**General Configuration for Licenses, Amendments, and Renewals**

**Renewal Configuration**

# Contents

[Contents 3](#_Toc324506841)

[Revision History 4](#_Toc324506842)

[Licenses Overview 5](#_Toc324506843)

[License Application Overview 8](#_Toc324506844)

[Scripts for License Application 9](#_Toc324506845)

[Contact Overview 10](#_Toc324506846)

[Scripts for Contacts 11](#_Toc324506847)

[Optional: Contact Public User Overview 13](#_Toc324506848)

[Optional: Contact Public User Configuration 14](#_Toc324506849)

[Optional: Changing a Contact Type (Role) After Issuance 15](#_Toc324506850)

[Renewal Overview 16](#_Toc324506851)

[Renewal Configuration 18](#_Toc324506852)

[FIDS, Server Constants, and Standard Choices 18](#_Toc324506853)

[Renewal Group Configuration 18](#_Toc324506854)

[V360 Configuration 19](#_Toc324506855)

[Batch Scripts 19](#_Toc324506856)

[Configuring Instant Renewal Records (ACA Issuance and Counter) 22](#_Toc324506857)

[Configuring Non-Instant Renewals (Workflow or some other approval) 22](#_Toc324506858)

[Configuring Renewal Email Notifications 23](#_Toc324506859)

[Loading the Renewal Scripts 23](#_Toc324506860)

[Calling the Renewal Scripts from Master Scripts using Variable Branching 24](#_Toc324506861)

[Amendment Overview 26](#_Toc324506862)

[Amendment Script Examples 27](#_Toc324506863)

[ACA Amendment Configuration 28](#_Toc324506864)

[Fee Estimates 30](#_Toc324506865)

[License Lookup 32](#_Toc324506866)

[Frequently Asked Questions 37](#_Toc324506867)

# Revision History

| ***Author*** | ***Date*** | ***Revision*** | ***Description*** |
| --- | --- | --- | --- |
| John Schomp | 06/29/2009 | 1.0 | Initial document regarding renewals |
| John Schomp | 09/16/2009 | 2.0 | Added license, contact, and public user configuration |
| John Schomp | 12/6/2011 | 2.1 | Added code to close out the renewal record on instant renewals |
| John Schomp | 12/12/2011 | 3.0 | Updated to new standards of best practices, using standard scripts, and new product functionality. |
| John Schomp | 1/5/2012 | 3.01 | Fixed issue with amendment script example |
| John Schomp | 5/9/2012 | 4.0 | Various updates. Updated to use master scripts 2.0. Added changing contact role, fee estimates, license lookup, batch scripts, etc. |

# Licenses Overview

License Records have the following unique characteristics:

* They have an expiration date and status, and are renewable via ACA or AA
* They can have child Caps that are actions upon the license, such as a renewal, amendment, case (case management), cancellation, verification request, etc.

Based on these requirements, there are some configuration challenges that have to be addressed:

* The Licensee must have the capability to manage their licenses in ACA. This includes the ability to pay fees, upload documents, amend, and renew their licenses. In order for the ACA user to see their licenses, their public user will need to be linked to either the record, an associated reference contact, or an associated reference licensed professional.
* Must provide the ability to handle renewals in the back office.
* ACA users may need the ability to search for licenses on the website. This would be a a services usually called a “business directory” or “professional directory”. License verification functionality is based on searching Licensed Professionals that are linked (identical IDs) to License Records.

The below Figure 1 represents an overall high level view of the entities involved in licensing configuration:

**Figure 1** – Licensing Configuration Entity Relationship



1. License Application Record (eg. Licenses/Dentistry/Dental Technician/Application)

* Is the starting point of licensing process, represents the initial application for a license. Think of this record as the “*process of obtaining a license*” in the overall licensing lifecycle.
* It consists of required contacts, address(es), and application data (eg. ASI and ASIT) required for review and approval of license application.
* There should not be a Licensed Professional configured on the smart choice group for this Record because the LP will be created when this record has gone through approval workflow
* Application and license fees
* Once the application Record is approved, the License Record (#2 below) will be created as a parent related record. The License Application Record will serve as historical snapshot of the license as it appeared at the time of initial issuance and will not be acted upon by the applicant or agency again. In addition, the Licensed Professional Record (#4 below) will be created via EMSE. It will have the same ID as the License Record.

1. License Record (eg. Licenses/Dentistry/Dental Technician/License)

* Will always represent the current version of the license.
* Renewal (#6 below) and Amendment (#7) records will be the mechanism to update data on the License Record
* Relevant fields are the record status, renewal status, and renewal expiration date.

1. Contacts

* Record contacts are very useful, as they can also be used to link the License Record to various public users.
  1. Some examples could be multiple partners on a business license, or a hospital nursing manager that needs access to all of his/her nurses’ licenses.
* For this contact functionality to work properly, reference contacts and contact validation must be used, this enables the linking of License Records to the appropriate Reference Contacts and Public Users.
  1. See the sections [Contact Overview](#_Contact_Overview), [Scripts for Contacts](#_Scripts_for_Contacts), [Optional: Contact Public User Overview](#_Optional:_Contact_Public) and [Optional: Contact Public User Configuration](#_Optional:_Contact_Public_1) for more details.

1. Reference Licensed Professionals (LP)

* When the license is issued, a reference LP will be created that will contain information about the license.
* The usage of the reference LP is used to support several functions.
  1. Used for license searching and verification in ACA.
* The reference LP should not be added to the Licensed Professional tab on the License Record because the relationship is already implied by the License Record AltID having the same ID as the reference LP.

1. Public User Account

* The Public User Account will be linked to any Reference Contacts to provide a link to the License Record.

1. Renewal Records

* Once a License Record has been marked as “About to Expire” it is now eligible for renewal.
  + ACA users will see a “Renew Application” link
  + AA users will see a “Renewal” button on the Renewals tab
* When the renewal is started a Renewal Record is created and if required applicable data can be copied from the License Record through control of EMSE.
* Upon approval of the Renewal Record the License Record is updated with any changes in data from Renewal Record as well as a new Expiration Status and Date.
* A new Renewal Record is created for each license renewal.
* See sections [Renewal Overview](#_Renewal_Overview) and [Renewal Configuration](#_Renewal_Configuration) for more details.

