# Table of Contents

**CHANGE HISTORY** .......................................................................................................................... 5  
**IMPLEMENTATION SUMMARY** ..................................................................................................... 6  
  Service Oriented Architecture User Interface ............................................................................... 7  
  Web Services .................................................................................................................................. 8  
  User Account Manager .................................................................................................................. 8  
  Constituent Transaction Manager ................................................................................................. 9  
  Admissions API .............................................................................................................................. 12  
**POST INSTALLATION SETUPS** ....................................................................................................... 15  
  Verifying Web Service Security Settings ....................................................................................... 15  
  Setting Up User Accounts ............................................................................................................... 17  
  Performing a Security Analysis ...................................................................................................... 19  
  Publishing Web Services ............................................................................................................... 20  
  Refreshing the HCM Registry Cache .............................................................................................. 20  
  Setting Up Integration Broker ....................................................................................................... 21  
  Verifying Campus Solutions SOA Framework Setup Entries ....................................................... 22  
  Setting Up Logging ....................................................................................................................... 24  
  Verifying Constituent Transaction Manager Setup Entries .......................................................... 25  
  Configuring AAWS ......................................................................................................................... 27  
**EXTENDING AAWS** ...................................................................................................................... 35  
**USER INTERFACE CONSIDERATIONS** ....................................................................................... 45  
  Discussion of the User Interface Flow ............................................................................................ 46  
**TROUBLESHOOTING** ................................................................................................................... 55  
**APPENDIX: LOADING APPLICATIONS IN BULK** ....................................................................... 56  
  Setting Up File Parser Definitions .................................................................................................. 56  
  Setting up Constituent Transaction ............................................................................................... 60  
  Running the File Parser Process .................................................................................................... 62  
  Reviewing Staged Data ................................................................................................................... 62  
  Running the Search Match Post Process ......................................................................................... 64  
**APPENDIX: DEVELOPERS REFERENCE** ....................................................................................... 66
# CHANGE HISTORY

<table>
<thead>
<tr>
<th>Release</th>
<th>Change Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Pack 3 (July, 2010)</td>
<td>First release of this guide for <em>Feature Pack 3: Admission Applications Web Services (AAWS)</em>.</td>
</tr>
<tr>
<td>Bundle 21 (April, 2011)</td>
<td>Added Error Handling section (page 57).</td>
</tr>
<tr>
<td>Additional Features July 2011 (Bundle 22)</td>
<td>- Made updates to account for the LOV and the Entity Registry enhancements included inside the July 2011 bundle.</td>
</tr>
<tr>
<td></td>
<td>- Updated <em>Appendix: Loading Applications in Bulk</em> to fix the issue Bug ID BugID 11647407 - AAWS: AAWS DEVELOPER'S GUIDE AND AAWS USER'S GUIDE NEED CORRECTIONS.</td>
</tr>
<tr>
<td></td>
<td>- Replaced the information given inside the <em>Appendix: Developers Reference</em> section by a reference to the section “Setting Up Entity Registry” added as part of the July 2011 bundle in the PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook.</td>
</tr>
</tbody>
</table>

This updated guide is available on My Oracle Support as part of the documentation delivered for the releases indicated in the table.
IMPLEMENTATION SUMMARY

This document is aimed for a technical audience who wants to install and configure Admission Applications Web Service (AAWS). AAWS interacts with the following Campus Solutions components:

- User Account Manager
- Constituent Transaction Manager
- Admissions API
- Student Financials Hosted Payment

This document lists the application packages and classes for these components. If you want to customize AAWS, you may need to modify some classes and packages. For instance, if you want to integrate a non PeopleSoft user management system with AAWS, you may need to modify the Campus Identity Manager Application package.

This document also describes how to use the File Parser feature to load applications from a text file into Campus Solutions.

AAWS adheres to open web service standards currently supported in the latest release of Enterprise PeopleTools 8.48+. Therefore, any modern day web service compliant front end (for example, an online application user interface) or third party application can use AAWS.

**Note:** The AAWS User's Guide describes the various AAWS operations. We recommend that you read the AAWS User's Guide before performing the configuration steps listed in this document.

To use AAWS, complete the following steps:

1. **Bundle Installation (Mandatory)**
   
   AAWS is delivered with Campus Solutions Bundle 18 (update ID 782229). Bundle 18 was posted to My Oracle Support in July 2010.

   Minor change to the Counter Setup page was made with Campus Solutions Bundle 19 (update ID 782230). Bundle 19 was posted to My Oracle Support in Oct 2010.

   AAWS Fee integration is delivered with Campus Solutions Bundle 20 (update ID 812865). Bundle 20 was posted to My Oracle Support in Jan 2011.

   Error handling changes were made with Campus Solutions Bundle 21 (update ID 812867). Bundle 21 was posted to My Oracle Support in April 2011.

   List of Values enhancements made with Campus Solutions Bundle 22 (update 812869).
2. Configuration (Mandatory)

Perform the configuration steps to make AAWS accessible to the user interface.

3. Implement Extensions (Optional)

Design and create the country or institution specific extensions that AAWS or any of the underlying Campus Community components can use.

The following diagram shows an overview of the AAWS architecture:

**Figure 1: AAWS Architecture**

**Service Oriented Architecture User Interface**

Due to AAWS's open standards Service Oriented Architecture (SOA), you can use any technology to develop your user interface, as long as the technology supports web service and SOAP.

The SOA user interface (SOA UI) could be an online application that interacts with Campus Solutions through AAWS. Before developing your online application user interface, we recommend that you use a SOAP service tester to become familiar with how the web services processes the admission application data and user information. One way to use a service tester could be by first passing the input parameters to a service operation and then viewing the service operation output.
Web Services

The Admission Applications Web Service contains the following web service operations:

- SAD_ADMISSIONS – Admissions Web Service
  - SAD_CREATEAPPL: Create Application
  - SAD_GETAPPL: Find Application
  - SAD_GETAPPLS: Find Applications
  - SAD_SAVEAPPL: Save Application (for later modification or submission)
  - SAD_SUBMITAPPL: Submit Application
  - SAD_GETATTACH: Get Attachments

- SCC_USERREG - User Account Web Service
  - SCC_USERREG_AUTHENTICATE: Authenticate User
  - SCC_USERREG_CREATEACCT: Create User Account

- SCC_LOV – List of Values Web Service
  - SCC_GET_LOV: Get List Of Values

- SSF_EPAYMENT_TRANSACTION – Hosted Payment Web Service
  - SSF_INITIATE_EPAYMENT: Initiate a Payment
  - SSF_COMPLETE_EPAYMENT: Complete a Payment

User Account Manager

The user registration and authentication web service operations interact with User Account Manager to register and authenticate the end user. PeopleTools User Profile, by default, stores the user account information.

The User Account Manager component contains a number of sub-modules including the Campus Identity Manager and the User Manager. The following diagrams show the core application packages and classes which support the User Account Manager:
Constituent Transaction Manager

*Constituent Transaction Manager* handles the admission application and applicant data that AAWS receives and sends to the user interface. When the user interface submits the application and applicant data, AAWS transmits this data to the Constituent Transaction Manager. The Constituent Transaction Manager stores this data in Campus Solutions staging tables.

