Chapter: Using Admission Applications Web Services

This Recruiting and Admissions PeopleBook chapter provides an overview of Admission Applications Web Services (AAWS) and discusses:

- AAWS and List of Values framework.
- CTM Self-Service and Administrator modes.
- Integration with other Campus Solutions components.
- AAWS and Entity Registry.
- AAWS service operations.
- Payment integration service operations.
- Setting up attachments URL.
- Setting up hosted payment.
- Post installation troubleshooting tips.
- User interface considerations.
- AAWS Web Services Description Language (WSDL).
- Sample code for an online application.

Note. The content of this chapter has been extracted from the *AAWS User's Guide and AAWS Developer's Guide*. Also, note that the Campus Solutions PeopleBook chapters listed under the following See Also heading may get updated with future bundle releases.

Understanding AAWS

AAWS can be used by end user devices (such as a browser or a PDA), enterprise applications or any other third party software that can interface through standards based means.

AAWS can be used by:

- Oracle presentation technologies, such as Portal, Application Development Framework (ADF), and PeopleSoft components and pages.
- Oracle middleware (such as Enterprise Service Bus and BPEL).
- Oracle Applications (such as Enterprise, EnterpriseOne, and E-Business Suite).
- Third party presentation technologies.
- Third party middleware and applications.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Different academic institutions have different needs for capturing admission application data. Also, application data can vary within the same academic institution between various campuses, departments, careers, and types of applicants. For example, application data for international and domestic applicants can vary within the same institution. This creates the need for institutions to develop their own online application user interface that can interact with the Campus Solutions system. With AAWS, institutions can connect their online application with the Campus Solutions system.

AAWS provides the SAD_ADMISSIONS service that enables access to the Campus Solutions Recruiting and Admissions functionality. The SAD_ADMISSIONS service provides the following service operations:

Service Operation	**Description**
SAD_CREATEAPP	Create admission application
SAD_SAVEAPPL	Save admission application
SAD_SUBMITAPPL	Submit admission application
SAD_GETAPPLS	Get admission applications
SAD_GETAPPL	Get admission application
SAD_GETATTACH	Get admission attachments.

With these service operations, AAWS offers functionality in the following areas:

- Admission Applications: Users, such as applicants, can use an online application user interface to:
    - Create admission applications in the Campus Solutions system.
    - Save, update, or submit the applications.
    - Retrieve an application to make further updates.
- Application Attachments: Online application users can associate attachments of various formats with an admission application. Administrators can track these attachments using the Application Transactions and Maintain Application components.

Note that when an online application invokes SAD_SAVEAPPL or SAD_SUBMITAPPL service operation, the system generates an admission transaction. This admission transaction contains the admission application data that the user has saved or submitted. An admission transaction is defined and configured using the CTM pages and Application Configuration component. You can use the delivered Application Transactions and CTM Constituent Staging components to review, edit, and post the application and applicant data in an admission transaction. Additionally, you can choose to post the transactions in batch.

Constituent Transaction Management (CTM), which is a Campus Community feature, provides the following two service operations related to New User Registration (NUR):

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Service Operation	Description
SCC_USERREG_CREATEACCT	Create user account
SCC_USERREG_AUTHENTICATE	Authenticate user

You can use these NUR service operations to:

- Provide usernames and passwords for access to your online application.
- Authenticate users who want to retrieve a saved application for further edits and submission.

The List of Values framework is also a Campus Community feature. The framework provides the SCC_GET_LOV service operation. An online application can use this service operation to retrieve the list of values from the Campus Solutions system for data entry fields on the online application. This is useful only if the online application user interface is built outside of your Campus Solutions system.

Service Operation	Description
SCC_GET_LOV	Get List of values

AAWS uses the following Student Financials payment integration service operations to enable application fees calculation and payment:

Service Operation	Description
SSF_INITIATE_EPAYMENT	Initiate an ePayment.
SSF_COMPLETE_EPAYMENT	Complete an ePayment.

All of these service operations use PeopleTools Integration Broker for interacting with the online application. Any online application user interface that is web-service enabled and SOAP compliant can access these service operations. AAWS adheres to open web service standards currently supported in the latest release of PeopleTools.

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.

Assessing Staff Skills

Developers working on the implementation of AAWS should have strong skills in:

- A tool or technology to build and deploy an online application user interface
- PeopleSoft Integration Broker.
- PeopleCode.
- Web services concepts mainly XML, SOAP, and WSDL.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- Campus Community Entity Registry functionality.
- Recruiting and Admissions functionality.
- Campus Community Constituent Transaction Management functionality

Setups for AAWS

The key setup areas that apply to AAWS:

- PeopleTools Integration Broker.
- Campus Community Constituent Transaction Management (CTM).
- Campus Community Entity Registry.
- Application Configuration (Configuring admission transactions).
- Campus Community List of Values.
- Attachments URL.
- Hosted Payment.

[[Bundle 27: The following See Also subsection has been added]]

See Also

The following documentation is posted to My Oracle Support and should be reviewed as part of your AAWS implementation:

- Using the AAWS Sample Online Application appendix (released as part of the Recruiting and Admissions Bundle 27 documentation updates)
- AAWS Documentation - Proactive Information and Known Issues (Doc ID 1427475.1)
- Managing PeopleSoft Admission Transactions chapter (the latest version of this chapter was posted as part of Recruiting and Admissions Bundle 27 documentation updates)
- *PeopleSoft Student Financials 9.0 PeopleBook*, Using Student Financials Web Services for Hosted ePayment Transactions
- Working with Constituent Transaction Management chapter (the latest version of this chapter was posted as part of Campus Community Bundle 27 documentation updates)
- Using New User Registration Framework chapter (released as part of the Campus Community Bundle 27 documentation updates)
- Setting Up Entity Registry chapter (the latest version of this chapter was posted as part of Campus Community Bundle 27 documentation updates)
- How to Test AAWS and Hosted Payment SSF_INITIATE_EPAYMENT Service Operation Tester [Doc ID 1349409.1]

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- Campus Solutions 9.0 Generic Service Tester Tool [Doc ID 1478817.1]
- Setting Up List of Values chapter which was posted to My Oracle Support as part of the Campus Community documentation updates for Bundle 22
- Enterprise PeopleTools PeopleBook: Integration Broker

AAWS and List of Values Framework

An online application user interface that you construct outside of the PeopleSoft Campus Solutions database may contain data fields that have predefined or *prompt* values. The user interface can use such fields to control and streamline data entry for the user. You set up these values in the PeopleSoft Campus Solutions database. To display these values on the user interface, whether it is for a prompt field or a field that contains translate values, you can use the List of Values service (SCC_GET_LOV). The service recognizes and provides the values that are used to populate drop-down list boxes and for validating selections on a user interface.

CTM Self-Service and Administrator Modes

In the Self-Service mode, an applicant creates and submits an admission application through AAWS. In the Administrator mode for AAWS, the administrative user creates and submits an application on behalf of an applicant. The administrator may create the application for an existing EMPLID (if the administrator knows the applicant's ID when creating the application), or for an unknown or new applicant.

In the Administrator mode, the system associates the application with the administrator's user ID. You can view this association on the Application Transactions component when reviewing the submitted applications. In other words, you can use the Application Transactions component to see which administrator has created an application for an applicant.

In the Self-Service mode, the system associates the application with the applicant's user ID.

In the Administrator mode, the user must provide an institution and application center to create an application. The system enforces application center and academic structure security for the administrative user.

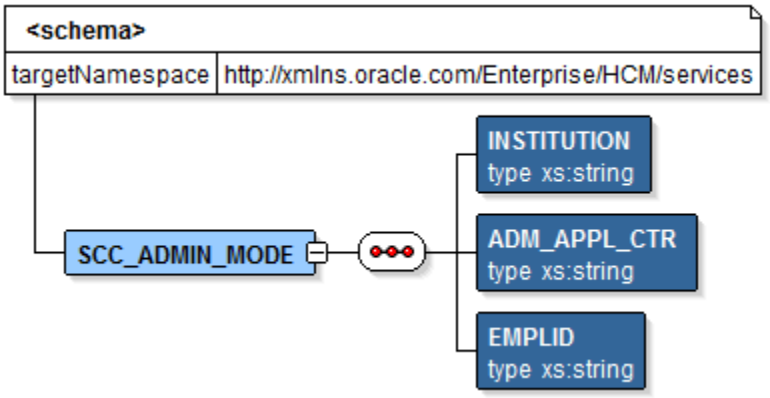
If an administrator saves an application, another administrator cannot submit it through AAWS. However, the other administrator can use the Application Transactions staging component to update the application status to submitted and post the application.

In the Administrator mode, the Get Applications service operation call returns only in-progress applications, that is, applications which have not been posted or cancelled. It is optional for an administrator to provide an EMPLID for a Get Applications call. If the administrator does not provide an applicant's EMPLID, the system returns all the in-progress applications.

The Administrator mode is activated by the presence of optional tag <SCC_ADMIN_MODE> in an incoming message.

The following diagram shows the SCC_ADMIN_MODE parameters:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



SCC_ADMIN_MODE parameters

For example, the following is a request message for the service operation SAD_CREATEAPPL that contains the SCC_ADMIN_MODE tag:

```
<?xml version="1.0"?>
<SAD_CREATEAPPL_REQ xmlns="http://xmlns.oracle.com/Enterprise/HCM/services">
  <SCC_ADMIN_MODE>
    <INSTITUTION>PSUNV</INSTITUTION>
    <ADM_APPL_CTR>BUSN</ADM_APPL_CTR>
  </SCC_ADMIN_MODE>
  <COUNTRY></COUNTRY>
  <CITIZENSHIP_STATUS></CITIZENSHIP_STATUS>
  <PREFERENCES>
    <PREFERENCE>
      <INSTITUTION>PSUNV</INSTITUTION>
      <ACAD_CAREER>BUSN</ACAD_CAREER>
      <ACAD_PROG>GRFIN</ACAD_PROG>
      <ACAD_PLAN>FINANC-MBA</ACAD_PLAN>
      <ACAD_SUB_PLAN></ACAD_SUB_PLAN>
      <ADMIT_TYPE>FYR</ADMIT_TYPE>
      <ADMIT_TERM>0630</ADMIT_TERM>
    </PREFERENCE>
  </PREFERENCES>
</SAD_CREATEAPPL_REQ>
```

Integration with Other Campus Solutions Components

AAWS interacts with the following Campus Solutions components:

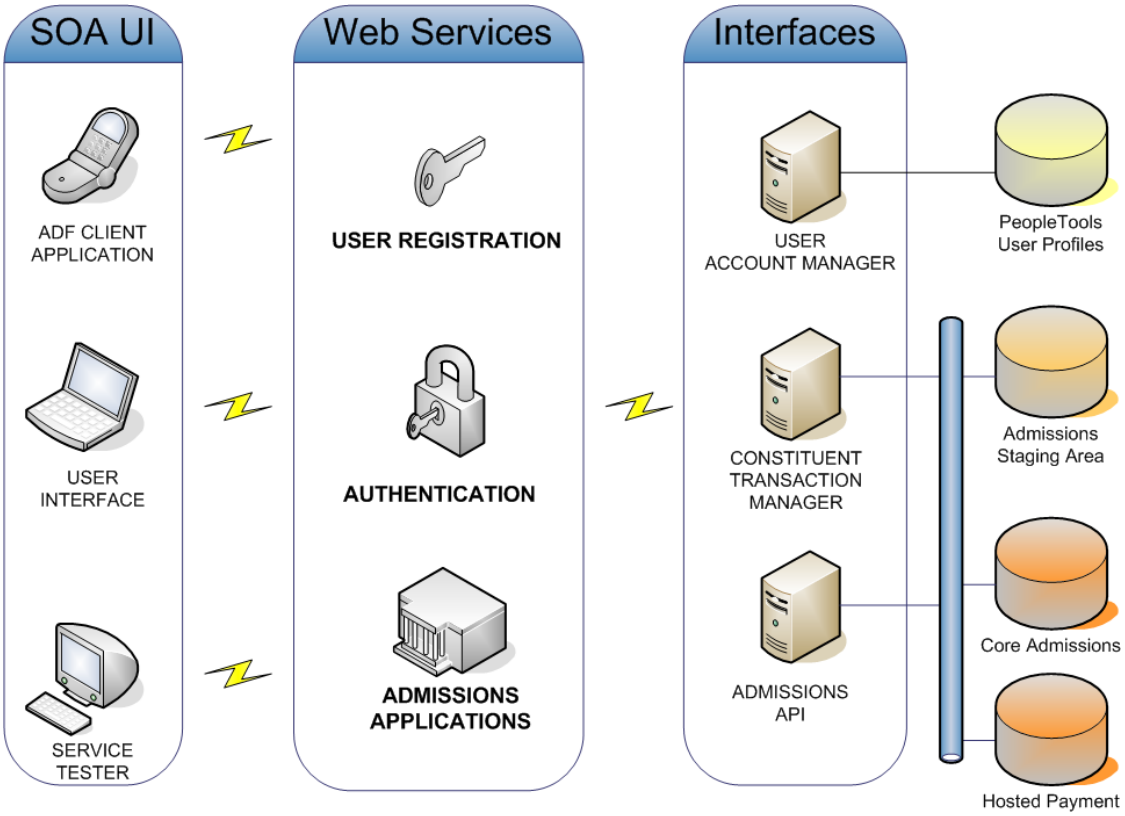
- User Account Manager
- Constituent Transaction Manager
- Admissions API
- Student Financials Hosted Payment

This section discusses the application packages and classes for these components. If you want to customize AAWS, you may need to modify some of these classes and packages. For instance, if you want to integrate a non PeopleSoft user management system with AAWS, a new application class should

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

be created that implements SCC_IDENTITY_MGR:INTFC:IIdentityManager. The application class must then be added to the table SCC_IDM_INST.

