagstaff","Water_Lines",100) ^ showMessage = true; comment("Current
    parcel is within 100 feet of city water lines");
06  true ^ editAppSpecific("Zoning
    Classification",getGISInfo("flagstaff","Parcels","ZONING"));
07  true ^ pinArray = new Array(); pinArray =
    getGISBufferInfo("flagstaff","Parcels",500,"PIN");
08  pinArray.length > 0 ^ showMessage = true; comment("The following PINs are located
    within 500 feet:"); for(z in pinArray) comment(pinArray[z]["PIN"]); ^ showMessage =
    true; comment("There are no PINs within 500 feet.");
```

Standard Choice: ASA:Building/User XX/NA/NA:LOOP

```
01  true ^ totalSqFt += ROOMDIMENSIONS[x]["Length"] *
    ROOMDIMENSIONS[x]["Width"]
```

Exercise 2:

Standard Choice: ASA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  true ^ totalSqFt = 0;
03  typeof(ROOMDIMENSIONS) == "object" ^ for(x in ROOMDIMENSIONS)
    branch("ASA:Building/User 0/NA/NA:LOOP");
04  totalSqFt > 0 ^ editAppSpecific("Total Sq Ft",totalSqFt);
05  proximity("flagstaff","Water_Lines",100) ^ showMessage = true; comment("Current
    parcel is within 100 feet of city water lines");
06  true ^ editAppSpecific("Zoning
    Classification",getGISInfo("flagstaff","Parcels","ZONING"));
07  true ^ pinArray = new Array(); pinArray =
    getGISBufferInfo("flagstaff","Parcels",500,"PIN");
08  pinArray.length > 0 ^ showMessage = true; comment("The following PINs are located
    within 500 feet:"); for(z in pinArray) comment(pinArray[z]["PIN"]); ^ showMessage =
    true; comment("There are no PINs within 500 feet.");
```

Exercise 3:

Standard Choice: ASA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  true ^ totalSqFt = 0;
03  typeof(ROOMDIMENSIONS) == "object" ^ for(x in ROOMDIMENSIONS)
    branch("ASA:Building/User 0/NA/NA:LOOP");
04  totalSqFt > 0 ^ editAppSpecific("Total Sq Ft",totalSqFt);
05  proximity("flagstaff","Water_Lines",100) ^ showMessage = true; comment("Current
    parcel is within 100 feet of city water lines");
06  true ^ editAppSpecific("Zoning
    Classification",getGISInfo("flagstaff","Parcels","ZONING"));
07  true ^ pinArray = new Array(); pinArray =
    getGISBufferInfo("flagstaff","Parcels",500,"PIN");
08  pinArray.length > 0 ^ showMessage = true; comment("The following PINs are located
    within 500 feet:"); for(z in pinArray) comment(pinArray[z]["PIN"]); ^ showMessage =
    true; comment("There are no PINs within 500 feet.");
```

Exercise 4:

Standard Choice: ASA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  true ^ totalSqFt = 0;
03  typeof(ROOMDIMENSIONS) == "object" ^ for(x in ROOMDIMENSIONS)
    branch("ASA:Building/User 0/NA/NA:LOOP");
04  totalSqFt > 0 ^ editAppSpecific("Total Sq Ft",totalSqFt);
05  proximity("flagstaff","Water_Lines",100) ^ showMessage = true; comment("Current
    parcel is within 100 feet of city water lines");
06  true ^ editAppSpecific("Zoning
    Classification",getGISInfo("flagstaff","Parcels","ZONING"));
07  true ^ pinArray = new Array(); pinArray =
    getGISBufferInfo("flagstaff","Parcels",500,"PIN");
08  pinArray.length > 0 ^ showMessage = true; comment("The following PINs are located
    within 500 feet:"); for(z in pinArray) comment(pinArray[z]["PIN"]); ^ showMessage =
    true; comment("There are no PINs within 500 feet.");
```

Exercise 5:

Standard Choice: WTUA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  wfTask == "Application Acceptance" && wfStatus == "Accepted" ^
    addFee("BLD_SCRIPT","BLD","FINAL",1,"Y");
03  wfTask == "Initial Review" && wfStatus == "Detailed Review Required" ^ revFee =
    0;
04  ^ revFee = Math.ceil(Math.pow(({Total Sq Ft} * .01),2));
05  ^ addFee("BLD_REV","BLD","FINAL",revFee,"Y")
06  wfTask == "Permit Issuance" && wfStatus == "Issued" ^ scheduleInspection("Final
    Inspection",1);
07  wfTask = "Application Acceptance" && wfStatus == "Received" ^ var emailAddress;
    ca = new Array(); ca = getContactArray(); for (y in ca) if(ca[y]["contactType"] ==
    "Applicant") emailAddress = ca[y]["email"];
08  wfTask = "Application Acceptance" && wfStatus == "Received" && emailAddress !=
    null ^ showMessage = true; comment("Applicant's email address is: " + emailAddress);
    ^ showMessage = true; comment("No applicant email address found");
09  wfTask = "Application Acceptance" && wfStatus == "Received" ^
    addAppCondition("Training","Applied","Training Condition","I am automatically
    applying this condition just because","Notice");
```

Exercise 6:

Standard Choice: WTUA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  wfTask == "Application Acceptance" && wfStatus == "Accepted" ^
    addFee("BLD_SCRIPT","BLD","FINAL",1,"Y");
03  wfTask == "Initial Review" && wfStatus == "Detailed Review Required" ^ revFee =
    0;
04  ^ revFee = Math.ceil(Math.pow(({Total Sq Ft} * .01),2));
05  ^ addFee("BLD_REV","BLD","FINAL",revFee,"Y")
06  wfTask == "Permit Issuance" && wfStatus == "Issued" ^ scheduleInspection("Final
    Inspection",1);
07  wfTask = "Application Acceptance" && wfStatus == "Received" ^ var emailAddress;
    ca = new Array(); ca = getContactArray(); for (y in ca) if(ca[y]["contactType"] ==
    "Applicant") emailAddress = ca[y]["email"];
08  wfTask = "Application Acceptance" && wfStatus == "Received" && emailAddress !=
    null ^ showMessage = true; comment("Applicant's email address is: " + emailAddress);
    ^ showMessage = true; comment("No applicant email address found");
09  wfTask = "Application Acceptance" && wfStatus == "Received" ^
    addAppCondition("Training","Applied","Training Condition","I am automatically
    applying this condition just because","Notice");
```

Exercise 7:

Standard Choice: IRSA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  inspType == "Final Inspection" && inspResult == "Denied" ^
    updateTask("Inspection","Denied",inspComment,"updated via script");
03  ^ scheduleInspection("Final Inspection",15);
04  inspType == "Final Inspection" && inspResult == "Approved" ^
    closeTask("Inspection","Approved",inspComment,"updated via script");
05  ^ closeTask("Closure","Closed","updated via script","");
06  inspType == "Final Inspection" && inspResult == "Denied - Violation" ^
    updateTask("Inspection","Denied",inspComment,"updated via script");
07  ^ scheduleInspection("Final Inspection",15);
08  ^ childId = createChild("Building","Violation","NA","NA","Inspection Violation: " +
    capIDString);
09  ^ editAppSpecific("Violation Details",inspComment,childId);
```

Bonus Exercise 1:

Standard Choice: IRSA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  inspType == "Final Inspection" && inspResult == "Denied" ^
    updateTask("Inspection","Denied",inspComment,"updated via script");
03  ^ scheduleInspection("Final Inspection",15);
04  inspType == "Final Inspection" && inspResult == "Approved" ^
    closeTask("Inspection","Approved",inspComment,"updated via script");
05  ^ closeTask("Closure","Closed","updated via script","");
06  inspType == "Final Inspection" && inspResult == "Denied - Violation" ^
    updateTask("Inspection","Denied",inspComment,"updated via script");
07  ^ scheduleInspection("Final Inspection",15);
08  ^ childId = createChild("Building","Violation","NA","NA","Inspection Violation: " +
    capIDString);
09  ^ editAppSpecific("Violation Details",inspComment,childId);
```