Constituent Transaction Manager is composed of a number of application classes and PeopleTools components which support the staging and loading of constituent (applicant) and related transaction (application) data into staging tables.

The following diagrams show the application packages and classes in the Constituent Transaction Manager component:
Figure 4: Constituent Transaction Manager (1 of 3)
Figure 5: Constituent Transaction Manager (2 of 3)
Extensions

The Entity Registry feature enables academic institutions to create their own constituent data extensions (that is, you can add entirely new constituent child entities). Refer to the "Extending AAWS" section in this guide for examples.

Admissions API

The Admission Applications Programming Interface (API) moves the application data from the staging tables to the core Recruiting and Admissions tables.

The API supports the creation, modification, and submission of online admission applications. The API is an Object Oriented (OO) based application class hierarchy that represents the admission applications. The class hierarchy contains an Application Manager API that performs key functions with application data. The API also supports the calculation of Application Fees via integration to the Application Fee hosted payment processing.
The following diagram shows the application packages and classes in the Admission Applications Manager component:
Figure 7: Admission Applications Manager

Extensions

The Entity Registry feature enables academic institutions to create their own application data extensions (that is, you can add entirely new application child entities). Refer to the "Extending AAWS" section in this guide for examples.
This chapter describes the setups that you must perform after installing Bundle 18. The chapter includes some setup steps that are listed only for verification purposes and may require little or no setup.

Verifying Web Service Security Settings

The SAD_ADMISSIONS, SCC_USERREG, SCC_LOV, SSF_EPAYMENT_TRANSACTION and SCC_SM_SERVICE service operations are delivered with **FULL ACCESS** to the **HCSPSERVICE** permission list.

**Note:** SCC_SM_SERVICE supports all aspects of the External Search Match feature and was delivered in Feature Pack 1 as part of the Constituent Web Services (CWS).

To verify whether **HCSPSERVICE** permission list is assigned to a service operation with **FULL ACCESS**:

1. Access the General page (**PeopleTools, Integration Broker, Integration Setup, Service Operations, General**).
Figure 8: Example of a General page

2. Click the Service Operation Security link. The Web Service Access page appears as shown in this example:

Web Service Access

Service: SAD_ADMISSIONS
Operation: SAD_CREATEAPPL

<table>
<thead>
<tr>
<th>Permission</th>
<th>Custom</th>
<th>First</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission List</td>
<td></td>
<td>Access</td>
<td></td>
</tr>
<tr>
<td>HCSERVICE</td>
<td></td>
<td>Full Access</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9: Example of a Web Service Access page
Setting Up User Accounts

In PeopleTools 8.48+, any anonymous inbound request originates from the ANONYMOUS node. For example, when an applicant requests for a user ID, the user registration service operation associates the request with the ANONYMOUS node.

To enable anonymous access to Campus Solutions web services in a secure fashion, the PeopleSoft system, by default, associates all anonymous requests with a default account. You will need to set up a default account that has minimal system access. This default account is the SCC_GUEST User Profile.

To create the SCC_GUEST account, use the User Profiles component (PeopleTools, Security, User Profiles, User Profiles).

This example shows how the General page of the User Profiles component is set up for SCC_GUEST:

![Figure 10: Example of a General page (SCC_GUEST)](image)

This example shows the ID page for SCC_GUEST:

![Figure 11: Example of an ID page (SCC_GUEST)](image)
This example shows the Roles page for SCC_GUEST:

Figure 12: Example of a Roles page (SCC_GUEST)

After successfully processing the user registration request, the user registration service operation creates a new account for the user. In other words, the service operation creates a PeopleTools user profile for the user. The service operation uses the template account SCC_SS_TEMPLATE to create this new account.

You will need to create the SCC_SS_TEMPLATE account. **Note:** Assign this account any common defaults that your institution wants new users to automatically receive when registering.

To create the SCC_SS_TEMPLATE account, use the User Profiles component (PeopleTools, Security, User Profiles).

This example shows how the General page of the User Profiles component is set up for SCC_SS_TEMPLATE:

Figure 13: Example of a General page (SCC_SS_TEMPLATE)

This example shows the ID page for SCC_SS_TEMPLATE:
Performing a Security Analysis

The example screenshots in the previous section related to SCC_SS_TEMPLATE role assignment and configuration are for use only in a testing or demo environment. Before you migrate AAWS and the other services to a production environment, we recommend that you complete a thorough analysis of your institution's security requirements.

Also, we recommend that you allocate the SCC_SS_TEMPLATE user only a minimal amount of system access that is required to execute the web services. This access includes authorization to access AAWS, user account, search match, and list of values web services and the underlying HCM component interfaces that are required to operate these services. To define this access, any role that you create should have access to these web services, along with at least the following component interfaces (which are used by AAWS to access the HCM Person data):

- HCR_EMPLOYEE_CHECKLIST_SRV
- HCR_IDENTIFICATN_DATA_SRV
• HCR_JOB_DATA_POI_SRV
• HCR_MIL_EE_TRK_SRV
• HCR_NAMES_OTHER_SRV
• HCR_PERSONAL_DATA_SRV

Publishing Web Services

Use the Provide Web Service page to publish the following web services (PeopleTools, Integration Broker, Web services, Provide Web Service).

• SAD_ADMISSIONS
• SCC_USERREG
• SCC_LOV
• SSF_EPAYMENT_TRANSACTION

**Optional: If you want the system to perform a search for potential duplicate records within an external system when a user submits an online application, ensure that the CWS External Search Match is set up. Refer to the "External Search/Match" chapter in the PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook for more information. If you are not using an external data hub to store applicant records, you do not need to set up CWS External Search Match.

Refreshing the HCM Registry Cache

A new version (1.1) of the Campus Solutions AddPersonCS service has been delivered. This may require the HCM service registry to be refreshed.

To refresh the HCM service registry:

1. Access the HCM Interface Registry page (Set Up HRMS, System Administration, HCM Registry, Service Registry).

HCM Interface Registry

![HCM Interface Registry page]

Figure 16: HCM Interface Registry page
2. Click the **Refresh Cache** button.

**Setting Up Integration Broker**

**Note:** We assume that you have performed and validated all the basic Integration Broker setups. Refer to the Integration Broker PeopleBooks for information on Integration Broker gateway setup and service configuration.

In PeopleTools 8.48+, any anonymous inbound request originates from the ANONYMOUS node. The PeopleSoft system, by default, associates all anonymous requests with the **SCC_GUEST** user profile. For information on setting up the **SCC_GUEST** user profile, refer to the "Setting Up User Accounts" section.

Perform the following steps to update the ANONYMOUS Integration Broker node:

1. Access the Node Definitions page (*PeopleTools, Integration Broker, Integration Setup, Service Operations*).
2. Ensure that Node Type is set to **External**, the Default User ID is set to **SCC_GUEST**, and the Active Node and Segment Aware options are selected:

![Figure 17: Node Definitions page](image)

3. Enable WS-Security for this node by selecting an authentication token type:
Verifying Campus Solutions SOA Framework Setup Entries

During installation, the system automatically inserts configuration data into the Campus Solutions SOA Framework setup tables.