The following diagram shows the AAWS architecture:



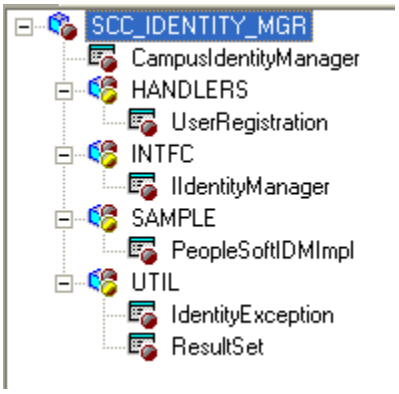
AAWS Architecture

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 graphics show the core application packages and classes which support the User Account Manager:

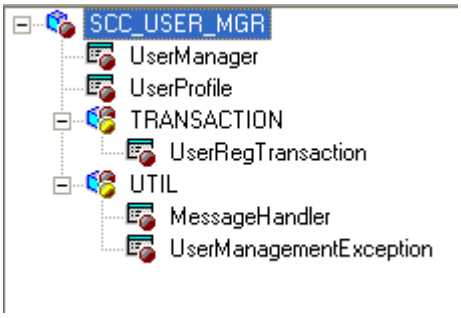
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



Campus Identity Manager

Class	**Description**
CampusIdentityManager	Manages all aspects of Self Service Transactions interactions with any configured Identity management system or New User Registration module. Call getInstance() on this class to get access to a concrete IDM implementation.
HANDLERS:UserRegistration	Implements the User Registration Web Service, the register & authenticate operation.
INTFC:IIdentityManager	Represents the functionality available through any typical generic Identity Manager product. All access to IDM functions will occur through this interface whether from code level API calls or webservices.
SAMPLE:PeopleSoftIDMImpl	The peoplesoft default implementation of IIdentityManager. Performs all Identity Management tasks against the peoplesoft db.
UTIL:IdentityException	All exceptions generated from within the IdentityManager interface should extend this class.
UTIL:ResultSet	Represents a tabular data object that is the basis for many of the Identity Manager operations.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



User Profile Manager

Class	Description
UserManager	Main API for user manager.
UserProfile	A data object to store user profile information.
TRANSACTION:UserRegTransaction	This handles the posting activities for User Registration sourced data as directed by the Constituent Staging framework. Note: This is not the implementation currently used.
UTIL:MessageHandler	Handles any messages, warnings, errors generated during User Management.
UTIL:UserManagementException	All exceptions generated from within the UserManager interface should extend this class.

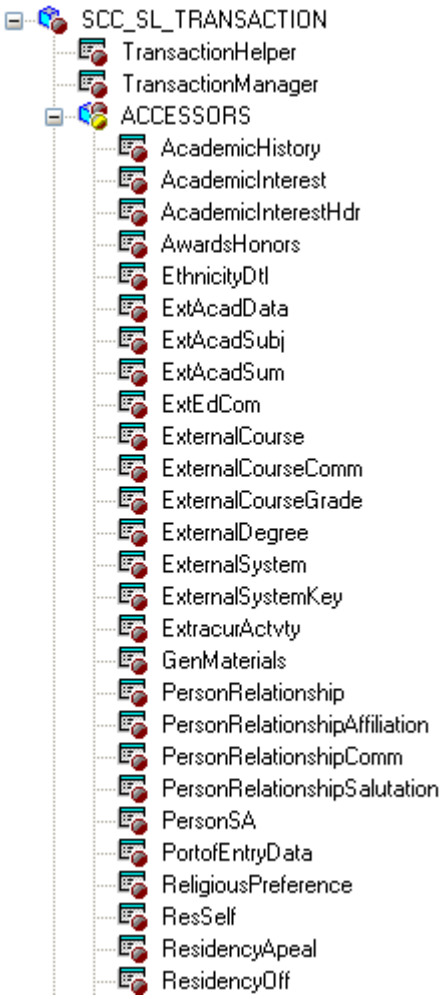
Constituent Transaction Manager

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 graphics show the application packages and classes in the Constituent Transaction Manager component:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



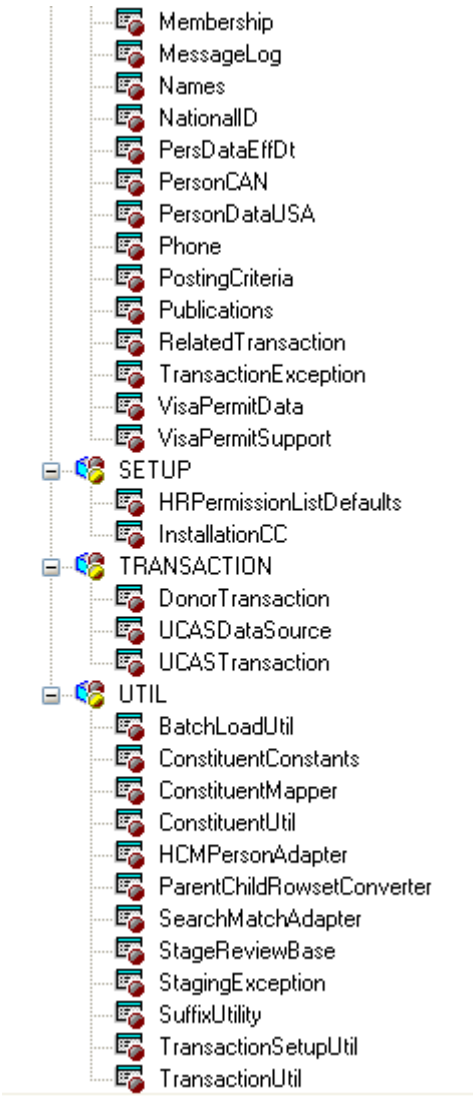
Constituent Transaction Manager (1 of 3)

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- ResidencyOff
- StdntCareer
- TestScore
- TestScoreComponent
- UCAS
- WorkExperience
- CountryExtensions
  - CanExtAcadData
  - CanReportStdnt
  - ChessnAUS
  - CitizHist
  - EduSubjNLD
  - HEPers
  - HESAPers
  - HEStdntBRN
  - HEStdntNLD
  - NamesNLD
  - PersNZL
  - PrevHepAUS
  - ScholarNLD
  - StdInfoNLD
  - StdntBankAccNLD
  - StdntCorresNLD
  - StdntData
  - StdntEduNLD
  - StdntNationalitiesNLD
  - TribalAffilNZL
  - Year12AUS
- COMPONENTS
  - SCC_CONST_STG
  - SCC_TRANSAC_DTLS
  - SCC_TRANSAC_POST
- INTFC
  - AbstractDataSource
  - AbstractTransaction
  - Address
  - Citizenship
  - CitizenshipPassport
  - Constituent
  - DefaultConstituent
  - Disability
  - DiversEthnic
  - Diversity
  - DriversLicense
  - DriversLicenseType
  - EmailAddresses
  - EmergencyContact
  - EmergencyPhone
  - IConstituent
  - ITransaction
  - Language
  - LicenseAndCertificate
  - Membership

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Constituent Transaction Manager (2 of 3)



Constituent Transaction Manager (3 of 3)

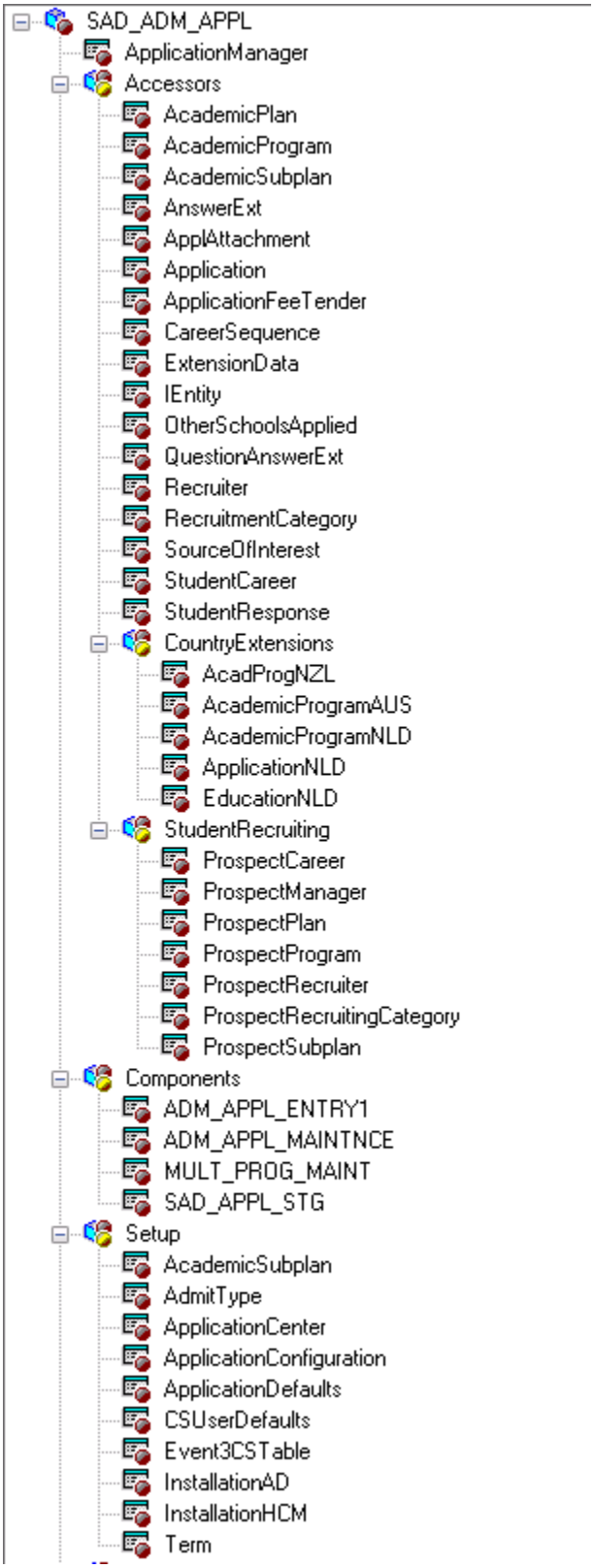
Admissions API

The Admissions 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.