1. Amendment Records

* During the lifecycle of a license there may be times when data on the License Record must be updated for reasons such as requested modifications to the license, cancelation of the license, the creation of an official copy, or the opening of a license investigation or case.
* To support these requirements the configuration of the necessary Amendment Records is required. The Amendment Record will contain the required details of the request and upon approval will act appropriately upon the license record.

1. Other Record Types

* There may be other record types that may interact with the license during its lifecycle such as enforcement and adjudication records.

Configuration

# License Application Overview

The primary element of license configuration is to automate the creation of the License Record and the License Professional record as a result of an approved license application. This is accomplished by using the “createParent” and “createRefLicProf” functions. Since these records may need to be populated in very different ways, it is likely that this function will need to be customized for your specific implementation. The configuration section should be considered a generic solution.

It is recommended that the License Record and the LP record be created when the license is issued, this typically occurs when the approval workflow task/status is completed.

The following actions will be implemented via scripting:

* Create the License Record (createParent())
* Copy relevant data to the License Record (eg. application status, application name, application specific information)
* Determine the License Type value for the new LP record. This can be determined from the Record type, or perhaps a cross reference based on an ASI value.
* Determine the source for the LP demographic data. Typically this is copied from the Applicant contact on the Record.
* Determine the LP License Number. It is recommended that it is the same as the Alt ID of the issuance Record.
* Determine if the LP already exists. If the LP exists, we will update the information instead of create a new license.
* Create/Update the LP
* (optional) Find the public user by contact email address and link to the Record

# Scripts for License Application

**Table 1:** **WTUA:Licenses/Business/\*/Application**

|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **02** | wfTask == "Issuance" && wfStatus == "Issued" ^ branch("LIC Issue Business License"); // run this line after syncing contacts |

**Table 2:** **LIC Issue Business License**

|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | true ^ newLic = null; newLicId = null ; newLicIdString = null ; newLicenseType = "Business"; monthsToInitialExpire = 12; |
| **02** | true ^ newLicId = createParent(appTypeArray[0], appTypeArray[1], appTypeArray[2], "License",null); // create the license record |
| **03** | newLicId ^ newLicIdString = newLicId.getCustomID(); updateAppStatus("Active","Originally Issued",newLicId); editAppName({Doing Business As (DBA) Name},newLicId); |
| **04** | ^ copyAppSpecific(newLicId); |
| **05** | newLicIdString ^ createRefLicProf(newLicIdString,newLicenseType,"Applicant"); |
| **06** | ^ newLic = getRefLicenseProf(newLicIdString); |
| **07** | newLic ^ newLic.setAuditStatus("A");// enable the LP |
| **08** | ^ newLic.setBusinessName({Doing Business As (DBA) Name}); // set LP name to ASI Field |
| **09** | ^ aa.licenseScript.editRefLicenseProf(newLic); |
| **10** | true ^ tmpNewDate = dateAddMonths(null, monthsToInitialExpire); |
| **11** | newLic ^ thisLic = new licenseObject(newLicIdString,newLicId) ; thisLic.setExpiration(dateAdd(tmpNewDate,0)) ; thisLic.setStatus("Active"); |

# Contact Overview

In general, it is recommended that a licensing implementation adopt a “contact-centric” approach. In practice this means that reference contacts should be used as much as possible, and the system should be configured to keep the reference contact library as pristine as possible by storing each person only once.

This is different than most historic permitting implementations, where a “snapshot” approach is generally accepted when storing information about people on a permit.

Accela Automation does not yet have native functionality to maintain contact centricity, so this must be implemented using scripts.

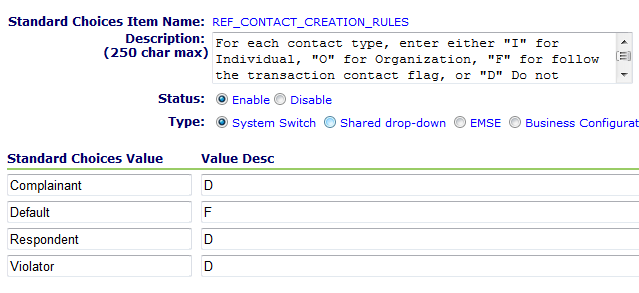
# Scripts for Contacts

Generally, scripts are used to ensure that most, if not all, contacts entered on a Record are linked to corresponding reference contacts that may or may not exist. To accomplish this, a script is run at some point during the license process to “sync” up the contacts.

If a contact is hand-entered (not selected from reference), the script will search for reference contacts based on the contact’s email address (or some other custom criteria, see below). If a reference contact is found, it will link it to the Record contact. If a reference record is not found, it will create one based on the Record contact data, then link it back.

To implement:

* If necessary, customize and save the function “comparePeopleStandard” to the master script custom include file. This will override the standard function. This function is referenced by *createRefContactsFromCapContactsAndLink* and used to determine if a reference contact exists. By default, the comparison is made by email address but this function can be edited to compare using additional fields and criteria. **The search criteria must be required fields for Record contacts and must be populated and unique in all reference contacts. Do not implement if this is not the case.**
* Configure the “REF\_CONTACT\_CREATION\_RULES” standard choice. In the master script version 2.0, the *createRefContactsFromCapContactsAndLink* function will use this standard choice to determine if the reference contact will be created, as well as the contact type that the reference contact will be created with. The “Default” in this standard choice determines the default action of all contact types. Other types can be configured separately. Each contact type can be set to “I” (create ref as individual), “O” (create ref as organization), “F” (follow the indiv/org flag on the cap contact), “D” (Do not create a ref contact) , or “U” (create ref using the transactional contact type”) determine the contact type of the reference contact that will be created. Suggested values are:



* Modify and add the following standard choice script controls, based on the Records required and event chosen:

**T****able 3:** **WTUA:Licenses/Business/\*/Application**

|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | wfTask == "Issuance" && wfStatus == "Issued" ^ branch("LIC Establish Links to Reference Contacts"); |

**Table 4:** **LIC Establish Links to Reference Contacts**

|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | true ^ iArr = new Array(); // attributes to ignore |
| **02** | true ^ contactTypeArray = new Array(); // ignored since we are using REF\_CONTACT\_CREATION\_RULES |
| **03** | true ^ createRefContactsFromCapContactsAndLink(capId,contactTypeArray,iArr,false,false,comparePeopleStandard); |

# Optional: Contact Public User Overview

This script provides the ability to automatically link or create an ACA public user to an application that is entered in Accela Automation. The script searches for a public user based on email address. If one is not found, a new public user record is created based on contact data. The public user is then linked as the creator of the Record, which allows access in ACA.