To verify whether the setup entries exist:

1. Access the Request Handlers component *(Set Up SACR, System Administration, Integrations, Request Handlers)*.
2. For all the SAD_ADMISSIONS web service operations, ensure that the fields on the Request Handlers page contain the following values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Name</td>
<td>SCC_OLA</td>
</tr>
<tr>
<td>Path</td>
<td>Handlers</td>
</tr>
<tr>
<td>Application Class ID</td>
<td>Admissions</td>
</tr>
</tbody>
</table>

3. For all the SCC_USERREG web service operations, ensure that the fields on the Request Handlers page contain the following values:
4. For all the SCC_LOV web service operations, ensure that the fields on the Request Handlers page contain the following values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Name</td>
<td>SCC_COMMON</td>
</tr>
<tr>
<td>Path</td>
<td>Handlers</td>
</tr>
<tr>
<td>Application Class ID</td>
<td>LOVHandler</td>
</tr>
</tbody>
</table>

This example shows the Request Handlers page for a service operation with the correct setup entries:

5. For all the SSF_EPAYMENT_TRANSACTION web service operations, ensure that the fields on the Request Handlers page contain the following values:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package Name</td>
<td>SSF_EPAYMENT_TRANS</td>
</tr>
<tr>
<td>Path</td>
<td>HANDLERS</td>
</tr>
<tr>
<td>Application Class ID</td>
<td>Payments</td>
</tr>
</tbody>
</table>

This example shows the Request Handlers page for a service operation with the correct setup entries:
Setting Up Logging

This is an optional setup.

Use the Logging page to enable and configure the logging feature (Set Up SACR, System Administration, Integrations, Logging).

Logging

Figure 20: Logging page
Note: You should not enable logging in a production environment. SOA framework logging is only appropriate for a testing, demo or development environment or when performing critical troubleshooting activities.

**Verifying Constituent Transaction Manager Setup Entries**

During installation, the system automatically installs several Transaction Manager Configuration entries. This includes sample Transaction Manager Configuration entries for the New User Registration (ADMISSIONS_REGISTRATION) and the Admission Applications (ADMISSIONS_APPLICATION) transactions. The sample configuration is shown in the following screen shots.

For information on adding new transactions, refer to the *Managing PeopleSoft Admission Transactions* PeopleBook chapter.

To verify whether the sample configuration exists in your system, access the Transaction Setup component (*Set Up SACR, System Administration, Utilities, Constituent Transaction Mgmt, Transaction Setup*). Compare the page settings on your system with the following examples. This shows the delivered Transaction Setup page configuration for the New User Registration transaction:

![Figure 21: Delivered Configuration for New User Registration (Transaction Setup page)](image)

This shows the delivered Transaction Setup page configuration for the Admission Applications transaction:
Note that:

- The system uses the Transaction Setup page to identify the Admission Applications transaction.
- In the delivered configuration, the system has indicated that the search match process runs in **Realtime** when the application is **submitted**.
This shows the delivered Search/Match Setup page configuration for the Admission Applications transaction:

![Delivered Configuration for Admission Applications (Search/Match Setup page)](image)

Note that you can change the search match parameters based on your requirements.

**Configuring AAWS**

This section discusses how to set up:

- Attachments URL
- AAWS Applications
- Application Extensions
- Data Update Rules
- List of Values

**Setting Up the Attachments URL**

Use the URL Maintenance page to define the location that AAWS uses to store application attachments. To navigate to the URL Maintenance page, select PeopleTools, Utilities, Administration, URLs.

In this example, attachments are stored in a FTP Server:
Setting Up AAWS Applications

The Application Configuration component is the main AAWS configuration component. To navigate to the Application Configuration component, select Set Up SACR, Product Related, Recruiting and Admissions, Application Configuration, Application Configuration.

This component has four tabs that are shown in the following screen shots. This section is a brief introduction to the typical configuration options available for AAWS. The configuration presented in the screen shots is minimal and you should adjust the configuration according to your institutions needs. Please see the AAWS User’s Guide and the Managing PeopleSoft Admission Transactions PeopleBook chapter for further configuration guidance.

Tab 1 – Appl Config (Transaction Mapping)

When a user submits an application through AAWS, the Campus Solutions system uses the Appl Config page to associate the application with a transaction. The system uses this association to run search/match and process the application data according to the Transaction Manager setup.

In this example, various academic careers are mapped on the Appl Config page:

![Figure 25: Appl Config page](image-url)
Tab 2 – Application Setup (Application Center Defaulting)

When a user submits an application through AAWS, the system uses the Application Setup page to assign an application center to the application. Each application must have an application center.

This is an example of the Application Setup page:

![Application Setup page](image)

Figure 26: Application Setup page

Tab 3 – Prospect Setup (Prospect Creation)

The system uses the Prospect Setup page to create prospect data when the user saves or submits the online application. This is an optional setup. If you do not want to create a prospect record when the application is saved or submitted, leave the page blank.

This is an example of the Prospect Setup page:

![Prospect Setup page](image)

Figure 27: Prospect Setup page

Tab 4 – School Setup (Unlisted Organizations and Home Schooling)

Use the School Setup page to enter optional processing defaults for Unlisted or Home Schooling organizations.

This is an example of the School Setup page:
Figure 28: School Setup page

The Require Manual Reconciliation option can be used as an alternative to supplying a single default. When this option is chosen and an application contains school data with an Unlisted or Home School, then the application itself is not immediately submitted but instead is delayed in the staging tables until such time as an Administrator can manually review the application and resolve the Unknown Organization via the Staging Review Components.

Setting Up Application Extensions

Use the Entity Registry page to configure AAWS extensions (Set Up SACR, System Administration, Entity, Entity Registry). Refer to the "Extending AAWS" section in this document for more information.

Setting Up Data Update Rules

This is an optional setup. Use the Data Update Rule Entry page to associate data entities with data update rules. This page enables you to configure whether or how AAWS can update data in the admissions and person tables. To navigate to the Data Update Rule Entry page, select Set Up SACR, System Administration, Utilities, Constituent Transaction Mgmt, Data Update Rule Entry.

Setting Up List of Values

List of Values configuration is optional but recommended. The degree of configuration required is entirely dependent on the requirements of your institution. The List of Values Web Service will perform out-of-the-box with no configuration for simple scenarios. To navigate to the setup page, select Set Up SACR, System Administration, Utilities, List of Values, List of Values.

Refer to the "List of Values Web Service Operations" and "Configuration Hints and Tips" sections in the AAWS User’s Guide for more information about using List of Values in the context of AAWS, and to PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook, “Setting Up List of Values” for more information on how to use the delivered functionality.
Setting Up Hosted Payment

Application Fees are optionally required during application submission as configured in the Application Center Setup page. If fees are required to be paid then a prerequisite is to set up the hosted payment adapter and merchant which will facilitate the online payment.

Firstly, create a new HTML object in Application Designer. The HTML object is used to store UI specific HTML and JavaScript which will direct the Self Service user interface during the payment integration flow. Please see the User Interface Considerations section for detailed technical information on what this HTML object may need to contain.

Next, set up a new hosted payment adapter for AAWS pointing to the same application class as the one used for PIA Self Service. This step is required because the HTML definition (created above) is attached to the Adapter itself.

Next, attach the new HTML definition to the Hosted Payment Adapter settings.