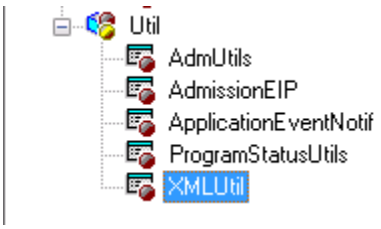
The following graphics show the application packages and classes in the Admission Applications Manager component:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



Admission Applications Manager (1 of 2)

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



Admission Applications Manager (2 of 2)

The Entity Registry feature enables academic institutions to create their own application data extensions (that is, you can add entirely new application child entities).

AAWS and Entity Registry

AAWS uses the Entity Registry feature to exchange data between the Campus Solutions system and the online application.

This section discusses the delivered Application and Constituent entities.

Application

The message SCC_ENTITY_APPLICATION contains the complete view of an admission application. The message uses the schema generated by the Application entity. The system derives the Application entity from the complete set of data available within the Campus Solutions Recruiting and Admissions core data model.

Use the PeopleTools Schema page to access the SCC_ENTITY_APPLICATION message schema (PeopleTools, Integration Broker, Integration Setup, Messages, Schema).

Constituent

The message SCC_ENTITY_CONSTITUENT represents the core constituent information associated with the person processed through AAWS. The system derives the Constituent entity from the complete set of person related data available in both the Campus Solutions and Human Resources person data models.

Use the PeopleTools Schema page to access the SCC_ENTITY_CONSTITUENT message schema (PeopleTools, Integration Broker, Integration Setup, Messages, Schema).

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

AAWS Service Operations

This section discusses the AAWS service operations that are delivered under the service called SAD_ADMISSIONS.

Note: The schema diagrams in this section may get updated with future bundle releases.

SAD_CREATEAPPL

This service operation enables the online application user to create an admission application in the Campus Solutions system. Using this service may result in one or more applications being created. It can also result in multiple applicant *preferences* being created within the same application, for example, entry of another program or plan for an existing application is a preference.

Details:

Description:	The Create Application service operation call results in one or more admission applications being initialized. The application center settings on the Application Configuration component control whether multiple applications (or *preferences*) can be created. For example, if you do not have an application center set up for an applicant who selects an Undergraduate career, Liberal Arts program, and History plan, the service operation throws an error. The service operation populates the admission application fields based on the settings on the User Defaults and Installation - AD components. At application creation time the operation does not physically save the data to the admissions staging tables. The save process does not occur until a later point in time when the user selects a save or submit option on the online application. Therefore, this operation is more concerned with the initialization of an application, driven by configuration, rather than creating a physical application in the database.
Users:	• An applicant using an online application. • An administrator using an online application for creating an application on behalf of an applicant who has or does not have an EMPLID.

Processing:	The service operation performs the following steps:
	
	1. Validate that the user is authenticated (and is not a *Guest*)
	2. If Administrator mode then:
	  ○ Validate the required input administrator parameters.
	  ○ Validate that the administrator is authorized to access the application center, institution, career, program and plan.
	3. Validate the required input parameters for application creation.
	4. Determine the transaction that must be used to process the

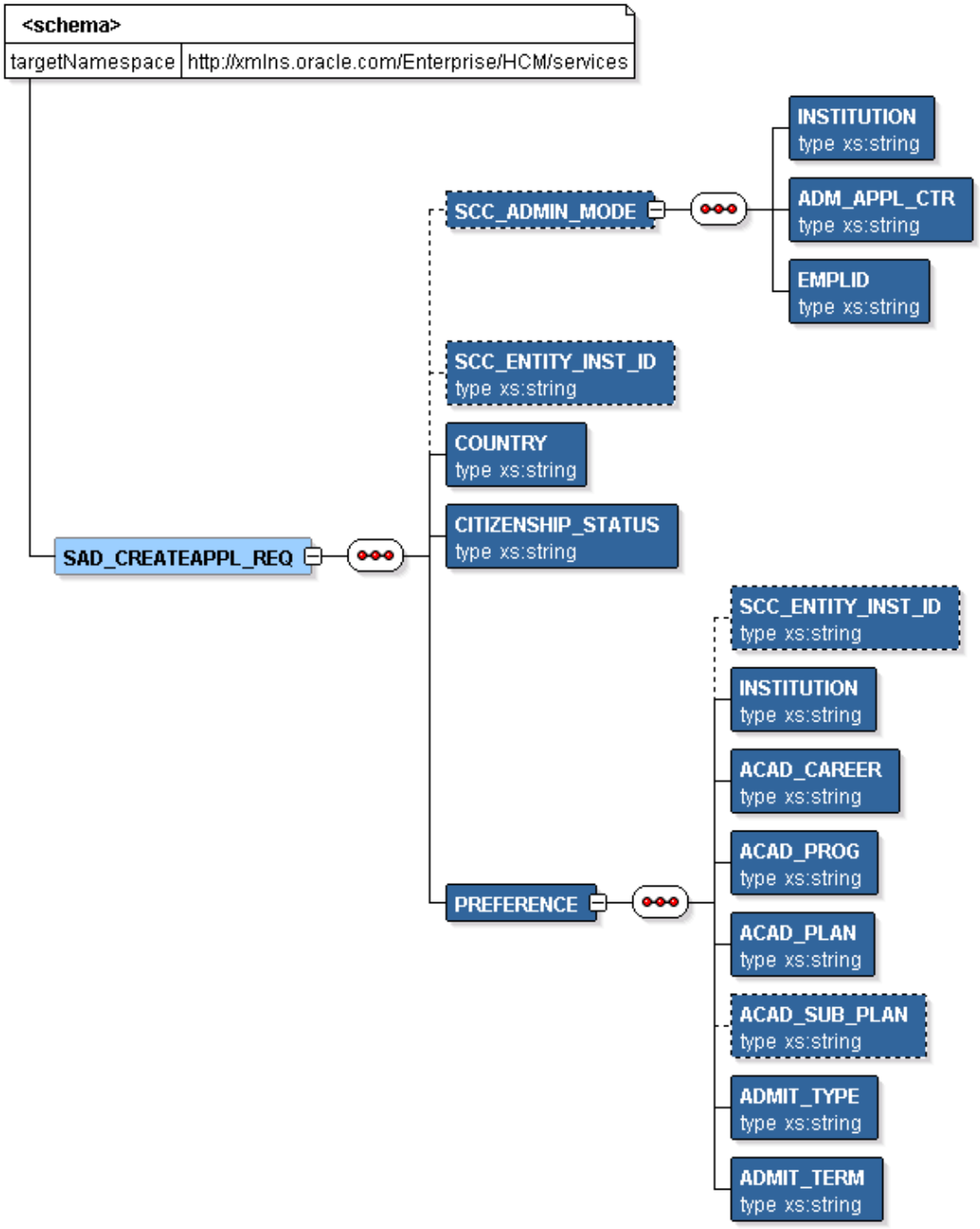
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	Institution/Academic Career combination. Define transactions on the Transaction Setup page. 5. Begin a new transaction for the request. Assign a new TEMP_ID to the constituent. 6. Using the supplied input parameters, look up the Application configuration component settings to determine how many application preferences should be created for this request. 7. Return response though the output parameters.
Output:	After processing, the operation creates an admission application with default data and preferences. The user can subsequently update and submit this application through the Save and Submit service operations.
Error Conditions:	The following conditions result in a service error: • User is not authenticated (for example, if an anonymous *Guest* tries to sign into the online application, the service throws an error message). • Administrative user is not authorized to access the application center, institution, career, program or plan. • Required input parameters have not been supplied by the user (see the graphic titled *SAD_CREATEAPPL_REQ Message Parameters* and the "AAWS and Entity Registry" section).

Input Message: SAD_CREATEAPPL_REQ

The following diagram shows the input parameters that the SAD_CREATEAPPL service operation receives:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



SAD_CREATEAPPL_REQ Message Parameters

Required data that the online application must supply to the service operation in *Non-administrator (Applicant) mode*:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- Institution
- Academic Career
- Admit Term
- Academic Program
- Academic Plan

Required data that the online application must supply to the service operation in *Administrator mode*:

- Institution
- Admission Application Center
- Academic Career
- Admit Term
- Academic Program
- Academic Plan
- Country and Citizenship Status (if your application mapping requires them).

The following is an example of the SAD_CREATEAPPL_REQ message that the SAD_CREATEAPPL service operation receives from an online application:

```
<?xml version="1.0"?>
<SAD_CREATEAPPL_REQ>
  <COUNTRY>AUS</COUNTRY>
  <PREFERENCE>
    <INSTITUTION>PSUNV</INSTITUTION>
    <ACAD_CAREER>UGRD</ACAD_CAREER>
    <ACAD_PROG>AA</ACAD_PROG>
    <ACAD_PLAN>ARTHIST</ACAD_PLAN>
    <ADMIT_TYPE>FYR</ADMIT_TYPE>
    <ADMIT_TERM>0747</ADMIT_TERM>
  </PREFERENCE>
</SAD_CREATEAPPL_REQ>
```

Output Message: SAD_CREATEAPPL_RESP

When the Integration Broker receives the SAD_CREATEAPPL_REQ message, it responds with the SAD_CREATEAPPL_RESP message.

The following diagram shows the output parameters that the SAD_CREATEAPPL service operation passes to the online application:

SAD_CREATEAPPL_RESP Message Parameters

Note that the SAD_CREATEAPPL_RESP schema includes the Application entity (ADM_APPL_DATA) and Constituent entity XSD schemas.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

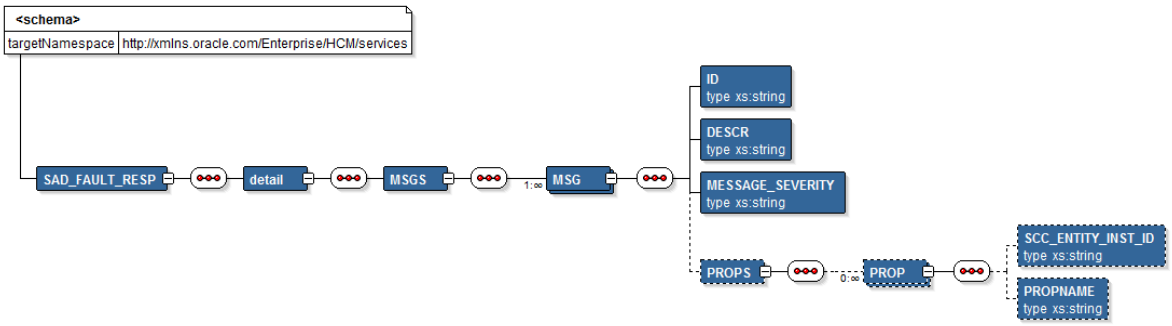
The following is an example of the SAD_CREATEAPPL_RESP message that the SAD_CREATEAPPL service operation transmits to the online application:

```
<?xml version="1.0"?>
<SAD_CREATEAPPL_RESP>
  <PREFERENCE>
    <ADM_APPL_DATA>
     <!-- Application data shape -->
    </ADM_APPL_DATA>
    <CONSTITUENT>
     <!-- Constituent data shape -->
    </CONSTITUENT>
  </PREFERENCE>
</SAD_CREATEAPPL_RESP>
```

Fault Message: SAD_FAULT_RESP

If the service operation encounters an error condition, it responds with the SAD_FAULT_RESP message.

The following diagram shows the SAD_FAULT_RESP parameters.