1. During application intake, the ASI Field “Create Public User Account?” field is checked “Yes”.
2. After the application is submitted, the script checks that there is an “Applicant” contact with an email address. If it exists, it will check to see if there is already a public user account using that email address.
3. If a public user account with the email address exists, the newly created application will be attached to their current online account.
4. If there is no public user account with that email address, the following will occur:
   1. The public user account will be created.
   2. The email address will be the login ID
   3. The account will be enabled for the current agency
   4. The account will be activated for the current agency
   5. The password will be reset
   6. The activate email will be sent
   7. The password reset will be sent
   8. The newly created application will be attached to the new online account

# Optional: Contact Public User Configuration

* Create a yes/no field in the ASI data for desired Records that will trigger the creation of the account. This field should not be made available for ACA. An example field name would be: “Create ACA Account?”
* Add the following standard choice script controls, based on the event that will trigger the public user creation. The function assumes that the “Applicant” contact contains the information needed to create the public user. If another contact type is desired, pass the name of the desired contact type to the function as an optional parameter:

**Table 5:** **Create Public User (ApplicationSubmitAfter or WorkflowTaskUpdateAfter)**

|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | true ^ onlineUser = null; |
| **02** | {Create ACA Account?} == “Yes” ^ onlineUser = createPublicUserFromContact(); |
| **03** | onlineUser ^ attachResult = aa.cap.updateCreatedAccessBy4ACA(capId,"PUBLICUSER" + onlineUser.getUserSeqNum(),"Y","Y") |

* For the source contact, email address should be a required field. The function will not create a public user if there is no email address.

# Optional: Changing a Contact Type (Role) After Issuance

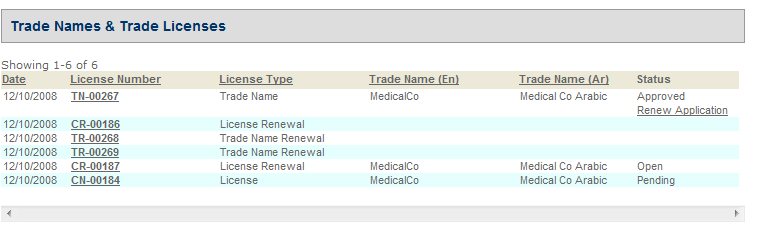
In the case of professional licenses it is often desirable to change the role of the Applicant to License Holder once the license is issued. This can be accomplished via script once the License Record is created from the License Application Record. Here is an example:

**Table 6:** **Changing Contact Type (WorkflowTaskUpdateAfter)**

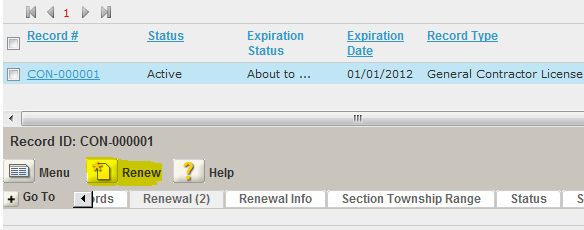
|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | newLic ^ conToChange = null; cons = aa.people.getCapContactByCapID(newLicId).getOutput() ; for (thisCon in cons) if (cons[thisCon].getCapContactModel().getPeople().getContactType() == "Applicant") conToChange = cons[thisCon].getCapContactModel(); ^ conToChange = null; |
| **02** | conToChange ^ p = conToChange.getPeople(); p.setContactType("License Holder"); conToChange.setPeople(p); aa.people.editCapContact(conToChange); |

# Renewal Overview

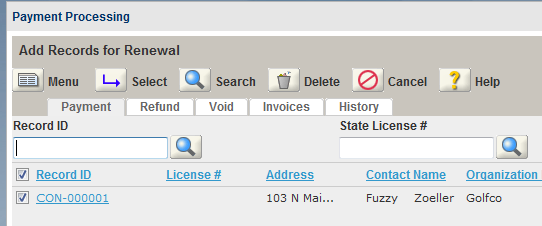
1. A License is created and approved. Once approved, it is given an expiration status of “Active” and an expiration date of sometime in the future.
2. The future arrives, and a batch script looks for licenses that are about to expire. The batch script will set the license expiration status to “About to Expire” and notify the licensee.
3. The licensee goes to ACA, mails in their renewal, or goes to the counter.
4. In ACA the user will see a “Renew Application” link. If they choose this link, it will start a new application for the renewal. It will also remember the parent license Record ID and pass that along to the AA scripts



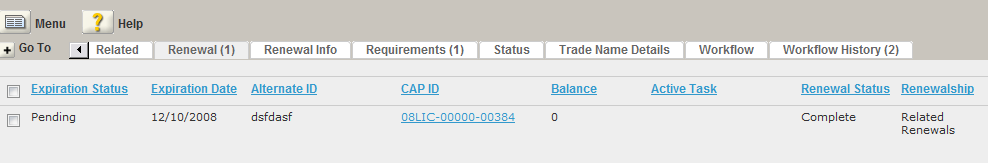
1. In V360, the license can be renewed from the “Renewals” tab. A link appears when the license can be renewed.



1. Also in V360, the License can be renewed from the “Payment Processing” portlet. This allows for rapidly renewing and paying for a number of licenses at a time.



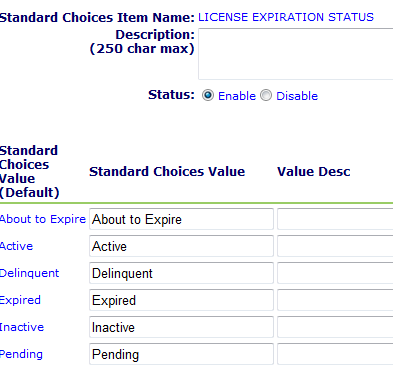
1. After the Renewal Record is completed (either paid or via workflow) the Renewal Record will be closed, and the License expiration date and status will be updated.
2. Renewal history can be reviewed in the Renewals tab.