Next, attach the new HTML definition to the Hosted Payment Adapter settings.
Figure 30: Hosted Payment Adapter Settings Setup page Hosted Payment Adapter Tab

The Hosted Payment Adapter Settings Parameters tab should be set up according to your Third Party Payment Provider’s requirements.

Figure 31: Hosted Payment Adapter Settings Setup page Parameters Tab

Next, the hosted payment adapter is attached to a Payment Merchant. If you are already using the hosted payment framework for Student Financials’ Self Service or Cashiering, you will still need to set up new Payment Merchants with a different set of Third Party Merchant IDs. Your third party will require
return URLs in order to navigate the user back to the online application. These return URLs are typically tied to the Third Party Merchant ID defined in the Payment Merchant.

Note that the hosted payment adapters as delivered in Feature Pack 2 (Bundle 16) use the Get Token (91) and Authorize Only (92) transactions defined in the Electronic Payment Setup section of the Payment Merchant setup. If you plan to use these adapters as delivered, you will need to set up this section in the same manner as defined in the Electronic Payment Integration Developer’s Reference Guide. In addition, you will need to set up the corresponding Integration Broker Related objects for these transactions. Please refer to the Electronic Payment Integration Developer’s Reference Guide for more details.

You are required to configure one for each payment method you accept (credit card or electronic check).

Setup SACR>Common Definitions>Electronic Payments>Payment Merchant

![Payment Merchant Setup page](image)

**Figure 32: Payment Merchant Setup page**
Finally, Payment merchants are attached to SF Merchants which are associated with the Admissions Application Center.

Credit transactions (Credit Only – 93 as defined above) can only be generated for transactions posted to the student account. Since payments received through the online admissions application are posted to the General Ledger directly, credit transactions are not supported for these application fee payments.

Setup SACR>Common Definitions>Self Service> Student Financials> SF Merchants

![SF Merchants Setup page](image)

Figure 33: SF Merchant Setup page

Note the convenience fees are not currently supported for online application fee payment.
This chapter covers how to add new entities or extend existing entities.

**Scenario** – An academic institution has a requirement to add one or more new entities into the application process. The new entity could represent entirely new data or it could represent an extension to one of the existing delivered entity data types. This may involve either or both application and constituent data. You can use the Entity Registry feature for all these situations.

Once you register an entity, the entity and its data can flow seamlessly in and out of the Campus Solutions system (the marshalling of a database record into an XML structure occurs automatically and is driven by the PeopleTools metadata).

To create an entity:

1. Create entity database tables (staging and production versions).
2. Create the supporting entity application class. Alternatively, create a placeholder for the application class, then use the Entity Registry page to generate the class code, and finally copy the generated code into the placeholder location.
3. Use the Entity Registry page to register the entity application class.

All AAWS entities are represented by the IEntity interface `SAD_ADM_APPL:Accessors:IEntity`. The IEntity interface defines the core behavior that any participating entity needs to support in order to be managed by AAWS. For each new distinct entity that you want to create for AAWS, an application class implementing the IEntity interface needs to be created and registered.

Example:

**Step 1:** Create entity tables:

This example extension record, SCC_EXTN, contains a map of sample questions and answers that need to be captured during the application submission process.

This is the *production* table for the entity:
In this example, the new extension table is a child entity to the top level Application entity. Consequently, the compound key is EMPLID, ADM_APPL_NBR and then zero, one or more additional low order keys. Here we are using SEQNO to identify any number of Q&A entries for the individual application.

The Staging table for this entity will hold the staged entity data before it is promoted to the production table. The keys for this table are similar to the staging table except that the high level key is SCC_TEMP_ID.

<table>
<thead>
<tr>
<th>Num</th>
<th>Field Name</th>
<th>Type</th>
<th>Key</th>
<th>Ord</th>
<th>Dir</th>
<th>Cur</th>
<th>Scr</th>
<th>List</th>
<th>Syn</th>
<th>Audit</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCC_TEMP_ID</td>
<td>Nbr</td>
<td>Key</td>
<td>1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ADM_APPL_NBR</td>
<td>Char</td>
<td>Key</td>
<td>2</td>
<td>Asc</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SEQNO</td>
<td>Nbr</td>
<td>Key</td>
<td>3</td>
<td>Asc</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>HINT_QUESTION</td>
<td>Char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>QUESTION_ANSWER</td>
<td>Char</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 35: Example of Staging Table Fields

**Step 2:** Create a new entity application class:

For this example, we have created the SAD_ADM_APPL:Accessors:QuestionAnswerExt class.

**Note:** Nothing else needs to be done here, the base code will be generated by the entity registry in Step 3.

The PeopleCode for SAD_ADM_APPL:Accessors:QuestionAnswerExt is delivered with Bundle 18. The significant aspects of the new QuestionAnswerExt entity application class are highlighted here.

The class is extending SCC_COMMON:ENTITY:StagedEntity:

```plaintext
class QuestionAnswerExt extends SCC_COMMON:ENTITY:StagedEntity
```

Any new class that you create must declare all its own properties:

```plaintext
property string ADM_APPL_NBR get set;
property number SEQNO get set;  property string DESCR100 get set;
property string QUESTION_ANSWER get set;
```

Any new class that you create may choose to override the validate() method in order to validate data prior to saving:

```plaintext
method validate
  /* &p_messageLog as SAD_ADM_APPL:AdmissionsApplicationResult out */
  /* Extends/implements SAD_ADM_APPL:Accessors:IEntity.validate */
  If &entityDelete Then
    /* Do nothing */
  Else
    /* Perform any required pre-Save validation */
  End-If;
end-method;
```

Any new class that you create may choose to override the preSave() method in order to default fields before the user saves the application:
method preSave
   
   If Value(%This.ADM_APPL_NBR) = 0 Then
   %This.ADM_APPL_NBR = (%This.getParent() As SAD_ADM_APPL:Accessors:Application).ADM_APPL_NBR;
   %This.data.ADM_APPL_NBR.Value = %This.ADM_APPL_NBR;
   End-If;
   
end-method;

Any new class must declare get/set methods for all of its properties:

get DESCR100
   
   Return %This.data.DESCR100.Value;
end-get;

set DESCR100
   
   %This.data.DESCR100.Value = &NewValue;
end-set;

get QUESTION_ANSWER
   
   Return %This.data.QUESTION_ANSWER.Value;
end-get;

set QUESTION_ANSWER
   
   %This.data.QUESTION_ANSWER.Value = &NewValue;
end-set;

get ADM_APPL_NBR
   
   Return %This.data.ADM_APPL_NBR.Value;
end-get;

set ADM_APPL_NBR
   
   %This.data.ADM_APPL_NBR.Value = &NewValue;
end-set;

get SEQNO
   
   Return %This.data.SEQNO.Value;
end-get;

set SEQNO
   
   %This.data.SEQNO.Value = &NewValue;
end-set;

Notes:

• Matching keys are inherited from the parent entities at save time.
This approach ensures integrity of the contained entity data via encapsulation.

**Step 3**: Register the Entity Application class with Entity Registry:

Access the Entity Registry page (*Set Up SACR, System Administration, Entity, Entity Registry*).

When creating a new entity, it will give you this page.