SAD_FAULT_RESP Message Parameters

The following is an example of the SAD_FAULT_RESP message that the SAD_CREATEAPPL service operation transmits to the online application:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>300</faultcode>
      <faultstring>A Error occurred processing this request</faultstring>
      <detail>
        <MSGS>
          <MSG>
            <ID>14200-556</ID>
            <DESCR> Validation error in preference
PSUNV/UGRD/AA/MUSIC/VOIC/FYR/0690</DESCR><PROPS/>
          </MSG>
          <MSG>
            <ID>14200-556</ID>
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

```
            <DESCR> Tag ACAD_SUB_PLAN has an invalid value VOIC. Check and try
again.</DESCR><PROPS><PROP><SCC_ENTITY_INST_ID>PREFERENCE ROW
1</SCC_ENTITY_INST_ID><PROPNAME>ACAD_SUB_PLAN</PROPNAME></PROP></PROPS>
          </MSG>
        </MSGS>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


SAD_SAVEAPPL

This service operation enables the user to save an admissions application to the Campus Solutions system (for later submission). The service operation initially posts the application to the admissions staging tables. Applications in the staging tables may at some point of time become eligible to be posted to the admissions production tables.

Details:

Description:	The Save Application service operation call results in the given admissions application being saved to the Campus Solutions system. The application may be partially complete when the online application user decides to use a Save option that triggers a call to this service operation. The service operation saves the application using an *Update Via Replacement* approach; in other words, the SAD_SAVEAPPL operation deletes any pre-existing application (based on a unique key) for the applicant and then saves the new (or current) version of the application. When saving an application, the operation assigns an application number (ADM_APPL_NBR) to the application for tracking. The application must pass through validation checks before it gets saved. The service operation validates all aspects of the application data including data entered and translate values selected by the user on the online application, and custom business validation rules that the institution has incorporated into the online application. The service operation saves the valid applications to the admissions staging tables. If the constituent (person) who initiated the application save process is known (for example, the constituent has an EMPLID), then the constituent portion of the application is immediately eligible to be posted to the Campus Solutions and HCM person production tables. Otherwise, the constituent data remains in the staging tables until addressed by an administrator. In any case, the application data itself is saved to the staging tables and is not posted to admissions production tables until a later *submit* occurs. Applications may include attachments. The online application must

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	encode the attachment in Base64 format before passing the attachment to Campus Solutions.
Users:	• An applicant using an online application. • An administrator using an online application for creating an application on behalf of an applicant or a student.

Processing:	The service operation performs the following steps:
	
	1. Validate that the user is authenticated (and is not a *Guest*)
	2. If administrator mode, then:
	  1. Validate the required input administrator parameters.
	  2. Validate that the administrator is authorized to access the application center, institution, career, program and plan.
	  3. Use the EMPLID supplied by the administrator for the transaction.
	3. Validate that the applicant is authorized to save the application (for example, verify whether the applicant is the same person who created the application in the call to the SAD_CREATEAPPL service operation).
	4. If administrator mode, then use the input parameter Application Center value for the application.
	5. If applicant mode, then use the input parameters and the settings on the Application Configuration component to assign an application center to the application.
	6. To enter the Application Citizenship details, use the applicant details in the input parameters, if necessary.
	7. To enter the Constituent Institution (needed for Data Update Rules), use the Admissions Application Institution input parameter value.
	8. Validate the constituent and application data using the Constituent and Application entity schemas.
	9. Associate the TEMP_ID (assigned during the SAD_CREATEAPPL processing) with the application.
	10. Save the validated constituent and application data to the staging tables.
	11. Add the assigned ADM_APPL_NBR to the response message.
	12. If the constituent is not known to the Campus Solutions system (for example, the applicant does not have an EMPLID):
	  1. Run the Search Match process according to the Search Match settings on the Transaction Setup component.
	  2. Add or update a HCM person record according to Search Match configuration and apply the necessary data update rules during this processing.
	13. Create or update a prospect record in the production tables according to the settings on the Application Configuration component.

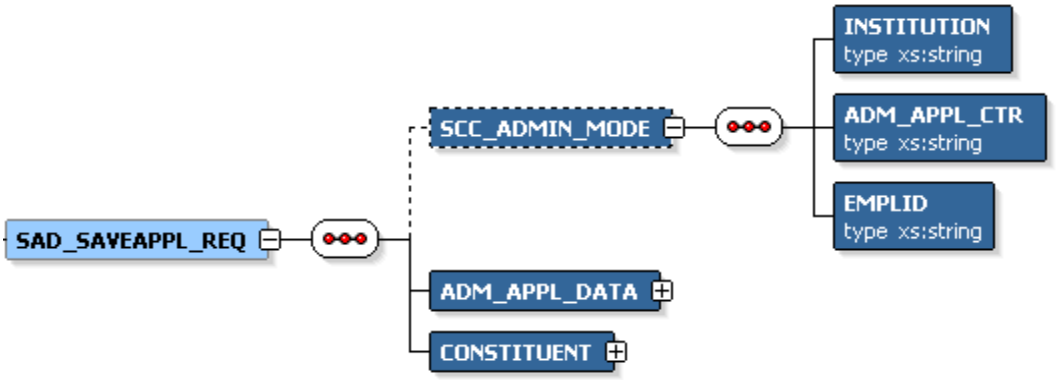
Output:	After completing the processing steps, the service operation: 1. Posts the valid application data to the Campus Solutions

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	admissions staging tables. 2. Posts the valid constituent data to the Transaction Manager constituent staging tables. 3. Optionally, creates or updates a HCM person record for a valid application. 4. Optionally, creates a prospect record for a known applicant.
Error Conditions:	The following conditions result in a service error: • User is not authenticated (for example, if an anonymous *Guest* tries to sign into the online application, the service throws an error message). • User is not authorized to access the application (for example, if the applicant is trying to access an application created by another applicant). • Application is invalid (that is, the application fails validation processing). • Required input parameters have not been supplied by the user (see the graphic titled *SAD_SAVEAPPL_REQ Message Parameters* and the "AAWS and Entity Registry" section).

Input Message: SAD_SAVEAPPL_REQ

The following diagram shows the input parameters that the SAD_SAVEAPPL service operation receives from an online application:



SAD_SAVEAPPL_REQ Message Parameters

Note that the SAD_SAVEAPPL_REQ schema includes the Application entity (ADM_APPL_DATA) and Constituent entity XSD schemas.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Required data that the online application must supply to the service operation in *Non-administrator (Applicant) mode*:

- ADM_APPL_DATA
- CONSTITUENT

Required data that the online application must supply to the service operation in *Administrator mode*:

- Institution
- Admission Application Center
- ADM_APPL_DATA
- CONSTITUENT

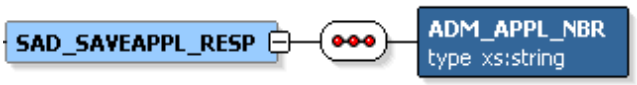
The following is an example of the SAD_SAVEAPPL_REQ message that the SAD_SAVEAPPL service operation receives from an online application:

```
<?xml version="1.0"?>
<SAD_SAVEAPPL_REQ>
  <ADM_APPL_DATA>
   <!-- Application data shape -->
  </ADM_APPL_DATA>
  <CONSTITUENT>
   <!-- Constituent data shape -->
  </CONSTITUENT>
</SAD_SAVEAPPL_REQ>
```

Output Message: SAD_SAVEAPPL_RESP

When the Integration Broker receives the SAD_SAVEAPPL_REQ message, it responds with the SAD_SAVEAPPL_RESP message.

The following diagram shows the output parameters that the SAD_SAVEAPPL service operation passes to the online application:



SAD_SAVEAPPL_RESP Message Parameters

The following is an example of the SAD_SAVEAPPL_RESP message that the SAD_CREATEAPPL service operation transmits to the online application:

```
<?xml version="1.0"?>
<SAD_SAVEAPPL_RESP>
  <ADM_APPL_NBR>00023456</ADM_APPL_NBR>
</SAD_SAVEAPPL_RESP>
```

Fault Message: SAD_FAULT_RESP

Refer to the SAD_FAULT_RESP message example in the SAD_CREATEAPPL section.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SAD_SUBMITAPPL

This service operation enables the user to submit a completed admissions application to the Campus Solutions system. The service operation initially saves the applications to the admissions staging tables and subsequently posts the staged application data to the admissions production tables.

Details:

Description:	The Submit Application service call results in the given admissions application being submitted to the Campus Solutions system. The application must be complete (according to the institution's data capture requirements). The online application user interface and the underlying Campus Solutions data model ensure that the submitted admission application is complete. The application submit process involves first validating the data, then saving the data to the admissions staging tables, and finally transferring the staged data into the admissions production tables. When initially saving the application to the staging tables, if the application does not have an application number, the service operation assigns an application number (ADM_APPL_NBR) to the application for tracking purpose. The operation does not assign an application number if another operation (for example, the SAD_SAVEAPPL operation) has already assigned an application number. The admission application must pass through validation checks before it gets submitted. The service operation validates all aspects of application data including data entered and translate values selected by the user on the online application and custom business validation rules that the institution has incorporated into the online application. If the constituent who initiated the application submit process is known to the Campus Solutions system (for example, the constituent has an EMPLID), then the constituent and application data is eligible to be posted to the Campus Solutions admissions and person production tables. Otherwise, the application remains in the staging tables until addressed by an administrator. When configured to do so, the constituent can pay an application fee and/or request an application fee waiver. Applications may include attachments. The online application must encode the attachment in Base64 format before passing the attachment to Campus Solutions.
Users:	• An applicant using an online application. • An administrator using an online application for creating an application on behalf of an applicant or a student.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Processing	This service operation performs the following steps:
	
	1. Validate that the user is authenticated (and is not a *Guest*)
	2. If administrator mode then:
	  1. Validate the required input administrator parameters.
	  2. Validate that the administrator is authorized to access the application center, institution, career, program and plan.
	  3. Use the EMPLID supplied by the administrator for the transaction.
	3. Validate that the applicant is authorized to submit the application (for example, verify whether the applicant is the same person who created the application in a call to the SAD_CREATEAPPL service operation.)
	4. If administrator mode, then use the input parameter Application Center value for the application.
	5. If applicant mode, then use the input parameters and the settings on the Application Configuration component to assign an application center to the application.
	6. To enter the Application Citizenship details use the applicant details in the input parameters, if necessary.
	7. To enter the Constituent Institution (needed for data update rules) use the Admissions Application Institution input parameter value.
	8. Validate the constituent and application data using the Constituent and Applicant entity schemas.
	9. Associate the TEMP_ID (assigned during the SAD_CREATEAPPL processing) with the application.
	10. Save the validated constituent and application data to the staging tables.
	11. If the constituent is not known to the Campus Solutions system (for example, the applicant does not have an EMPLID):
	  1. Run the Search Match process according to the Search Match settings on the Transaction Setup component.
	  2. Add or update a HCM Person record according to Search Match configuration and apply the necessary data update rules during this processing.
	12. If Fee payment is required but has not yet been paid:
	  1. Call the Student Financials Fee Calculation API to calculate the fee amount.
	  2. If Fee Waivers are allowed then allow the application to progress to submitted status
	  3. If Fee Waivers are Not allowed then
	  1. Set the application data to Saved Status
	  2. Add the Fee details to the response message
	  3. The caller must handle this response by initiating the payment services. Note it is possible for Fee calculation to result in a zero (0) amount in which case the caller should *not*

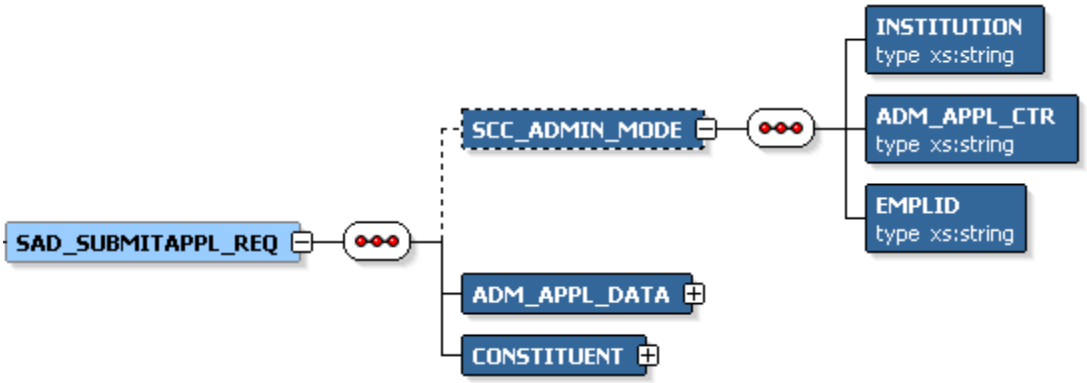
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	initiate the payment service. 13. Create or update a prospect record in the production tables according to settings on the Application Configuration component. If the constituent is known (for example, the applicant has an EMPLID), post the application data into the admissions production tables and apply the necessary data update rules during this processing.
Output:	After completing the processing steps, the service operation: 1. Posts the valid application data to the Campus Solutions admissions staging tables. 2. Posts the valid constituent data to the Transaction Manager constituent staging tables. 3. Transfers the application and constituent data from the staging tables to the Campus Solutions person and admissions production tables. 4. Optionally, creates or updates a HCM person record for a valid application. 5. Optionally, creates a prospect record for a known applicant.
Error Conditions:	The following conditions result in a service error: • User is not authenticated (for example, if an anonymous *Guest* user tries to sign into the online application, the service operation throws an error message.) • User is not authorized to access the application (for example, the applicant is trying to access an application created by another applicant). • Application is invalid (that is, the application fails validation processing). • Required input parameters have not been supplied by the user (see the graphic titled *SAD_SUBMITAPPL_REQ Message Parameters* and the "AAWS and Entity Registry" section).

Input Message: SAD_SUBMITAPPL_REQ

The following diagram shows the input parameters that the SAD_SUBMITAPPL service operation receives from an online application:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



SAD_SUBMITAPPL_REQ Message Parameters

Note that the SAD_SUBMITAPPL_REQ schema includes the Application entity (ADM_APPL_DATA) and Constituent entity XSD schemas.

Required data that the online application must supply to the service operation in *Non-administrator (Applicant) mode*:

- ADM_APPL_DATA
- CONSTITUENT

Required data that the online application must supply to the service operation in *Administrator mode*:

- Institution
- Admission Application Center
- ADM_APPL_DATA
- CONSTITUENT

The following is an example of the SAD_SUBMITAPPL_REQ message that the SAD_SUBMITAPPL service operation receives from an online application:

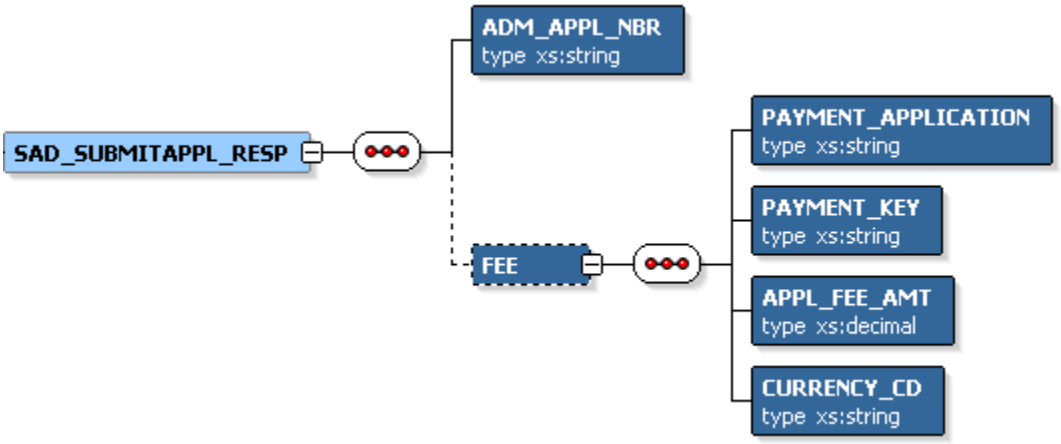
```
<?xml version="1.0"?>
<SAD_SUBMITAPPL_REQ>
  <ADM_APPL_DATA>
   <!-- Application data shape -->
  </ADM_APPL_DATA>
  <CONSTITUENT>
   <!-- Constituent data shape -->
  </CONSTITUENT>
</SAD_SUBMITAPPL_REQ>
```

Output Message: SAD_SUBMITAPPL_RESP

When the Integration Broker receives the SAD_SUBMITAPPL_REQ message, it responds with the SAD_SUBMITAPPL_RESP message.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The following diagram shows the output parameters that the SAD_SUBMITAPPL service operation passes to the online application:



SAD_SUBMITAPPL_RESP Message Parameters

The following is an example of the SAD_SUBMITAPPL_RESP message that the SAD_SUBMITAPPL service operation transmits to the online application:

```
<?xml version="1.0"?>
<SAD_SUBMITAPPL_RESP>
  <ADM_APPL_NBR>00023456</ADM_APPL_NBR>
  <FEE>
    <PAYMENT_APPLICATION>SADAAWS</PAYMENT_APPLICATION>
    <PAYMENT_KEY>00023456</PAYMENT_KEY>
    <APPL_FEE_AMT>20</APPL_FEE_AMT>
    <CURRENCY_CD>USD</CURRENCY_CD>
  </FEE>
</SAD_SUBMITAPPL_RESP>
```

Fault Message: SAD_FAULT_RESP

Refer to the SAD_FAULT_RESP message example in the SAD_CREATEAPPL section.

SAD_GETAPPLS

This service operation enables the user to retrieve a summary of all active admission applications from the Campus Solutions system for the current authenticated user. The applications summary includes applications found in both the admissions staging and production database tables.

Details:

Description:	Use this service operation to retrieve a summary of all the applications that reside in the Campus Solutions staging and production tables. The service operation retrieves only those applications that the authenticated user is authorized to access. SAD_GETAPPLS can operate under applicant or administrative modes. When using administrator mode, the application data returned from the

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	Admissions production tables is subject to application center row level security. When using applicant mode, the application data returned is restricted to those applications entered previously by the authenticated online application user.
Users:	• An applicant using an online application. • An administrator using an online application to retrieve data that he or she previously saved or submitted. The administrator can retrieve an application for an applicant who has or does not have an EMPLID.

Processing	This service operation performs the following steps:
	
	1. Validate that the user is authenticated (and is not a *Guest*).
	2. If administrator mode then:
	  1. Validate the required input administrator parameters.
	  2. Validate that the administrator is authorized to access the application center.
	  3. Use the EMPLID supplied by the administrator for the transaction.
	3. Determine the list of configured application transactions from the Transaction Setup component.
	4. Retrieve all the application transactions for all the configured application transactions.
	5. Retrieve all admission applications from the Admissions staging tables that the authenticated user is authorized to access.
	6. If applicant mode, then retrieve all the applications associated with the authenticated EMPLID from the production tables.
	7. To retrieve the application based on the ADM_APPL_NBR key:
	  1. First, check whether the application exists in the admissions staging tables using the SCC_TEMP_ID key.
	  2. If the application does not exist in the staging tables, check the admissions production tables using the EMPLID key.
	8. Return the Application Summary, which indicates ID (EMPLID or SCC_TEMP_ID), Application Number, Institution, Academic Career and Application Status (SAVED, SUBMITTED, ERROR, or POSTED).

Output:	After processing the steps, a summary of all active applications is returned.
Error Conditions:	The following conditions result in a service error: • User is not authenticated (for example, if an anonymous *Guest* user tries to sign into the online application, the service operation throws an error message). • Required input parameters have not been supplied by the user (see the graphic titled *SAD_GETAPPLS_REQ Message Parameters*).

Input Message: SAD_GETAPPLS_REQ

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The following diagram shows the input parameters that the SAD_GETAPPLS service operation receives from an online application:

SAD_GETAPPLS_REQ Message Parameters

Required data that the online application must supply to the service operation in *Non-administrator (Applicant) mode*: No required parameters

Required data that the online application must supply to the service operation in *Administrator* mode:

- Institution
- Admission Application Center
- Academic Program

The following is an example of the SAD_GETAPPLS_REQ message that the SAD_GETAPPLS service operation receives from an online application:

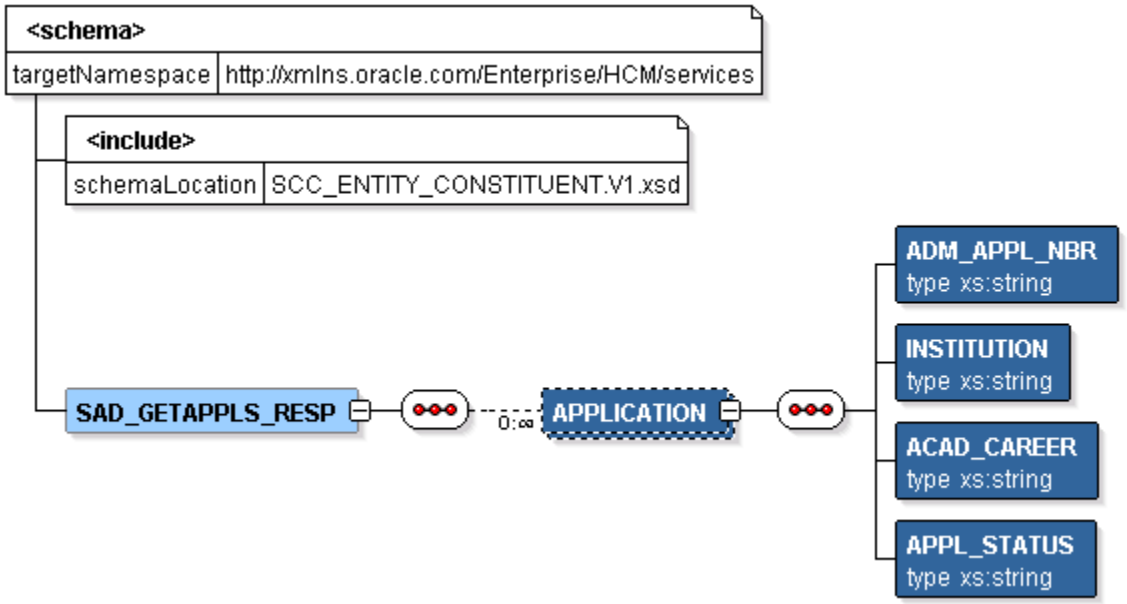
```
<?xml version="1.0"?>
<SAD_GETAPPLS_REQ>
</SAD_GETAPPLS_REQ>
```

Output Message: SAD_GETAPPLS_RESP

When the Integration Broker receives the SAD_GETAPPLS_REQ message, it responds with the SAD_GETAPPLS_RESP message.

The following diagram shows the output parameters that the SAD_GETAPPLS service operation passes to the online application:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



SAD_GETAPPLS_RESP Message Parameters

The following is an example of the SAD_GETAPPLS_RESP message that the SAD_GETAPPLS service operation transmits to the online application:

```
<?xml version="1.0"?>
<SAD_GETAPPLS_RESP>
  <APPLICATION>
    <ADM_APPL_NBR>00024656</ADM_APPL_NBR>
    <INSTITUTION>PSUNV</INSTITUTION>
    <ACAD_CAREER>UGRD</ACAD_CAREER>
    <APPL_STATUS>DRAFT</APPL_STATUS>
        </APPLICATION>
  <APPLICATION>
    <ADM_APPL_NBR>00024657</ADM_APPL_NBR>
    <INSTITUTION>PSUNV</INSTITUTION>
    <ACAD_CAREER>UGRD</ACAD_CAREER>
    <APPL_STATUS>SUBMITTED</APPL_STATUS>
        </APPLICATION>
</SAD_GETAPPLS_RESP>
```

Fault Message: SAD_FAULT_RESP

Refer to the SAD_FAULT_RESP message example in the SAD_CREATEAPPL section.

SAD_GETAPPL

This service operation enables the user to retrieve a specific admissions application from the Campus Solutions system. The application may exist in the admissions staging tables or in the admissions production tables. Saved applications reside in the staging tables and submitted applications reside in the production tables.

Details:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Description:	Use this service operation to retrieve a saved or submitted application using the ADM_APPL_NBR key. An online application can obtain the ADM_APPL_NBR application key by calling the SAD_GETAPPLS service, which retrieves a summary of all the applications associated with the current authenticated user. The SAD_GETAPPL operation can operate under applicant or administrative modes. When using applicant mode, the application data returned is restricted to those applications entered previously by the authenticated online application user. When using administrator mode, the application data returned from the admissions production tables is subject to application center row level security.
Users:	• An applicant using an online application. • An administrator using an online application to retrieve data that they previously saved or submitted. The administrator can retrieve an application for an applicant who has or does not have an EMPLID.

Processing:	This service operation performs the following steps:
	
	1. Validate that the user is authenticated (and is not a *Guest*).
	2. If administrator mode, then:
	  1. Validate the required input administrator parameters.
	  2. Validate that the administrator is authorized to access the application center.
	  3. Use the EMPLID supplied by the administrator for the transaction.
	3. Validate that the applicant is authorized to retrieve the application (for example, verify whether the applicant is the same person who created the application).
	4. If administrator mode, then use the EMPLID input parameter to retrieve the application.
	5. If applicant mode, then use the EMPLID input parameter associated with the authenticated user profile to retrieve the application.
	6. To retrieve the application based on the ADM_APPL_NBR key:
	  1. First, check whether the application exists in the admissions staging tables using the SCC_TEMP_ID key.
	  2. If the application does not exist in the staging tables, check the admissions production tables using the EMPLID key.
	7. Return the application data, if found.

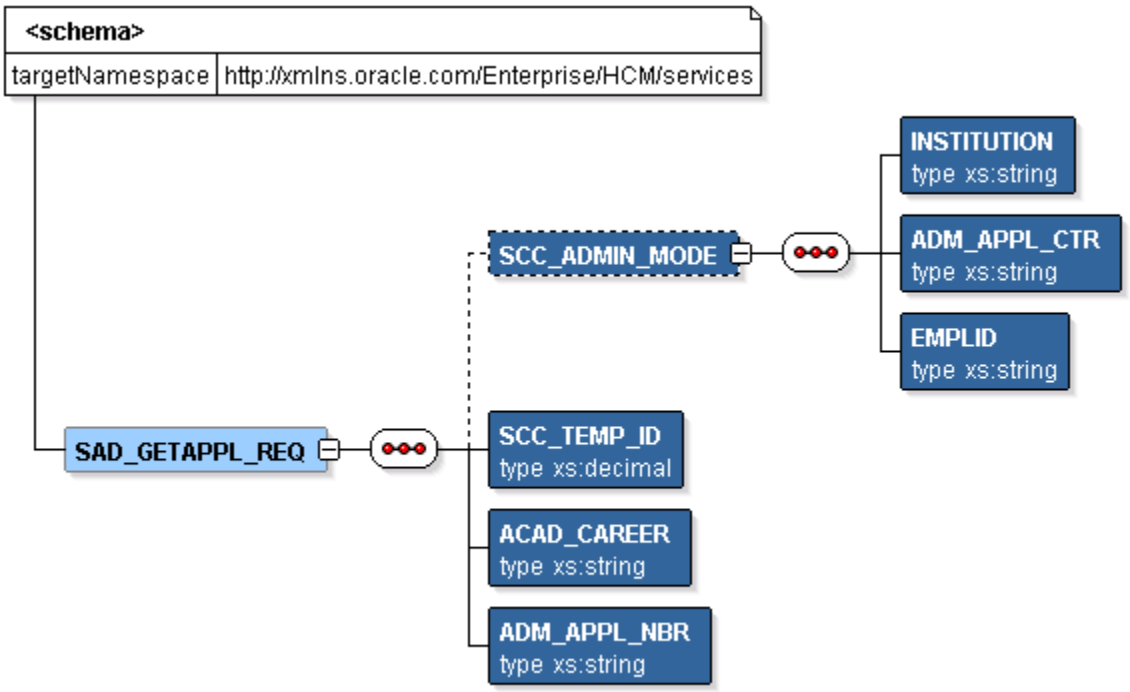
Output:	After completing the processing steps, the service operation returns a single application if it exists in the staging or production tables. All the application details along with constituent details are returned, if the application exists.
Error Conditions:	The following conditions result in a service error:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	• User is not authenticated (for example, if an anonymous *Guest* user tries to sign into the online application, the service operation throws an error message). • User is not authorized to access the application (for example, the applicant is trying to access an application created by another user). • Required input parameters have not been supplied by the user (see the graphic titled *SAD_GETAPPL_REQ Message Parameters*). • Administrative user is not authorized to access the application due to the application center security configuration.

Input Message: SAD_GETAPPL_REQ

The following diagram shows the input parameters that the SAD_GETAPPL service operation receives from an online application:



SAD_GETAPPL_REQ Message Parameters

Required data that the online application must supply to the service operation in *Non-administrator (Applicant) mode*:

- Academic Career
- Application Number

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Required data that the online application must supply to the service operation in *Administrator mode*:

- Institution
- Admission Application Center

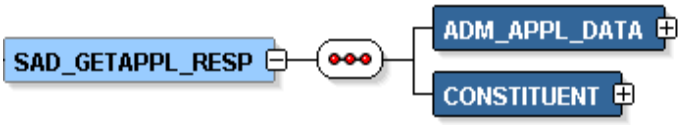
The following is an example of the SAD_GETAPPL_REQ message that the SAD_GETAPPL service operation receives from an online application:

```
<?xml version="1.0"?>
<SAD_GETAPPL_REQ>
  <ADM_APPL_NBR>00024656</ADM_APPL_NBR>
  <ACAD_CAREER>UGRD</ACAD_CAREER>
</SAD_GETAPPL_REQ>
```

Output Message: SAD_GETAPPL_RESP

When the Integration Broker receives the SAD_GETAPPL_REQ message, it responds with the SAD_GETAPPL_RESP message.

The following diagram shows the output parameters that the SAD_GETAPPL service operation passes to the online application:



SAD_GETAPPL_RESP Message Parameters

Note that the SAD_GETAPPL_RESP schema includes the Application entity (ADM_APPL_DATA) and Constituent entity XSD schemas.

The following is an example of the SAD_GETAPPL_RESP message that the SAD_GETAPPL service operation transmits to the online application:

```
<?xml version="1.0"?>
<SAD_GETAPPL_RESP>
  <ADM_APPL_DATA>
   <!-- Application data shape -->
  </ADM_APPL_DATA>
  <CONSTITUENT>
   <!-- Constituent data shape -->
  </CONSTITUENT>
</SAD_GETAPPL_RESP>
```

Fault Message: SAD_FAULT_RESP

Refer to the SAD_FAULT_RESP message example in the SAD_CREATEAPPL section.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SAD_GETATTACH

This service operation enables the user to retrieve the byte content of an admissions application attachment from the Campus Solutions system.

Details:

Description:	Use this service operation to retrieve an attachment associated with the admission application. The online application must encode the attachment byte content in Base64 format when a user saves or submits the application data through AAWS. Subsequently, the online application can use SAD_GETATTACH to retrieve the attachment. After retrieval, the online application must decode the attachment. The applicant or administrator can then use the appropriate file type viewer to view the attachment. The attachment is stored within the Campus Solutions attachment framework in a database table. You can use the Application Transactions or Maintain Applications components to access the application attachments.
Users:	• An applicant using an online application. • An administrator using an online application to retrieve data that he or she previously saved or submitted. The administrator can retrieve an application (and the associated attachments) for an applicant who has or does not have an EMPLID.

Processing:	This service operation performs the following steps:
	
	1. Validate that the user is authenticated (and is not a *Guest*).
	2. If administrator mode, then:
	  1. Validate the required input administrator parameters.
	  2. Validate that the administrator is authorized to access the application center.
	  3. Use the EMPLID supplied by the administrator for the transaction.
	3. Verify whether the applicant is authorized to retrieve the attachment (for example, verify whether the applicant is the same person who originally created the attachment through AAWS).
	4. If administrator mode, then use the EMPLID input parameter to retrieve the attachment.
	5. If applicant mode, then use the EMPLID input parameter associated with the authenticated User Profile to retrieve the attachment.
	6. To retrieve the attachment:
	  1. First, check whether the attachment exists in the staging tables using the SCC_TEMP_ID key.
	  2. If the attachment does not exist in the staging tables, check the production tables using the EMPLID key.
	7. Return the attachment, if found.

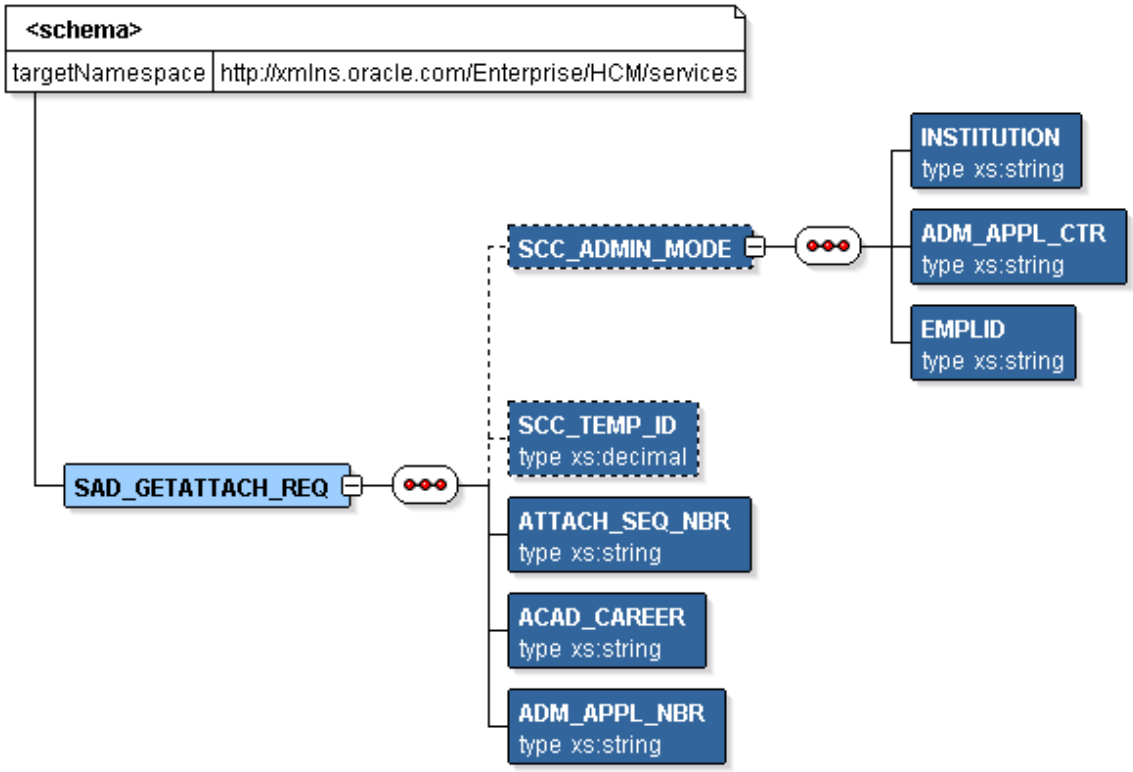
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Output:	After completing the processing steps, if the attachment exists, the service operation returns the attachment data to the online application.
Error Conditions:	The following conditions result in a service error: • User is not authenticated (for example, if an anonymous *Guest* user tries to sign into the online application, the service operation throws an error message). • User is not authorized to retrieve the attachment (for example, the applicant is trying to access an attachment created by another user). • Required input parameters have not been supplied by the user (see the graphic titled *SAD_GETATTACH_REQ Message Parameters*).Administrative user is not authorized to access the attachment due to application center configuration.

Note: The Online Application may issue multiple calls to the SAD_GETATTACH service operation in order to retrieve multiple Application attachments.

Input Message: SAD_GETATTACH_REQ

The following diagram shows the input parameters that the SAD_GETATTACH service operation receives from an online application:



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SAD_GETATTACH_REQ Message Parameters

Required data that the online application must supply to the service operation in *Non-administrator (Applicant) mode*:

- Attachment Sequence Number
- Academic Career
- Application Number

Required data that the online application must supply to the service operation in *Administrator mode*:

- Institution
- Admission Application Center

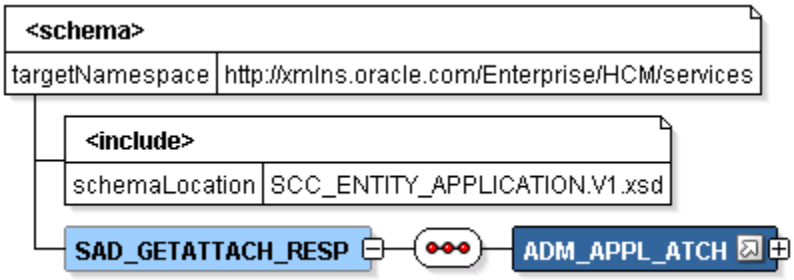
The following is an example of the SAD_GETATTACH_REQ message that the SAD_GETATTACH service operation receives from an online application:

```
<?xml version="1.0"?>
<SAD_GETATTACH_REQ>
  <ACAD_CAREER>UGRD</ACAD_CAREER>
  <ADM_APPL_NBR>00024501</ADM_APPL_NBR>
  <ATTACH_SEQ_NBR>1</ATTACH_SEQ_NBR>
</SAD_GETATTACH_REQ>
```

Output Message: SAD_GETATTACH_RESP

When the Integration Broker receives the SAD_GETATTACH_REQ message, it responds with the SAD_GETATTACH_RESP message.

The following diagram shows the output parameter that the SAD_GETATTACH service operation passes to the online application:



SAD_GETATTACH_RESP Message Parameters

The following is an example of the SAD_GETATTACH_RESP message that the SAD_GETATTACH service operation transmits to the online application:

```
<?xml version="1.0"?>
<SAD_GETATTACH_RESP>
  <ADM_APPL_ATCH class="R">
    <EMPLID>2</EMPLID>
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

```
    <ACAD_CAREER>UGRD</ACAD_CAREER>
    <STDNT_CAR_NBR>0</STDNT_CAR_NBR>
    <ADM_APPL_NBR>00024501</ADM_APPL_NBR>
    <ATTACH_SEQ_NBR>1</ATTACH_SEQ_NBR>
    <DESCR254>test file</DESCR254>
    <SCC_ROW_ADD_OPRID/>
    <SCC_ROW_ADD_DTTM/>
    <SCC_ROW_UPD_OPRID/>
    <SCC_ROW_UPD_DTTM/>
    <ATTACHSYSFILENAME>ADM_APPL_ATCH_20090331161239_Who
Columns.doc</ATTACHSYSFILENAME>
    <ATTACHUSERFILE>Who Columns.doc</ATTACHUSERFILE>
    <AV_ATCH_TYPE/>
<BASE64FILE>0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAA  </BASE64FILE>
  </ADM_APPL_ATCH>
  <CONSTITUENT>
   <!-- Constituent data shape -->
  </CONSTITUENT>
</SAD_GETATTACH_RESP>
```


Fault Message: SAD_FAULT_RESP

Refer to the SAD_FAULT_RESP message example in the SAD_CREATEAPPL section.

Payment Integration Web Service Operations

This section discusses the SSF_INITIATE_EPAYMENT and SSF_COMPLETE_EPAYMENT service operations for the SSF_EPAYMENT_TRANSACTION service.

Note: The payment integration service operations are delivered for only supporting online applications. These service operations are not intended to be used for any other purpose.

Note: The schema diagrams in this section may get updated with future bundle releases.

SSF_INITIATE_EPAYMENT

This service operation enables the user to record a newly initiated electronic payment in Student Financials and provides access to Campus Solutions' hosted payment framework. The hosted payment framework provides information necessary to transition the user to a designated third party payment provider's hosted site to make a payment. The insert into the GL_TXN table, for the general ledger process, will not be done until the Complete ePayment service operation (SSF_COMPLETE_EPAYMENT) is invoked successfully.

This web service operation is adapter-based. AAWS has been delivered with its own custom Payment Application adapter and will perform its own validation according to the admissions application functional requirements.

Note that the Initiate ePayment service operation operates only within the Self Service mode (and cannot be used within the Administrator mode).

Details:

Description:	The Initiate ePayment service operation call will create a new SF Payment in Student Financials in *Initiated* status, assigning a new SF

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	Payment Reference Number (SF_PMT_REF_NBR) for tracking. This service uses the payment application's own tracking number as the merchant reference number passed to the third party payment processor. For AAWS, that tracking number is the application number (ADM_APPL_NBR). The application number will be stored in the SF Payment as ADM_APPL_NBR and SSF_EXT_TRANS_REF and the SF Payment Reference number will be stored in the Admissions Tender Staging table. In addition, the Temp ID will be stored in the SF Payment and Log. The service, through the AAWS adapter, will create new Admissions Tender Staging records (if they do not exist) for all tenders associated with the application center used in the creation of the application. The Initiate ePayment service will perform initial validation. For instance, the payment method must be either credit card or electronic check (CC or EC), the amount has to be greater than zero, and the Payment Application must be defined in the system table SSF_ELEC_PMTS. When the AAWS adapter is invoked, it will perform further validation on the transaction. • The payment amount and currency must match the previously calculated application fee stored in the Admissions Application Staging record (SAD_APL_DAT_STG). This amount should have been previously calculated in and returned from the submit web service operation. • The transaction must have been originated by the owner of the application. • The application must be in Saved (*SV*) status. • If the payment is for a credit card tender type, the SF Merchant must be defined in the application center associated with the application. For electronic check transactions, the Electronic Check Merchant ID must be defined in the Application Center. These SF Merchants must be associated with Payment Merchants that are defined as *hosted*. • The tender used must be defined as a valid tender on the Application Fee code associated with the application center.
Users:	• An applicant using an online application.

Processing:	This service operation performs the following steps:
	
	1. Validate the parameters passed in to the web service.
	2. Create a SF Payment to record the transaction. The SF Payment will be in *Initiated* status. For AAWS, SF Payment Detail records, which will later be used to record the transaction in the General Ledger, will be based on the Application Fee Item Type setup.
	3. Call the AAWS adapter to:
	  • Validate the application
	  • Validate the payment parameters
	  • Create Application Tender Staging records
	  • Save the application

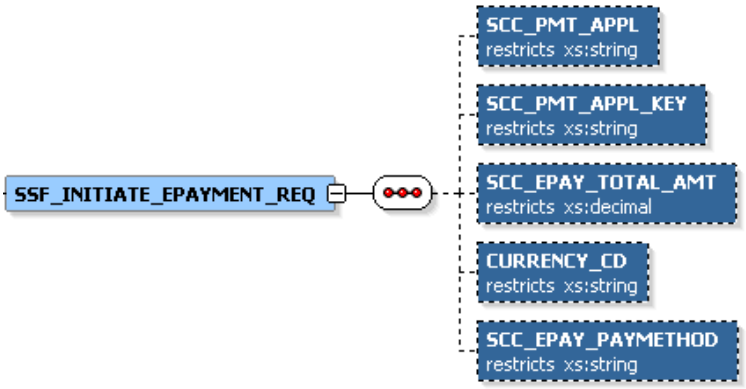
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	4. Call the hosted payment framework. The hosted payment framework will: • Based on the hosted payment adapter associated with the Payment Merchant, perform an initial web service callout to the third party to pass demographic information about the applicant, such as name, address, email and phone. Note that this information may or may not be used, depending on the third party payment processor. In addition, a shared secret may be passed to the third party to authenticate electronic check transactions. For AAWS, this shared secret is the Constituent birth date. The third party will return a temporary token that identifies the transaction to both the third party and Campus Solutions. • Using the HTML object definition on the Payment Merchant, create a string that will be used by the Online Application to either http post or redirect to the third party.
Output:	After completing the processing steps, the service operation: 1. Records the transaction in the SF Payment table and Detail records. 2. Creates new Admissions Application Tender Staging records. 3. Optionally, performs a web service callout to the third party payment processor to pass demographic information and receive a token identifying the transaction.
Error Conditions:	The following conditions result in a service error: • The payment method is invalid. • The payment amount is zero. • The Payment Application is invalid. • Required input parameters have not been supplied by the user • The transaction does not pass the AAWS adapter validation.

Input Message: SSF_INITIATE_EPAYMENT_REQ

The following diagram shows the input parameters that the SSF_INITIATE_EPAYMENT service operation receives from an online application:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



SSF_INITIATE_EPAYMENT_REQ Message Parameters

Required data that the online application must supply to the service operation:

- Payment Application
- Payment Application Key (Application Number)
- Payment Amount
- Currency Code
- Payment Method (Credit card or electronic check)

The following is an example of the SSF_INITIATE_EPAYMENT_REQ message that the SSF_INITIATE_EPAYMENT service operation receives from an online application:

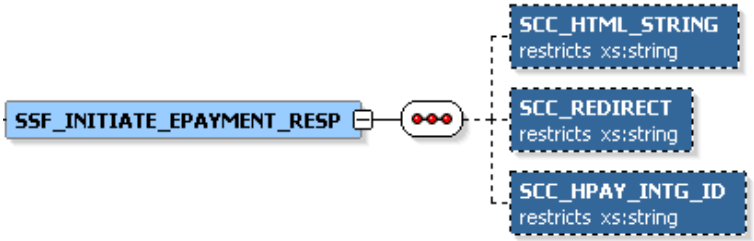
```
<?xml version="1.0"?>
<SSF_INITIATE_EPAYMENT_REQ xmlns="http://xmlns.oracle.com/Enterprise/HCM/services">
  <SCC_PMT_APPL>SADAAWS</SCC_PMT_APPL>
  <SCC_PMT_APPL_KEY>00024828</SCC_PMT_APPL_KEY>
  <SCC_EPAY_TOTAL_AMT>20</SCC_EPAY_TOTAL_AMT>
  <CURRENCY_CD>USD</CURRENCY_CD>
  <SCC_EPAY_PAYMETHOD>CC</SCC_EPAY_PAYMETHOD>
</SSF_INITIATE_EPAYMENT_REQ>
```

Output Message: SSF_INITIATE_EPAYMENT_RESP

When the Integration Broker receives the SSF_INITIATE_EPAYMENT_REQ message, it responds with the SSF_INITIATE_EPAYMENT_RESP message.

The following diagram shows the output parameter that the SSF_INITIATE_EPAYMENT service operation passes to the online application:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.



SSF_INITIATE_EPAYMENT_RESP Message Parameters

The following is an example of the SSF_INITIATE_EPAYMENT_RESP message that the SSF_INITIATE_EPAYMENT service operation transmits to the online application:

[[Bundle 27: The example message has been updated]]

```
<?xml version="1.0"?>
<SSF_INITIATE_EPAYMENT_RESP xmlns="http://xmlns.oracle.com/Enterprise/HCM/services">
  <SCC_REDIRECT>N</SCC_REDIRECT>
  <SCC_HTML_STRING>
    <![CDATA*<SCRIPT LANGUAGE='JavaScript'>
writeConsole('https://someserver/C30002test_upay/web/index.jsp', '<input type="HIDDEN"
name="TICKET_NAME" value="SADAAWS"><input type="HIDDEN" name="UPAY_SITE_ID"
value="12"><input type="HIDDEN" name="TICKET" value="ce224625-e55c-4346-b87c-
96f784f5ae90">');
function writeConsole(bind1, bind2)
{
var stuff = '<form name="payform" action="'+bind1+
'" method="POST">'+bind2+'<SCRIPT LANGUAGE="JavaScript" onload="submitForm()">'+
'function submitForm(){document.payform.submit()}'+
'</'+'SCRIPT>'+'</'+'form'+'>';
document.body.innerHTML = stuff;
document.payform.submit();
}
</SCRIPT>*]>
  </SCC_HTML_STRING>
  <SCC_HPAY_INTG_ID>QA_CS_AAWS_MODEL1A</SCC_HPAY_INTG_ID>
</SSF_INITIATE_EPAYMENT_RESP>
```

Fault Message: SSF_TRANS_FAULT_RESP

If the service operation encounters an error condition, it responds with the SSF_TRANS_FAULT_RESP message.