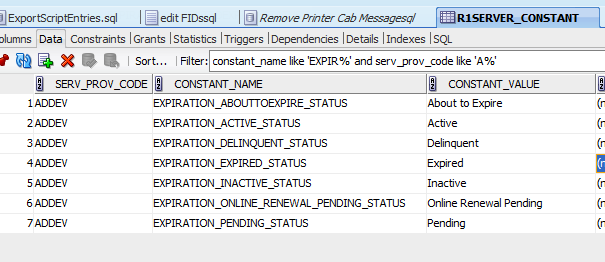
# Renewal Configuration

## FIDS, Server Constants, and Standard Choices

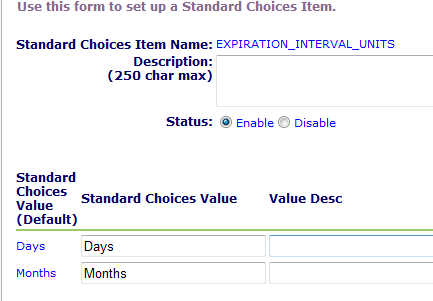
* Set up FIDS for Licensing
* Set up the Standard Choices



* Add your Server Constants

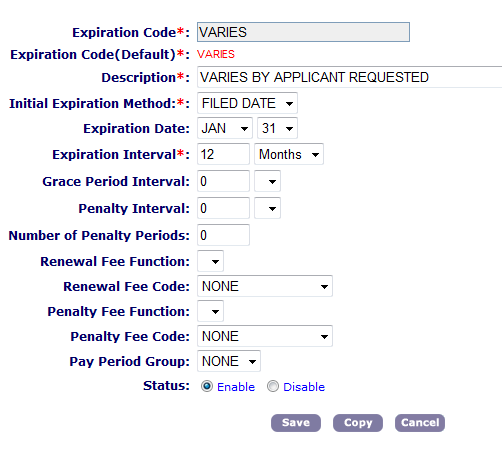


* Set up this standard choice



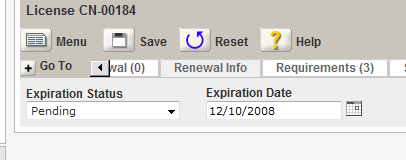
## Renewal Group Configuration

Set up a Renewal Expiration Group, and attach to the License Record. I called it “Varies” since the user can select the length of the license. So we will set the expiration date via script.



## V360 Configuration

Since we are only using the expiration date and the status, I removed unnecessary elements from the Expiration portlet.



## Batch Scripts

The batch jobs should be configured based on the business rules defined during project analysis. Example of business rules would be the timing of expirations, grace periods, late fees, inspections to be scheduled, license deactivation, etc.

While batch expiration scripts are usually customized to meet the needs of these business rules, the standard expiration script should be used as a starting point. This section describes the standard expiration script.

Configuring a Batch Job in Accela

1. Go to the Batch page on V360. If you do not have it set up, add it to your console.
2. Define a batch job. Depending on the difference in business rules, different license types may need to have their own job and script definitions. The jobs are named based on the new expiration status of the license. For example “CL About to Expire” is the job that will change licenses from status “Active” to status “About to Expire”.
3. Define the parameters for the batch job. The standard script has the following parameters:

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **Scheduled Status** | Set this parameter to “Scheduled” for the job to run |
| **Start Date** | Job will not run until on or after this date |
| **Start Time** | The time at which the job will run. |
| **Interval** | Interval at which the job will run. Typically this will be daily. |
| **LookAheadDays** | The batch job will search for records that have a certain expiration date. This date is located in the “Renewal Info” tab.    The LookAheadDays parameter is added to the current date to calculate a start date for the search. For example:  Today’s Date: 5/1/2012  LookAheadDays: 7  The start date for the search will be 5/8/2012 |
| **DaySpan** | This parameter is added to the search start date in order to come up with a range of days for the search. For example:  Today’s Date: 5/1/2012  LookAheadDays: 7  DaySpan: 5  The job will search for expiration records starting 5/8/2012 to 5/13/2012 |
| **GracePeriodDays** | The number of days that are added to the record expiration date. Usually zero. |
| **AppGroup** | The first level of CAP type to search for. Can use an asterisk “\*” for a wildcard. Usually “Licenses” |
| **AppTypeType** | The second level of CAP type to search for. Can use an asterisk “\*” for a wildcard. |
| **AppSubType** | The third level of CAP type to search for. Can use an asterisk “\*” for a wildcard. For example: “Issue” |
| **AppCategory** | The fourth level of CAP type to search for. Can use an asterisk “\*” for a wildcard. |
| **ExpirationStatus** | The current expiration status to search for, located in the “Renewal Info” tab. |
| **NewExpirationStatus** | The expiration status on the CAP will be updated to this value. |
| **NewApplicationStatus** | (optional) the CAP status will be updated to this value. |
| **SkipAppStatus** | A comma-separated list of CAP Status values. If a CAP has a status in this list, it will be ignored. For example “Void,In Progress,Open”. |
| **SetPrefix** | If the batch job finds one or more CAPs to update, it will create a Set of all CAPs in the Set portlet. The set name will be a date stamp, with the prefix supplied here. For example, “AD\_EXPIRE” will result in a Set as shown below: |
| **EmailAddress** | (optional) Any messages from the script (debug, info, etc) will be emailed to the address here |
| **sendEmailToContactTypes** | A comma-separated list of contact types that will receive email notifications. For example “Applicant,License Holder”. |
| **Email Template** | The Notification Template that will be used to generate the outgoing emails. |
| **ShowDebug** | Set to “Yes” if debug out should be recorded. |
| **DeactivateLicense** | If set to “Yes”, will set the License Professional record to “Disabled”. This will prevent anyone from selecting this license for any application. See example: |
| **inspSched** | The Inspection Type that should be scheduled when this CAP is processed. |
|  |  |

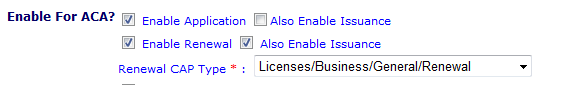
For License Applications, the standard script can be used for all phases of the license expiration. The following table demonstrates the various parameters that can be used to enforce license expiration. In this example the Applicant will be notified 30 days before their License expires. They will receive a second notice the day after the License expires. Two days after expiration, the record status will be set to “Expired” and the License will be deactivated.