**Entity Registry**

![Entity Registry screenshot](image)

Figure 36: Entity Registry page

Fill out the following fields:

- **Name (Required)** – A descriptive name of the entity.
- **Entity Type (Required)** – Specifies the type of the entity. For Extending Applicant or Constituent use *Staged Entity*.
- **Status (Required)** – Set it to active, as inactive makes the entity, and any children, effectively ignored.
Description – A description of what the entity is used for.

AppClass (Required) – This should be the empty application class created in the previous step, and it will contain the entity code.

Prod Record (Required) – The production record created in step 1.

Stage Record (Not shown unless Use Staged Record is checked in the setup of the Entity Type selected) – While this field is optional, in the context of AAWS, it should be populated with the stage record created in step 1.

Element (XML) (Required) – The element name to be used for this entity in the xml; it should not match any other entity’s element name and cannot contain spaces.

Ignore the Apply Data Update Rule or anything within the Children grid, unless you intend for this entity to have children. Adding a child will be covered in a later step.

Save the entity. This will cause the buttons to appear and to trigger the initial sync of the properties.

Save the entity. This will cause the buttons to appear and to trigger the initial sync of the properties.

**Entity Registry**

**Entity Configuration**

<table>
<thead>
<tr>
<th>Entity ID:</th>
<th>SCC_ENTITY_20190625091929</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Q&amp;A Application Extension</td>
</tr>
<tr>
<td>Status:</td>
<td>Active</td>
</tr>
<tr>
<td>Entity Type:</td>
<td>Staged Entity</td>
</tr>
<tr>
<td>Description:</td>
<td>Demonstrate how to extend the application</td>
</tr>
<tr>
<td>*AppClass:</td>
<td>SAD_ADM_APPL_Accessors.QuestionAnswerExt</td>
</tr>
<tr>
<td>Prod Record:</td>
<td>SCC_EXTN</td>
</tr>
<tr>
<td>Stage Record:</td>
<td>SCC_EXTN_STG</td>
</tr>
<tr>
<td>Element (XML):</td>
<td>SCC_EXTN</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Children**

<table>
<thead>
<tr>
<th>Order</th>
<th>Entity Name</th>
<th>Status</th>
<th>Embed</th>
<th>*Min</th>
<th>*Max</th>
<th>Element Wrapper (XML)</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parents**

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
</table>

**Figure 37: An Example Entity Registry page**

Pressing the Properties Button will show the Entity Properties page.
Entity Properties

This page syncs the entity properties to the underlying records (production and stage records), allows property customizations, and the addition of transitional properties.

Everything that needs to be done for property setup is handled during the initial sync. However any customizations or transitional properties should be defined here before generating the source code.

For more information on setting up Properties, refer to PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook, “Setting Up Entity Registry”

Pressing the Generate Code button will show the following page.

Entity Code Generation

For more information on the Entity Code Generation page, refer to PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook, “Setting Up Entity Registry”.

Once you have decided on the specific options, press Generate Code button. This will provide the code so you can cut and paste that code into Application Designer.
The entity can now be added as a child to a currently existing entity.

Open the entity you want to add your newly created entity as a child of. The grid is located at the bottom of the page as shown below:

For more information on the Children grid, refer to *PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook*, "Setting Up Entity Registry".

To be clear, at this point the WSDL will not reflect the new entity. However, if the entity is added to an incoming message, and no schema validation is occurring, the incoming message containing this data will be processed and this entity will be handled.

As an example, the Q&A Application Extension entity is registered as a child entity to the Application entity. The Q&A Application Extension entity appears in the `EXTENSION_DATA` section of the outbound XML response message. This section appears within the core `ADM_APPL_DATA` section of both inbound and outbound messages. Note that the choice of tag names (for example, `EXTENSION_DATA`) is configurable.
The following is an example of the XML message that contains the Q&A Application Extension entity data:

```
<ADM_APPL_DATA class="R">
    ...
    <ADM_APP_CAR_SEQ class="R">
        <ADM_APPL_PROG class="R">
            ...
            <ADM_APPL_PLAN class="R">
                ...
                </ADM_APPL_PLAN>
            </ADM_APPL_PROG>
        </ADM_APP_CAR_SEQ>
    </ADM_APPL_DATA>
    <EXTENSION_DATA>
        <SCC_EXTN class="R">
            <EMPLID>101</EMPLID>
            <ADM_APPL_NBR>00000000</ADM_APPL_NBR>
            <SEQNO>0</SEQNO>
        </SCC_EXTN>
    </EXTENSION_DATA>
```
Press the Generate XSD button on the root entity (constituent or application) you have modified. This will provide a schema that you can copy and paste into the messages to modify the overall message schema.

**Entity Schema**

![Image of XML schema](image)

**Figure 43: Q&A Application Extension Generated Code**

Once an extension has been added to the registry it needs to be reflected in the XML schema definition for either the Constituent or Application entity.

Copy the schema generated for the Application entity to the schema in Integration Broker message SCC_ENTITY_APPLICATION.

Open PeopleTools > Integration Broker > Integration Setup > Messages. Search for message SCC_ENTITY_APPLICATION and select tab “Schema”.

---

<QUESTION_ANSWER>42</QUESTION_ANSWER>

<SCC_EXTN>
</EXTENSION_DATA>
</ADM_APPL_DATA>
Press Edit Schema and paste in the generated schema, then press Save.

Copy the schema generated for the Constituent entity to the schema in message SCC_ENTITY_CONSTITUENT.

Open PeopleTools > Integration Broker > Integration Setup > Messages. Search for message SCC_ENTITY_CONSTITUENT and select tab “Schema”.

Press Edit Schema and paste in the generated schema, then press Save. Once the schemas have been updated and the service WSDL re-published, any newly added extensions will be recognized.
USER INTERFACE CONSIDERATIONS

It is beyond the scope of this guide to cover the best practices for user interface design and an effective online application user experience. However, this chapter presents some key considerations for designing and creating a user interface that interacts with AAWS.

In addition to user interfaces interacting with AAWS, a third party application can also be interacting with AAWS by passing in completed application data to AAWS for further processing.

We recommend academic institutions to use Oracle Application Development Framework (Oracle ADF) to develop an online application user interface. However, academic institutions can use any technology or tool that complies with the minimal technical prerequisites specified in the AAWS User's Guide. Any third party application technology deployed in your institution can be used to communicate with AAWS as long as it complies with the specified prerequisites.

In this chapter, we will examine a typical data collection flow for an online application user interface. This will reveal several recommendations on how a user interface can communicate with AAWS. The following figure depicts a typical data collection flow for an online application user interface:
Figure 46: Typical Data Collection Flow for an Online Application

Discussion of the User Interface Flow

Start: Unregistered User
This is the starting point for an online application user interface.

Essentially all applicant users start in this state. Some users may already have a username and password to access the online application (these users are not required to complete the new user registration process).

The PeopleSoft system considers any user at this point of time as anonymous. Therefore, these anonymous users receive minimal access rights to AAWS until they either log into the system or complete the new user registration process.

**New User Registration**

Any (anonymous) user who needs an account (username/password combination) in order to access the online application must complete the new user registration process.