Standard Choice: WTUA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  wfTask == "Application Acceptance" && wfStatus == "Accepted" ^
    addFee("BLD_SCRIPT","BLD","FINAL",1,"Y");
03  wfTask == "Initial Review" && wfStatus == "Detailed Review Required" ^ revFee =
    0;
04  ^ revFee = Math.ceil(Math.pow(({Total Sq Ft} * .01),2));
05  ^ addFee("BLD_REV","BLD","FINAL",revFee,"Y")
06  wfTask == "Permit Issuance" && wfStatus == "Issued" ^ scheduleInspection("Final
    Inspection",1);
```

```
07  wfTask = "Application Acceptance" && wfStatus == "Received" ^ var emailAddress;
    ca = new Array(); ca = getContactArray(); for (y in ca) if(ca[y]["contactType"] ==
    "Applicant") emailAddress = ca[y]["email"];
08  wfTask = "Application Acceptance" && wfStatus == "Received" && emailAddress !=
    null ^ showMessage = true; comment("Applicant's email address is: " + emailAddress);
    ^ showMessage = true; comment("No applicant email address found");
09  wfTask = "Application Acceptance" && wfStatus == "Received" ^
    addAppCondition("Training","Applied","Training Condition","I am automatically
    applying this condition just because","Notice");
```

Bonus Exercise 2:

Standard Choice: WTUA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  wfTask == "Application Acceptance" && wfStatus == "Accepted" ^
    addFee("BLD_SCRIPT","BLD","FINAL",1,"Y");
03  wfTask == "Initial Review" && wfStatus == "Detailed Review Required" ^ revFee =
    0;
04  ^ revFee = Math.ceil(Math.pow(({Total Sq Ft} * .01),2));
05  ^ addFee("BLD_REV","BLD","FINAL",revFee,"Y")
06  wfTask == "Permit Issuance" && wfStatus == "Issued" ^ scheduleInspection("Final
    Inspection",1);
07  wfTask = "Application Acceptance" && wfStatus == "Received" ^ var emailAddress;
    ca = new Array(); ca = getContactArray(); for (y in ca) if(ca[y]["contactType"] ==
    "Applicant") emailAddress = ca[y]["email"];
08  wfTask = "Application Acceptance" && wfStatus == "Received" && emailAddress !=
    null ^ showMessage = true; comment("Applicant's email address is: " + emailAddress);
    ^ showMessage = true; comment("No applicant email address found");
09  wfTask = "Application Acceptance" && wfStatus == "Received" ^
    addAppCondition("Training","Applied","Training Condition","I am automatically
    applying this condition just because","Notice");
```

Bonus Exercise 3:

Standard Choice: WTUA:Building/User XX/NA/NA

```
01  true ^ showMessage = false; showDebug = false;
02  wfTask == "Application Acceptance" && wfStatus == "Accepted" ^
    addFee("BLD_SCRIPT","BLD","FINAL",1,"Y");
03  wfTask == "Initial Review" && wfStatus == "Detailed Review Required" ^ revFee =
    0;
04  ^ revFee = Math.ceil(Math.pow(({Total Sq Ft} * .01),2));
05  ^ addFee("BLD_REV","BLD","FINAL",revFee,"Y")
06  wfTask == "Permit Issuance" && wfStatus == "Issued" ^ scheduleInspection("Final
    Inspection",1);
```

```
07    wfTask = "Application Acceptance" && wfStatus == "Received" ^ var emailAddress;
      ca = new Array(); ca = getContactArray(); for (y in ca) if(ca[y]["contactType"] ==
      "Applicant") emailAddress = ca[y]["email"];
08    wfTask = "Application Acceptance" && wfStatus == "Received" && emailAddress !=
      null ^ showMessage = true; comment("Applicant's email address is: " + emailAddress);
      ^ showMessage = true; comment("No applicant email address found");
09    wfTask = "Application Acceptance" && wfStatus == "Received" ^
      addAppCondition("Training","Applied","Training Condition","I am automatically
      applying this condition just because","Notice");
```

- Solutions can also be found in Accela Automation under the following standard choices:
  - ASA:Building/User 0/NA/NA
  - ASA:Building/User 0/NA/NA:LOOP
  - WTUA:Building/User 0/NA/NA
  - IRSA:Building/User 0/NA/NA