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | TN About to Expire | TN Expire | TN Inactive |
| appCategory | New | New | New |
| appGroup | Licenses | Licenses | Licenses |
| appSubtype | Issue | Issue | Issue |
| appTypeType | Trade Name | Trade Name | Trade Name |
| daySpan | 0 | 0 | 0 |
| deactivateLicense | N | N | Y |
| emailTemplate | TN About to Expire Message | TN Expire Message | TN Inactive Message |
| expirationStatus | Active | About to Expire | Expired |
| gracePeriodDays | 0 | 0 | 0 |
| lookAheadDays | 30 | -1 | -2 |
| newApplicationStatus |  |  | Expired |
| newExpirationStatus | About to Expire | Expired | Inactive |
| sendEmailToContactTypes | Applicant | Applicant | Applicant |
| setPrefix | TN\_ABOUT | TN\_EXPIRE | TN\_INAC |
| skipAppStatus | Open,In Progress,Void,Licensed | Open,In Progress,Void,Licensed | Open,In Progress,Void,Licensed |
|  |  |  |  |
|  |  |  |  |

## Configuring **Instant Renewal Records (ACA Issuance and Counter)**

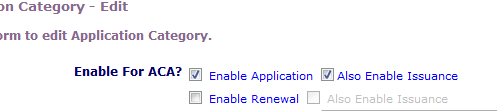
If the renewals only require fees and there is no review or workflow involved, then set up your Renewals using this method. The License Record should have a renewal Record type, as shown below:

**License Record set up for instant renewals:**



The Renewal Record should be set up as shown below:

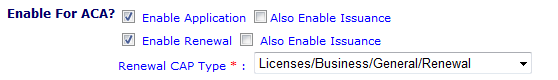
**Renewal Record set up for instant renewals:**



## **Configuring Non-Instant Renewals (Workflow or some other approval)**

This type of renewal can be submitted via ACA, but require processing by the agency. Typically this is accomplished via workflow.

**License Record set up for non-instant renewals**



**Renewal Record set up for non-instant renewals:**



The renewal record will also need to have a workflow step that triggers the completion of the renewal. By default, the standard scripts are looking for:

Task: “L-License Status” and Status “L-About To Expire” that denotes that the renewal is denied.

Task “L-License Status” and Status “L-Active” that denotes that the renewal is approved.

If you need to have different task/status names, you will need to replace them in the WorkflowTaskUpdateAfter4Renew script.

## Configuring Renewal Email Notifications

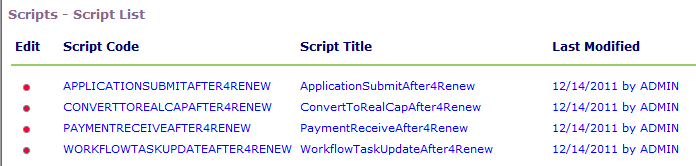
There are email notifications that are automatically sent for renewal actions. Be sure to set these up using information located in the ACA Administrators guide.

## Loading the Renewal Scripts

Load the four standard renewal scripts into the agency. These scripts are associated to Salesforce case #08ACC-06264. They are named according to the associated event, but it is recommended that they are saved in the agency as shown so they do not conflict with any master scripts that are in use.

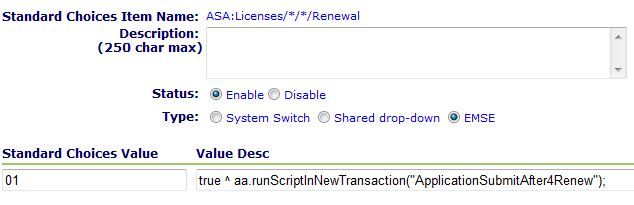
|  |  |
| --- | --- |
| ***Script Name*** | ***Save Script as*** |
| **ApplicationSubmitAfter.txt** | ApplicationSubmitAfter4Renew |
| **Convert2RealCapAfter.txt** | Convert2RealCapAfter4Renew |
| **PaymentReceiveAfter.txt** | PaymentReceiveAfter4Renew |
| **WorkflowTaskUpdateAfter.txt** | WorkflowTaskUpdateAfter4Renew |

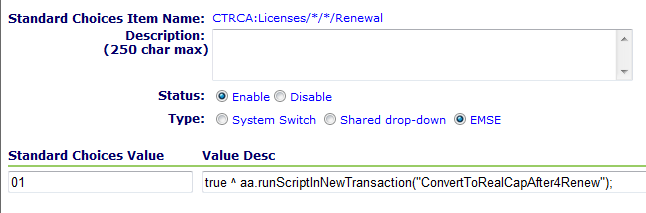
After loading, a script search for “%4Renew” should return the following:

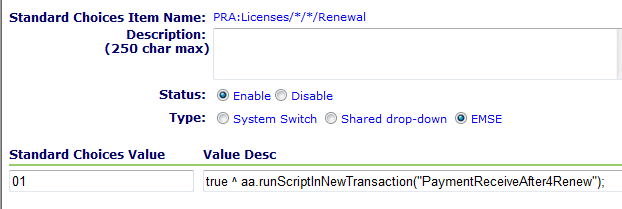


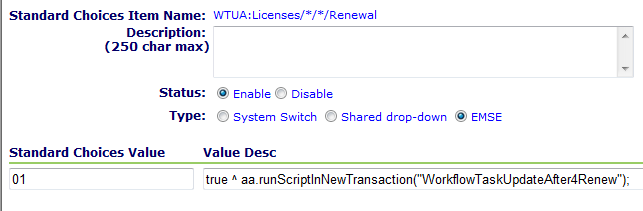
## Calling the Renewal Scripts from Master Scripts using Variable Branching

The following master script standard choices are the best practice for calling the renewal scripts. Essentially the scripts should be called whenever one of these events occurs on a renewal record.









# Amendment Overview

We’ll call any child of an issued License Record that isn’t a renewal an “Amendment”. Amendments can be created to modify the license, cancel the license, create an official copy, an investigation, or a case regarding the initial license. Typically we will use EMSE scripts to copy information from the parent license to the new amendment as required. When the amendment Record is complete, standard EMSE functions may be used to update the parent data as required.

# Amendment Script Examples

The following script will locate the parent license and copy information to the new Record. It assumes that the License record is already associated as the parent which will happen automatically from ACA. Typically amendments are not initiated from AA.