A typical self service online application UI might perform the following actions:

- Present a “Register” hyperlink that the user can click to initiate the new user registration process.
- Present a New User Registration page to capture basic information from the user. This might typically include:
  - Name
  - Address
  - Email Address
  - Username
  - Password
  - Any constituent-related information that is of interest to the institution. This can be core constituent information or an institution extension as defined in the Entity Registry.
- Validate that required information has been entered by the user.
- Formulate a request message from the constituent information.
- Submit the request message to the SCC_USERREG_CREATEACCT service operation.
- Present the user with any error response messages from AAWS and allow the user to correct the submitted data.

**LOGON (Authentication)**

Logon is the process in which the institution's Identity Management or User Profile Management system authenticates the username and password. This authentication is not a one-time event. Re-authentication must occur on every request that the online application sends to AAWS. This is a security requirement for the web service architecture and is enforced by PeopleSoft Integration Broker as a standard behavior.

Academic institutions can use the delivered SCC_USERREG_AUTHENTICATE service operation to validate a user's credentials before the user can access the online application. Typically you want to ensure that a user has the key to the door before allowing them to proceed. This is the purpose of the SCC_USERREG_AUTHENTICATE service operation.

Once the system has authenticated the user, it is also an appropriate time for the user interface to perform any initialization or setup to prepare the user for entry into the online application. Such setup may include requesting AAWS for a bulk list of values data that the user interface can subsequently
display to the user. Performing this activity up front may result in a general performance improvement and improved user experience.

A typical self service online application UI might perform the following actions:

- Present a “Login” hyperlink that the user can click to initiate the user authentication process.
- Present a Login page where the user can enter a username and password.
- Validate that the required information has been entered.
- Formulate a request message from the login information.
- Submit the request message to the SCC_USERREG_AUTHENTICATE service operation.
- Present the user with any error response messages from AAWS and allow the user to correct the submitted data.
- Once a successful login response is received, the user interface should maintain the credentials within the program state and supply these credentials on every subsequent request to AAWS as part of the WS-Security headers (thereby enabling ongoing authentication).
- The academic institution's Identity Management or User Account Management system will have its own configured rules surrounding login retries.
- Optionally, formulate a bulk List-Of-Values request message.
- Submit the request message to the SCC_GET_LOV service operation.
- Store the List-Of-Value results in a temporary storage area for use throughout the online application functioning.

**Get Application Summary**

Once the user has been authenticated, it is likely that the user would then expect to see some form of online application launch page. It is also likely that returning users would expect to view or update any applications they have saved or submitted previously. To implement this kind of scenario, the online application user interface can request from AAWS the application summary information for the logged in user and then display this information on the launch page.

For first time users, the online application does not display saved or submitted application information. For these users, the online application can display options for creating a new application or for logging out.

A typical self service online application UI might perform the following actions:

- Formulate a request message.
- Submit the request message to the SAD_GETAPPLS service operation.
- Receive the application summary response message from AAWS.
- Based on the application status, present each application to the user in a grid style format for subsequent actions.
- Typically, applications which are in a Submitted status are not eligible for update and a user interface would need to enforce this rule depending on the institution's requirements.

**Create Application**

From the online application launch page, a user may choose to start the application process by creating a new application.

A typical self service online application UI might perform the following actions:
• Present an “Apply” hyperlink that the user can click to initiate the application process.
• Present an initial Academic Learning Area page where the user can choose a field of study. For example, on this page the user supplies the required values to the SAD_CREATEAPPL service operation. The required values that the user enters might include academic program, plan, and sub plan.
• Formulate a request message.
• Submit the request message to the SAD_CREATEAPPL service operation.
• Receive one or more admission application templates with default values from AAWS.
• Present the one or more applications with the default values to the user through one or more user interface pages.
• Allow the user to enter in all required application details.

Get Fee Configuration

Application fees while optional are likely a common scenario that needs to be supported by the UI. At some point after application creation (where the Application Center assigned to the application will be known to the UI) but prior to application submission (when fees may be payable) the UI needs to obtain the Application Fee configuration options. The fee options dictate what fee payment preferences should be presented to the self service applicant by the UI. Options surrounding fee payment include:

1. Whether an application fee is required
2. Whether the applicant is given the option to “Pay later”.
3. Whether the applicant can be given the option to request a “Waiver” for the fee.

Based on what fee options are configured the UI needs to react appropriately and present the applicant with only the valid options and this will in some cases alter the UI page flow presented to the applicant.

A typical self service online application UI might perform the following actions:

• Collect the Academic Learning Area preferences that the applicant enters during the Application creation step.
• Use the preferences to create a request message for the List of Values Web Service, the request must include a look up of:
  1. The SAD_LISTVAL_VW record to retrieve the Application Fee options.
  2. The SSF_ADM_TND_VW record to retrieve the tendering options.
• Submit the request message to the SCC_LOV service.
• Receive the Application Fee configuration options.
• Act on the Application Fee configuration Options in the following way:
  o If the “Pay Later” option is activated, then present the applicant with the choice to pay later (the applicants choice is then used in the later Submit Application step).
  o Store all Application Fee configuration and Tendering configuration options in UI storage for use in later Present Fee and Options step.

Update Application

The online application launch page may show applications that the user has previously saved or submitted. The user may wish to submit a previously saved application.

A typical self service online application UI might perform the following actions:

• Present an “Update” hyperlink that the user can click to initiate the application update process.
• Formulate a request message.
• Submit the request message to the SAD_GETAPPL service operation.
• Receive an *in-process* or *draft* application from AAWS.
• Present the application to the user through one or more user interface pages.
• Allow the user to modify or enter the application details.

**Edit Application**

This activity represents the user completing details on one or more pages in the online application. This activity mostly involves the user interface presenting the page flow and collecting data from the user. If your institution has implemented attachments, then the returning user may wish to view any existing attachments associated with an application.

When the user edits the application data, the online application user interface may present dynamically generated prompts and dropdown list boxes. The online application may need to populate the values for these prompts and list boxes through calls to the *List-Of-Values* service. Dynamic prompting occurs in situations where the system uses the user input to display the user interface dynamically. An example of dynamic prompting is when the user selects an undergraduate career; the Program field on the online application displays only undergraduate programs for the user to select.

A typical self service online application UI might perform the following actions:

• Present a series of pages for the user to interact with.
• Optionally, formulate a dynamic *List-Of-Values* request message.
• Submit the request message to the SCC_GET_LOV service operation.
• Present the *List-Of-Value* results in the online application page fields.
• Optionally, formulate an attachment request message.
• Submit the request message to the SAD_GET_ATTACHMENT service operation.
• Decode the response from AAWS and present the binary content of the attachment in a suitable viewer.

**Save Application**

When the user is creating or updating an application, he or she may want to pause and save the application for later modification or submission. The saved application is an *in-process* or *draft* application and AAWS places this *draft* application in a staging area. Subsequently, the user may submit this application or the system administrator may delete this application.

A typical self service online application UI might perform the following actions:

• Present a “Save For Later” hyperlink on any page in the page sequence. Generally, any page after the user starts creating the application may display this link. The user clicks this link to save the application.
• Prompt the user that the application is saved and the academic institution might use the applicant information for marketing purposes (if prospect creation is configured to occur on save).
• Formulate a request message.
• Submit the request message to the SAD_SAVEAPPL service operation.
• Receive and process the response from AAWS:
  o On a *Success* response there are various options:
    1. Present the user with a confirmation page.
    2. Take the user back to the *Get Application Summary* page.
    3. Log out the user from the online application.
  o On an *Error* response:
1. Based on the error codes that AAWS returns, position the user at an appropriate page of the user interface to allow data correction.
2. Present validation error messages to the user.
3. Present system error messages to the user.

Submit Application

After a user completes the application, the user interface should present a Submit Application option. Also, at submission time, the interface can present any number of disclaimer pages to the user. Generally, after a user submits the application, the user cannot update the application and the user interface would need to enforce this rule using the application status value. The user interface can retrieve this value using the SAD_GETAPPLS service operation.

A typical self service online application UI might perform the following actions:

- Present a “Submit” button on the final online application page.
- Prompt the user with any required disclaimers.
- Formulate a request message. Note that if a fee is due and the applicant has opted to “Pay Later” during the Get Fee Configuration step then this intent must be indicated in the request message.
- Submit the request message to the SAD_SUBMITAPPL service operation.
- Receive and process a response from AAWS. There are three alternative responses that the UI needs to cater for:
  - A Success response and Fee Payment is NOT required:
    1. Present the user with a confirmation page.
    2. Take the user back to the Get Application Summary page.
    3. Log out the user from the online application.
  - A Success response and Fee Payment IS required:
    1. Extract the fee amount and currency code from the response.
    2. Proceed to the Present Fee and Options page.
  - An Error response:
    1. Based on the error codes that AAWS returns, position the user at an appropriate page of the user interface to allow data correction.
    2. Present validation error messages to the user.
    3. Present system error messages to the user.

Present Fee and Options

If the Get Fee Configuration step determines that there is a fee payable at submission time then this page flow must be presented to the applicant. Fees are calculated by Student Financials according to the Fee Code attached to the Application Center. This page flow needs to present the applicant with his/her payment options based on the values returned in the earlier Get Fee Configuration step. The options here will include the ability to specify the tender category (CC, check etc) and optionally the ability to request a Waiver if that is allowed. Partial payments are not allowed. Therefore the self service online application must not allow the applicant to change amount presented for payment.

A typical self service online application UI might perform the following actions:

- Present a page showing the:
  1. Amount payable.
  2. Currency code of the fee amount.
3. Optionally the ability to request a Waiver.
4. Optionally the ability to specify a Waiver Basis code to be associated with the Waiver request.
5. Optionally the ability for the applicant to attach one or more attachments to be used as supporting documentation for the Waiver request.

- Formulate a request message based on the choice of the applicant:
  1. If the applicant requests a Waiver then:
     - Add Waiver details to the request
     - Submit the request to the SAD_SUBMITAPPL Web Service
     - Process the response as in the Submit Application flow
  2. If the applicant chooses to proceed with payment
     - Submit the request message to the SSF_INITIATE_EPAYMENT web service
     - Receive and process the response from the payment service
     - The response may be either a HTTP redirect OR a HTML string (this is determined by the hosted payment configuration)
     - In the case of a redirect the UI needs to respect the HTTP redirect and it will also need to store state of the current logged on userid/password in persistent client storage (e.g. a cookie or database table)
     - In the case of a HTML string response the UI can handle this in a number of ways, some examples being:
       - Spawn a new child window to display the HTML string response
       - Incorporate the HTML response into a parent window frame
  3. The response style is configurable and customizable and by use of dynamic HTML. There is sufficient flexibility for the UI to initiate the Payment Processing flow in either a new or an existing browser window. The decision will hinge entirely on the desired user experience and the technical capabilities of the UI technology being employed.

- The end result of this activity is that the user will be presented with a browser page provided by the 3rd party Hosted Payment processor from where the applicant can proceed with payment as given in the Payment Processing flow.

**Payment Processing**

At this point the applicant has indicated that they wish to proceed with payment of a fee and they will have been presented with a browser window which contains a HTML page provided by the 3rd party hosted payment provider (e.g. the applicant will have been redirected to the 3rd party site).

The applicant would then proceed to enter payment options and submit or cancel the payment. The 3rd party provider would then redirect the applicant’s browser to one of three configured URLs:

- CANCEL – the URL destination for when the applicant cancels the payment
- SUCCESS – the URL destination for when the payment is completed successfully
- FAIL - the URL destination for when the payment is not completed successfully

In the event of a successful payment then one final step needs to occur in order to complete the payment cycle. The Complete ePayment web service needs to be invoked to finalize the payment and submit the application. The self service online application UI should display the appropriate message to the applicant based on the Complete ePayment (SSF_COMPLETE_EPAYMENT) response message.

A typical self service online application UI might perform the following actions:

- Formulate a request message
- Submit the request to the SSF_COMPLETE_EPAYMENT web service
• Present a page to notify the applicant of successful payment
• Optionally transfer the applicant back to the Online Application home page and initiate the Get Application Summary flow to show the latest status.

**ERROR HANDLING**

When the system validates the data during save and submit, if validation errors occur, the error handling feature returns a list of textual descriptions of those errors along with error IDs and the field names related to the errors. As much information as possible is returned to the user interface to present the errors to the user, and allow correction of the errors.

The user interface must perform the following steps to use the error handling feature:

- The UI must populate a unique ID into the tag called SCC_ENTITY_INST_ID in the request xml for each entity. In the event of error during data validation, error handling returns to the UI the unique ID (that is obtained from the request message) and the field name related to the error along with the error message text.
- The UI can use this unique error ID, the field name, and the error message to know what fields on what pages are in error. This allows the UI to highlight the fields with errors and allows the user to correct the data.

The unique ID is optional in the request schema. Therefore, institutions should populate the unique ID only if they want to uptake the error handling feature.

Institutions are free to populate the SCC_ENTITY_INST_ID in any format they please, but it is recommended that they use the official UUID format, 36 character ISO standard found at http://en.wikipedia.org/wiki/Universally_unique_identifier

**LOGOFF**

An option to leave the online application could potentially be provided from any page in the online application. Clicking the Logout hyperlink is a signal from the users that they intend to leave the system and that any sensitive information managed by the user interface should now be cleared.

A typical self service online application UI might perform the following actions:

- Present a "Logout" hyperlink that the user can click to log out of the online application.
- Clear the username and password combination from the local UI temporary storage area.
- Present the user with a logged off confirmation page.

**Building the Hosted Payment HTML object**

For a detailed account of the Hosted payment feature please see the Electronic Payment Integration Developers Reference Guide found at the following location:

https://support.oracle.com/CSP/ui/flash.html#tab=KBHome(page=KBHome&id=()).(page=KBNavigator&id=(viewingMode=1143&bmDocDocSrc=KB&bmDocTitle=PeopleSoft%20Enterprise%20Campus%20Solutions%209.0%20Feature%20Pack%202&bmDocID=968171.1&from=BOOKMARK&bmDocType=REFERENCE))

The HTML object is provided as a flexible means to allow a diverse range of UI technologies to interact with the Hosted payment feature. The concept is that the HTML object can be constructed using dynamic HTML. The HTML object can then be presented in a browser window to allow seamless flow between the Admissions Online Application and the 3rd party payment provider. In theory there should
be no page flow or User Experience which cannot be achieved using a JavaScript based HTML object. Note that the HTML object is not limited to JavaScript and can contain any proprietary client side scripting language as is dictated by the UI technology in use by the institution.