**Table 7: ASA:Licenses/Address Modification/\*/\***

|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | true ^ parentName = null ; parentCapId = getParent(); |
| **02** | parentCapId ^ parentCap = aa.cap.getCap(parentCapId).getOutput(); if (parentCap) parentName = parentCap.getSpecialText(); |
| **03** | parentName ^ editAppName(parentName);^ copyContacts(TNCap, capId); |

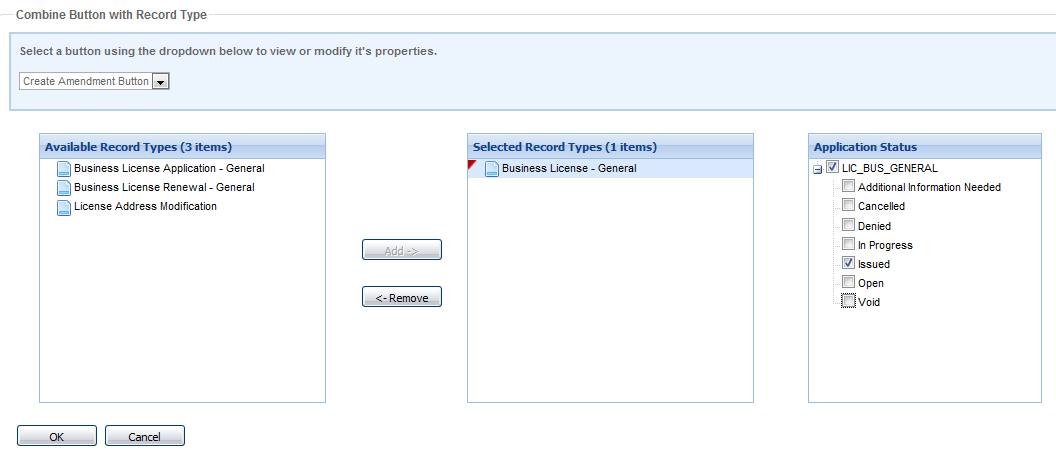
The following script will use the amendment record to update the “Applicant” contact address on the parent license record. It assumes that the License record is already associated as the parent which will happen automatically from ACA. Typically amendments are not initiated from AA.

**Table 8:** **Amendment Script Example**

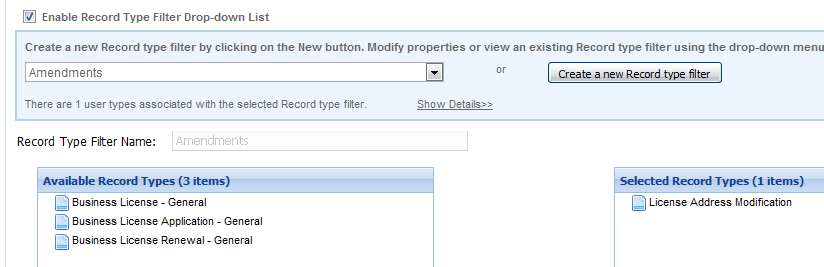
|  |  |
| --- | --- |
| **Line** | **Script Action** |
| **01** | true ^ contactType = "Applicant"; ca = null; p = null ; contactToEdit = null; contactTypeArray = new Array(contactType); iArr = new Array(); |
| **02** | parentCapId ^ capContacts = aa.people.getCapContactByCapID(parentCapId).getOutput(); |
| **03** | capContacts ^ for (thisContact in capContacts) if (contactType.equals(capContacts[thisContact].getCapContactModel().getPeople().getContactType())) contactToEdit = capContacts[thisContact]; |
| **04** | contactToEdit ^ ca = contactToEdit.getCapContactModel().getPeople().getCompactAddress(); |
| **05** | ca ^ if ({Address Line 1} != "") ca.setAddressLine1({Address Line 1}); else ca.setAddressLine1(""); |
| **06** | ca ^ if ({Address Line 2} != "") ca.setAddressLine2({Address Line 2}); else ca.setAddressLine2(""); |
| **07** | ca ^ if ({City} != "") ca.setCity({City}); else ca.setCity(""); |
| **08** | ca ^ if ({State} != "") ca.setState({State}); else ca.setState(""); |
| **09** | ca ^ if ({Zip} != "") ca.setZip({Zip}); else ca.setZip(""); |
| **10** | contactToEdit ^ p = contactToEdit.getCapContactModel().getPeople(); |
| **11** | p ^ if ({Phone} != "") p.setPhone1({Phone}); else p.setPhone1(""); |
| **12** | p ^ if ({Mobile Phone} != "") p.setPhone2({Mobile Phone}); else p.setPhone2(""); |
| **13** | p ^ if ({Fax} != "") p.setFax({Fax}); else p.setFax(""); |
| **14** | contactToEdit ^ aa.people.editCapContact(contactToEdit.getCapContactModel()); |
| **15** | contactToEdit ^ createRefContactsFromCapContactsAndLink(parentCapId,contactTypeArray,iArr,false,true,comparePeopleGeneric); |
| **16** | true ^ updateAppStatus("Approved","Instant Approval and update via ACA"); |

# ACA Amendment Configuration

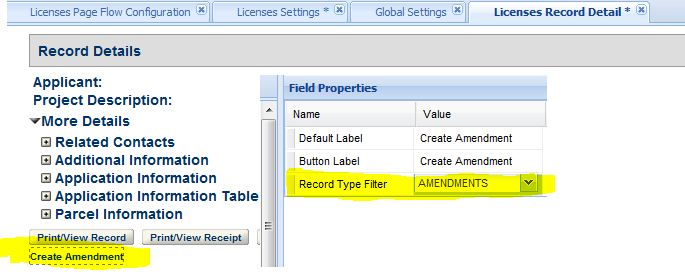
In ACA, the “Combine Button with Record Type” can be used to create amendment choices. Select “Create Amendment Button”, and link all the records for which amendments are applicable. You can also select the record statuses that apply:



Next, define which record types are to be shown as amendments by creating a record type filter, such as the one below:



Finally, associate the record type filter to the “Create Amendment” button on the record detail page:



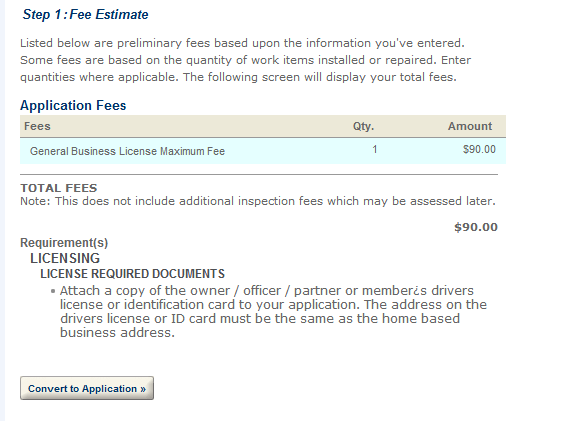
For these record types/statues, the “Amendment” link will appear in the license management page in ACA. By creating the amendment Record in this fashion, only two additional things will happen:

1. The amendment will be automatically linked to the license record in hierarchy. Use getParent() to obtain the parent record ID.
2. For page flow scripts, aa.env.getValue("CapModel").getParentCapID() will be populated with the parent Cap ID.
3. The hierarchy between the parent and child are created properly.

Since we have the parent Cap ID in both ACA and AA scripts, we can populate data from the parent into amendment as required.

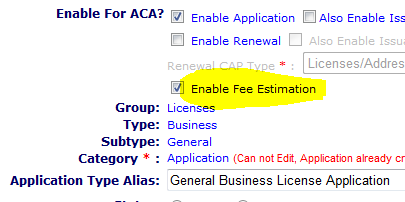
# Fee Estimates

You can offer a way for public users to get a fee estimate online. This is useful so that an applicant can budget their costs and other requirements for a license application, renewal, or amendment. For example, a fee estimate can appear as follows:

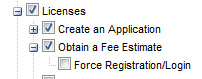


Once a fee estimate is obtained, the user can quickly convert the estimate to a real application. The following are the steps to configuring fee estimates in ACA:

1. In the Record Type definition, enable fee estimation for the desired license type(s):



1. In ACA Admin, Feature Settings, turn on the fee estimation function for the desired module(s). Check the Force Registration/Login box if you want to require the user to log in before obtaining a fee estimate.



1. In the Record Type configuration, ensure that any ASI/ASIT field that is used for fee calculation has “Required” = “Yes” and “Req for Fee Calc” = “Yes”, even if these fees are entirely scripted. This will ensure that the fee estimate page will prompt the user for these values.
2. If any of your fees or conditions are scripted, you will need to load the FeeEstimateAfter4ACA script. There is a master script available for this event. The event is enabled in the Global Settings of the ACA Admin page. You can use the master script to execute any standard choice entries that create fees or attach conditions. Since this event only occurs on temporary records, do not execute any other script actions.

# License Lookup

In order to utilize the licensee lookup functionality in ACA it is required that the license record types be synchronized with their reference licensed professional. The licensee lookup synchronization EMSE script is used to update the reference licensed professional record whenever a business or contractor license record is create, updated, renewed, or amended.

The EMSE:LicProfLookup script leverages the licenseProjObject introduced in the Master Script 2.0 release. It is recommended that the object definition detailed in the Master Script 2.0 document be thoroughly reviewed before making any modifications to the standard licensee lookup script.

There are two sections that need to be manually configured for each implementation. The first is the default license professional state. It is recommend this be set in the “EMSE:GlobalFlags” standard choices (see example below). The second section that needs to be configured is the “EMSE:LicProfLookup:getLicenseType”, this section controls the association between the Record Type and the reference licensed professional type (by default this is based upon the best practice templates).

When the EMSE:LicProfLookup branch can be called from the License, Renewal, Application, or Amendment record to sync the license professional information. Once invoked the EMSE:LicProfLookup will perform the following:

1. Locate the License Record Type
2. If the license record is found the script will branch to EMSE:LicProfLookup:getLicenseType and sets the LICENSETYPE variable that will control the reference license professional type that is create and/or updated. By default this is based upon the best practices templates (BPT) and will need to be updated if the license record type structure differs from the BPT.
3. Attempt to find the associated reference license using the License Record ID and derived License Type
   1. If the license is not found then the script will create a new reference licensed professional using the Record ID and derived License Type.
4. Update the reference license professional with the following mappings. These are located in the EMSE:LicProfLookup:UpdateLP:BaseFields standard choice.
   1. License Issued Date = License Record created date
   2. License Expiration Date = License Record expiration date
   3. Licensing Board = ASI “Business Type” value
   4. Contact Information from Primary License Record Contact
   5. Address information from Primary License Record Address
   6. Business Name = ASI “Doing Business as (DBA) Name”
5. Update the reference license professional with the Record Status. This logic is located in the EMSE:LicProfLookup:UpdateLP:ApplicationStatus standard choice. In the example below Business Name 2 is used to store the Record Status.

To ensure that the records are always kept in sync is it is recommended that the “EMSE:LicProfLookup” branch be invoked whenever the following actions occur:

1. License Application is approved
   1. WTUA:Licenses/\*/\*/Application
      1. wfTask == "License Issuance" && wfStatus == "Issued" ^ branch("EMSE:LicProfLookup");
2. License Amendment is approved
   1. WTUA:Licenses/Address Modification/Premise/\*
      1. parentCapId && wfTask.equals("Amendment Approval") && wfStatus.equals("Approved") ^ branch("EMSE:LicProfLookup");
3. License Renewal record is completed
   1. WTUA:Licenses/\*/\*/Renewal
      1. true ^ branch("EMSE:LicProfLookup");
   2. PRA:Licenses/\*/\*/Renewal
      1. true ^ branch("EMSE:LicProfLookup");
4. License record is manually updated (in the event this is allowed outside of an amendment)
   1. Application Status
   2. Application Specific Information
   3. Address added/deleted/updated
   4. Contact added/deleted/updated
   5. Expiration status updated
5. Batch process updates the License record

The following script example is built utilizing the Master Script 2.0 standard functionality and is designed to be called from anywhere in the license record hierarchy from any event.