As a simple example the following HTML object could be used to auto-redirect the applicant’s browser to a 3rd party payment provider:

```html
<BODY onload="submitForm()">

<form name="myform" action="https://someserver/upay/web/index.jsp" method="post">

<input type="hidden" name="UPAY_SITE_ID" value="3">

<input type="hidden" name="TICKET" value="12345">

<input type="hidden" name="TICKET_NAME" value="12345">

</form>

<SCRIPT LANGUAGE="JavaScript">function submitForm()

{document.myform.submit();}

</SCRIPT>

</BODY>
```
Troubleshooting

Avoiding caching issues:

Your web server or browser may cache old content. One way to overcome this issue is to enclose the online application page content in a HTML wrapper as shown in the following code:

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<html>
<head>
<!-- BEGIN INSERT -->
<META HTTP-EQUIV="Expires" CONTENT="Mon, 04 Dec 1999 21:29:02 GMT"><!-- END INSERT -->
<!-- BEGIN INSERT -->
<HEAD><META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE"></HEAD><!-- END INSERT -->
<title>OLA Demo</title>
</head>
<body bgcolor=white lang=EN-US link="#3366cc" vlink="#9999cc" style='tab-interval:.5in' leftmargin=10 RIGHTMARGIN=10 alink="#0000cc">
<object width="800" height="500">
<param name="movie" value="OLA.swf">
<embed src="OLA.swf" width="800" height="500">
</embed>
</object>
</body>
</html>
```

Microsoft Internet Explorer 7 Trusted sites Issue:

When using Microsoft Internet Explorer 7, you may need to add your web server URL hostname to Internet Explorer 7's trusted sites. If you do not add the URL to the trusted sites, you will receive a security error: "Error in line 1 char 2 .......". This is a rather cryptic error, but adding your web server host to the trusted sites setup in your browser will correct this issue.

Adobe Flex Domain Security Issue:

This issue is applicable if you are using Adobe Flex to develop an online application.

If the user loads this online application from a particular domain, the online application cannot retrieve data from another domain. To avoid this issue, you can create a crossdomain.xml file that allows your online application to load content from other domains. Refer to the Adobe Flash or Adobe Flex website for more information on the crossdomain.xml file.
This chapter presents a scenario where you use the File Parser, AAWS, and Transaction Manager features to load applications into Campus Solutions from a text file. In this scenario we are using a subset of admissions application data found in a typical application load file.

To load applications from an external file:

1. Set up File Parser definitions.
2. Set up the Constituent transaction.
3. Run the File Parser process to load the data into staging tables.
4. Review the staged data.
5. Run the Search Match Post process.

### Setting Up File Parser Definitions

**Steps:**

- Create the File Parser definitions.
- Make any necessary additions and/or deletions to the File Parser definitions.

**Step 1:** Create the File Parser definitions.

The File Parser definition presented in the following sections serves as an example of how an institution might configure a bulk load.

**Step 2:** Make any necessary changes to the File Parser definitions:

*Note:* For more information on File Parser, see *PeopleSoft Enterprise Campus Community Fundamentals 9.0 People Book*: "Using the File Parser Process".

The File Parser definitions consist of several key components:
• **Field Conversion Definitions** – Identifies external file field values to convert before storing the data in the target staging table.

• **Context Definitions** – Identifies the parent-to-child relationship of the staging tables that will be used to hold the data.

• **File Mapping Definitions** – Associates fields on the external file with the context definition.

Use the Field Conversion Definition page to define field values in the external file that must be converted to internal values. Add any additional fields and complete the definition for the fields delivered. To navigate to the Field Conversion Definition page, select **Set Up SACR, System Administration, Utilities, File Parser, Field Conversion Definition**

![Field Conversion Definition](image)

**Figure 47: Field Conversion Definition page**

Use the Context Definition page to add any additional extension staging tables that need to be updated when loading data from the external file. You may have created these tables by performing steps listed in the Extending AAWS chapter. Ensure that you enter the correct Parent Record field value. To navigate to the Context Definition page, select **Set Up SACR, System Administration, Utilities, File Parser, Context Definition**.
Use the File Definition page to provide information about the external file. To navigate to the File Definition page, select Set Up SACR, System Administration, Utilities, File Parser, File Mapping Definition.

Use the File Layout page to define fields from the external file that will be loaded into the staging tables.
Use the Mapping page to associate fields defined on the File Layout page with staging table fields defined in the context definition.

Use the Preview Data page to test your external file with the File Parser Definitions. **Note:** The preview feature shows only the first row of each row type; therefore, you should use a small test file.
Figure 52: Preview Data page

Setting up Constituent Transaction

After you set up the file parser, set up a Transaction Manager Constituent transaction that represents data in the external file.

Use the Transaction Setup page to configure a Constituent Transaction Management (CTM) transaction. To navigate to the Transaction Setup page, select Set Up SACR, System Administration, Utilities, Constituent Transaction Mgmt, Transaction Setup.
Figure 53: Transaction Setup page for Bulk Load

Enter the Implementation Class Details as shown in the above screenshot.

Ensure that the following parameters are set up:

- **Transaction Status** is set to *Active*.
- **Online Transaction** is unchecked (the data will be added to the stage records offline).
- The **Process Search/Match**, **Batch** option is selected.
Running the File Parser Process

Run the File Parser process to load the application data from a text file into the staging tables.

In order to run the File Parser process, set up a run control with the File Mapping ID and filename information on the Run File Parser page. To navigate to the Run File Parser page, select Set Up SACR, System Administration, Utilities, File Parser, Run File Parser.

Run File Parser

Run Control ID: BATCH_FILE_LOAD

File Mapping ID: ADMISSIONS SAMPLE FILE 1

Figure 54: Run File Parser page

Reviewing Staged Data

After loading the application data into the staging tables, perform the following steps:

Steps:

- Review the staged constituent data for the BATCH_APPLICANTS transaction.
- Review the staged application data.

Step 1: Review the staged constituent data:

To navigate to the Constituent Staging component, select Student Admissions, Application Transaction Mgmt, Constituent Staging.

Enter the BATCH_APPLICANTS transaction code in the Constituent Staging search component to review the loaded data:
Use the Constituent Staging component to review and correct the constituent data if errors occurred:

**Figure 55: Constituent Staging component**

Save the data after making modifications.

**Step 2:** Review the Staged Application Data:

After reviewing the constituent data, use the Application Transactions component to review and correct the application data.

To navigate to the Application Transactions component, click the View Transaction Staged Data link in the Constituent Staging component or, alternatively, select *Student Admissions, Application Transaction Mgmt, Application Transactions*.

Save any changes.
Figure 56: Application Transactions component

Running the Search Match Post Process

After loading application data into staging tables, reviewing and correcting the staging data, the data needs to be posted into the Campus Solutions production tables.


Figure 57: Transaction Management Process component
It is important to note the following:

- In the screen shot, search match post process is performed only for **Submitted** data, which was loaded in the previous step.

- The process will run Search Match and, based on a match, post the person and application transaction data.
APPENDIX: DEVELOPERS REFERENCE

For detailed information related to the application classes used by the Entity Registry, refer to *PeopleSoft Enterprise Campus Community Fundamentals 9.0 PeopleBook*, "Setting Up Entity Registry", *Entity Application Class Reference*. 