------------------------------------------------

EMSE:GlobalFlags

------------------------------------------------

Licensee Lookup Default State true ^ LICENSESTATE = "CA";

------------------------------------------------

EMSE:LicProfLookup

------------------------------------------------

10 true ^ logDebug("Using LICENSESTATE = " + LICENSESTATE + " from EMSE:GlobalFlags");

20 true ^ LICENSETYPE = ""; //License Type to be populated

30 true ^ licCapId = null; isNewLic = false; licIDString = null; licObj = null; licCap = null;

40 true ^ branch("EMSE:LicProfLookup:getLicenses"); //Get License CAP

50 licCapId !=null ^ branch("EMSE:LicProfLookup:getLicenseType");

60 true ^ licObj = licenseProfObject(licIDString,LICENSETYPE); //Get LicArray

70 !licObj.valid && lookup("LICENSED PROFESSIONAL TYPE",LICENSETYPE) != null ^ branch("EMSE:LicProfLookup:CreateLP"); licObj = licenseProfObject(licIDString,LICENSETYPE );

80 licObj.valid ^ branch("EMSE:LicProfLookup:UpdateLP"); ^ logDebug("LP Not found to update");

------------------------------------------------

EMSE:LicProfLookup:getLicenses

------------------------------------------------

01 true ^ var searchCap = capId; var tmpId = capId; var prjArr = null;

02 appMatch("\*/\*/\*/License") ^ var childArr = getChildren("\*/\*/\*/Application"); if(childArr != null) searchCap = childArr[0];

03 true ^ capId = tmpId; var vRelationType = "R"; if(appMatch("\*/\*/\*/Renewal")) vRelationType="Renewal";

04 true ^ var prjArrRes = aa.cap.getProjectByChildCapID(searchCap,vRelationType,null); if(prjArrRes.getSuccess()) prjArr = prjArrRes.getOutput();

05 prjArr != null ^ for(prj in prjArr) if(appMatch("\*/\*/\*/License",prjArr[prj].getProjectID())) licCapId = prjArr[prj].getProjectID();

06 licCapId == null && appMatch("\*/\*/\*/License") ^ licCapId = capId; //In the event license has no application

07 licCapId != null ^ licCapId = aa.cap.getCapID(licCapId.getID1(),licCapId.getID2(),licCapId.getID3()).getOutput(); logDebug("Got Lic Cap " + licCapId.getCustomID());

------------------------------------------------

EMSE:LicProfLookup:getLicenseType

------------------------------------------------

10 licCapId !=null ^ licIDString = licCapId.getCustomID();

20 licCapId !=null ^ licCap = aa.cap.getCap(licCapId).getOutput(); licCapType = licCap.getCapType().toString(); licCapTypeArr = licCapType.split("/"); licCapStatus = licCap.getCapStatus();

30 licCapId !=null ^ if(licCapTypeArr[1] == "Business") LICENSETYPE = "Business"; else LICENSETYPE = getAppSpecific("License Type",licCapId)+"";

------------------------------------------------

EMSE:LicProfLookup:CreateLP

------------------------------------------------

01 true ^ var vNewLic = aa.licenseScript.createLicenseScriptModel();

02 true ^ vNewLic.setAgencyCode(aa.getServiceProviderCode()); vNewLic.setAuditDate(sysDate); vNewLic.setAuditID(currentUserID); vNewLic.setAuditStatus("A");vNewLic.setLicenseType(LICENSETYPE); vNewLic.setLicState(LICENSESTATE); vNewLic.setStateLicense(licIDString);

03 true ^ aa.licenseScript.createRefLicenseProf(vNewLic);

04 true ^ var tmpLicObj = licenseProfObject(licIDString,LICENSETYPE);

05 tmpLicObj.valid ^ isNewLic = true;

------------------------------------------------

EMSE:LicProfLookup:UpdateLP

------------------------------------------------

01 true ^ branch("EMSE:LicProfLookup:UpdateLP:BaseFields");

02 true ^ branch("EMSE:LicProfLookup:UpdateLP:ApplicationStatus");

03 licObj.updateRecord() ^ logDebug("LP Updated Successfully"); ^ logDebug("LP Update Failed");

------------------------------------------------

EMSE:LicProfLookup:UpdateLP:BaseFields

------------------------------------------------

01 true ^ licObj.refLicModel.setState(LICENSESTATE);

02 true ^ licObj.refLicModel.setLicenseBoard(LICENSETYPE);

03 true ^ licObj.refLicModel.setLicenseIssueDate(licCap.getFileDate());

04 true ^ var expObj = null; var expDt = null; var expObjRes = aa.expiration.getLicensesByCapID(licCapId); if(expObjRes.getSuccess()) var expObj = expObjRes.getOutput();

05 expObj != null ^ expDt = aa.date.parseDate(expObj.getExpDateString());

06 expDt != null ^ licObj.refLicModel.setLicenseExpirationDate(expDt); //Expiration Date

07 licCapTypeArr[1] == "Business" ^ licObj.refLicModel.setLicenseBoard(getAppSpecific("Business Type",licCapId)); ^ licObj.refLicModel.setLicenseBoard(LICENSETYPE);

08 licObj.updateFromRecordContactByType(licCapId,"",true,false); ^ logDebug("LP Updated from Primary Contact"); ^ logDebug("LP Failed to Update from Primary Contact trying License Holder"); if(licObj.updateFromRecordContactByType(licCapId,"License Holder",true,false)) logDebug("Updated from License Holder"); else logDebug("Couldn't Update Contact Info");

09 licObj.updateFromAddress(licCapId) ^ logDebug("LP Address Updated from License Address"); ^ logDebug("LP Address Failed to update from License Address");

10 getAppSpecific("Doing Business As (DBA) Name",licCapId) ^ licObj.refLicModel.setBusinessName(getAppSpecific("Doing Business As (DBA) Name",licCapId) );

------------------------------------------------

EMSE:LicProfLookup:UpdateLP:ApplicationStatus

------------------------------------------------

01 true ^ licObj.refLicModel.setBusinessName2(licCapStatus); logDebug("Lic Cap Status: " + licCapStatus);

# Frequently Asked Questions

         Is it possible to skip certain fields when creating reference license or contact records?

Yes, this is completely configurable since we are creating the reference record from scratch.

        Do we have to use “About to Expire” status?

Yes there is AA logic tied to this.

         Do you recommend that we always set the expiration date in the script?  Or is the example to meet specific business requirement?

Just an example, but yes the expiration date should be set via EMSE.

         Are there still problems with batch scripts timing out?  are we going to have a problem renewing 11,000 nurse licenses at the same time?  should we find some way to create smaller batches?

There are documented workarounds for timeout issues – see services intranet documents.

        Will the ASI, etc data be copied down from the parent license Record to the renewal Record?

Yes. By default the standard scripts copy information from the parent to renewal.

        Where does the ‘Create Amendment’ button appear in ACA?

On the Record summary as well as the manage licenses screen.