The following is an example of the SSF_TRANS_FAULT_RESP message that the SSF_INITIATE_EPAYMENT service operation transmits to the online application:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>Client</faultcode>
      <faultstring>An Error occurred processing this request (14098,286)</faultstring>
      <detail>
        <MSGS>
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

```
          <MSG>
            <ID>14871-11</ID>
            <DESCR>The Payment Amount must be greater than zero. (14871,11)</DESCR>
          </MSG>
        </MSGS>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SSF_COMPLETE_EPAYMENT

This web service operation completes a previously initiated (SSF_INITIATE_EPAYMENT) SF Payment, by authorizing the transaction and submitting the application. If authorization is successful, the SF Payment Transaction Manager will create the Direct GL Transaction.

[[Bundle 27: The following paragraph has been updated.]]

After having successfully invoked the SSF_INITIATE_EPAYMENT service operation, the Online Application should transition the applicant to the third party payment provider using the SCC_HTML_STRING returned from the SSF_INITIATE_EPAYMENT response. This next web service should be invoked after the transaction is submitted by the applicant from the third party payment processor user interface.

Details:

Description:	The Complete ePayment web service operation interprets the response from the third party payment processor by invoking the hosted payment framework. The data received from the third party is transformed into Campus Solutions data elements to allow the update of both Student Financials tables and the calling Payment Application (AAWS). If the user submitted the transaction successfully from the third party processor, this web service operation will complete any processing required in Student Financials to record the payment. The session ID, which is a unique token generated and returned by the third party to identify the payment, as well as the last 4 digits of the credit card or account number, will be stored in the SF Payment table from within the hosted payment framework. The ePayment API will be invoked to authorize the transaction. If successful, the payment will be processed in Student Financials by setting the SF Payment status to Complete (SF_PAYMENT_STATUS = *B*), creating the SF Payment Log with the results of the authorization, and eventually deleting the SF Payment. The transaction will be recorded directly in the General Ledger using the Direct to GL API (as opposed to posting to a Student Financials account). Finally, the adapter for AAWS will be invoked to submit the application.

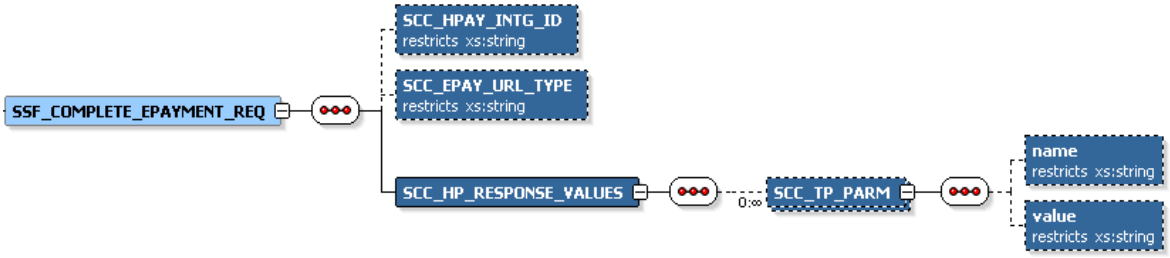
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

	The application fee status will be set to *Received*, the Application Fee Paid amount and GL Sent Amount will be set to the Payment Amount. The Admissions Application Tender Staging record's tender amount will also be set to the Payment Amount. If the user cancelled the transaction from the third party payment processor's hosted site, the SF Payment status will be set to Cancelled. No update will occur on the AAWS adapter and the applicant can submit another payment against the application.
Users:	• An applicant using an online application.
Processing:	This service operation performs the following steps: 1. Validate the parameters passed in to the web service. 2. Call the Hosted Payment framework using the Hosted Payment adapter passed into the web service. The hosted payment framework will interpret the name/value pairs and transform them into Campus Solutions data elements using the hosted payment adapter settings. It will extract the Payment Application ID and Key (application number) to correctly identify the transaction. It will update the session id and last 4 digits of the account or credit card number in the SF Payment. 3. Complete the SF Payment associated with the application. For successfully submitted transactions, the transaction will be authorized and sent to the General Ledger. 4. Call the AAWS adapter to update the proper fields in the application and submit it.
Output:	After completing the processing steps, the service operation: 1. Sets the SF Payment status to *Complete* or *Cancelled* according to the user action on the third party processor. It will set the status to *Declined* if authorization fails. The SF Payment will be deleted and the SF Payment Log will be created to record the transaction results. 2. Creates the General Ledger Transaction records for later processing to the General Ledger. 3. Submits the application if the payment was successfully authorized.
Error Conditions:	The following conditions result in a service error: • The Hosted Payment Adapter ID is invalid. • The Return URL type is invalid (Valid values are S, C or E) • The Payment Application is invalid. • Required input parameters have not been supplied by user. • The transaction does not pass the AAWS adapter validation.

Input Message: SSF_COMPLETE_EPAYMENT_REQ

The following diagram shows the input parameters that the SSF_COMPLETE_EPAYMENT service operation receives from an online application:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

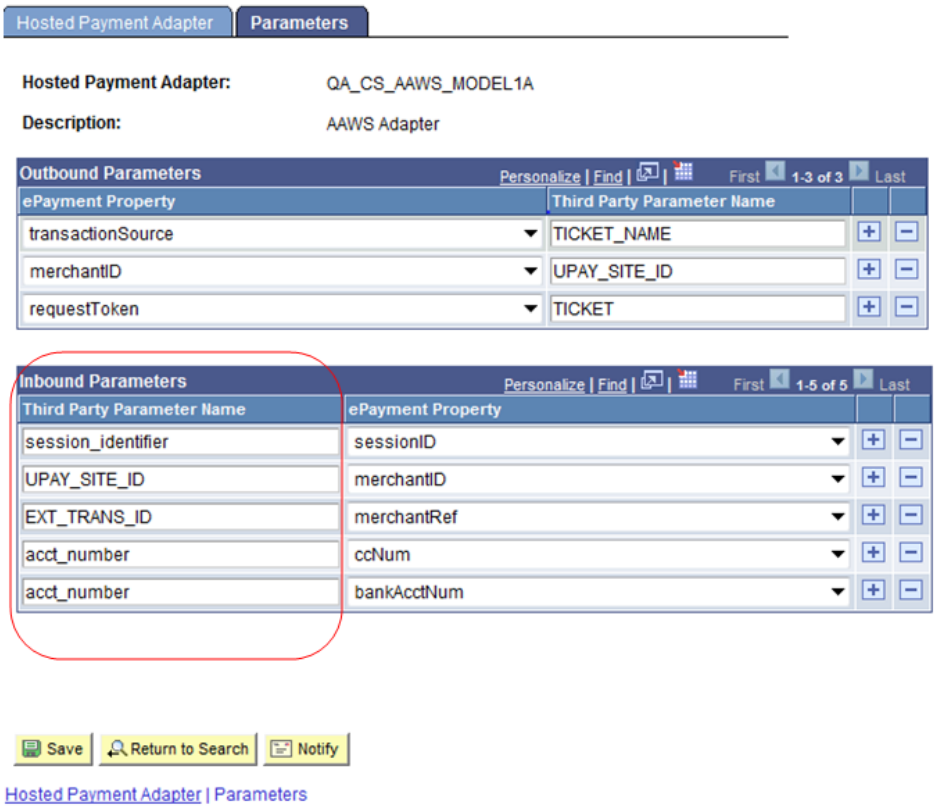


SSF_COMPLETE_EPAYMENT_REQ Message Parameters

[[Bundle 27: Added information about the Hosted Payment Adapter Setting, including page example.]]

Required data that the online application must supply to the service operation:

- Hosted Payment Adapter ID
- Return URL Type (Success, Cancelled, Error)
- Hosted Payment Response Values (Name/Value pairs returned from third party). These values are expected to correspond to the values under the Third Party Parameter Name of Inbound Parameter grid of Hosted Payment Adapter Setting. For example:



The following is an example of the SSF_COMPLETE_EPAYMENT_REQ message that the SSF_COMPLETE_EPAYMENT service operation receives from an online application:

[[Bundle 27: The example message has been updated]]

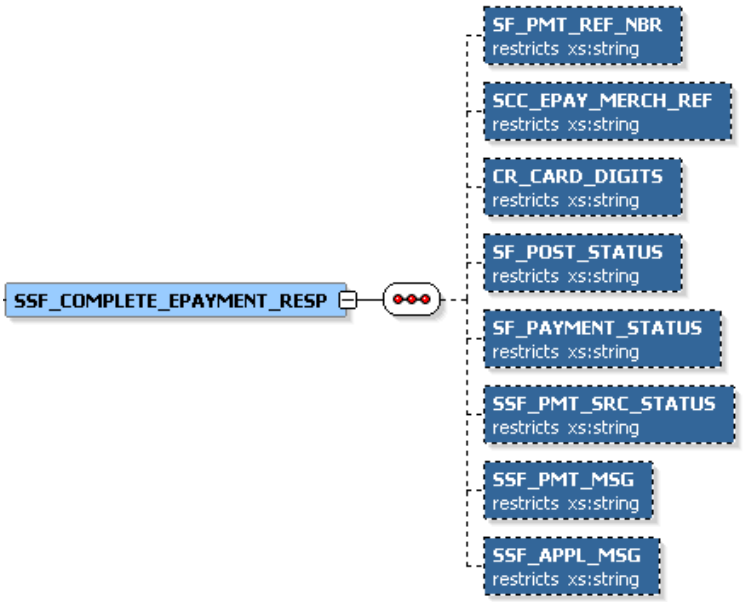
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

```
<?xml version="1.0"?>
 <SSF_COMPLETE_EPAYMENT_REQ
 xmlns="http://xmlns.oracle.com/Enterprise/HCM/services">
   <SCC_HPAY_INTG_ID>QA_CS_AAWS_MODEL1A</SCC_HPAY_INTG_ID>
   <SCC_EPAY_URL_TYPE>S</SCC_EPAY_URL_TYPE>
   <SCC_HP_RESPONSE_VALUES>
     <SCC_TP_PARM>
       <name>session_identifier</name>
       <value>5c012fb4-43a4-4ffd-be63-a1415ce2b7e4</value>
     </SCC_TP_PARM>
     <SCC_TP_PARM>
       <name>UPAY_SITE_ID</name>
       <value>12</value>
     </SCC_TP_PARM>
     <SCC_TP_PARM>
       <name>EXT_TRANS_ID</name>
       <value>SADAAWS_00024863</value>
     </SCC_TP_PARM>
     <SCC_TP_PARM>
       <name>acct_number</name>
       <value>XXXXXXXXX2222</value>
     </SCC_TP_PARM>
     <SCC_TP_PARM>
       <name>acct_number</name>
       <value>XXXXXXXXX2222</value>
     </SCC_TP_PARM>
   </SCC_HP_RESPONSE_VALUES>
 </SSF_COMPLETE_EPAYMENT_REQ>
```


Output Message: SSF_COMPLETE_EPAYMENT_RESP

When the Integration Broker receives the SSF_COMPLETE_EPAYMENT_REQ message, it responds with the SSF_COMPLETE_EPAYMENT_RESP message.

The following diagram shows the output parameter that the SSF_COMPLETE_EPAYMENT service operation passes to the online application:



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

SSF_COMPLETE_EPAYMENT_RESP Message Parameters

The following is an example of the SSF_COMPLETE_EPAYMENT_RESP message that the SSF_COMPLETE_EPAYMENT service operation transmits to the online application:

```
<?xml version="1.0"?>
<SSF_COMPLETE_EPAYMENT_RESP
xmlns="http://xmlns.oracle.com/Enterprise/HCM/services">
  <SF_PMT_REF_NBR>000000000043</SF_PMT_REF_NBR>
  <SCC_EPAY_MERCH_REF>00024828</SCC_EPAY_MERCH_REF>
  <CR_CARD_DIGITS>1111</CR_CARD_DIGITS>
  <SF_POST_STATUS>S</SF_POST_STATUS>
  <SF_PAYMENT_STATUS>B</SF_PAYMENT_STATUS>
  <SSF_PMT_SOURCE_STATUS>S</SSF_PMT_SOURCE_STATUS>
  <SSF_PMT_MSG>Your payment was successfully processed.</SSF_PMT_MSG>
  <SSF_APPL_MSG>Your application 00024828 was successfully
submitted.</SSF_APPL_MSG>
</SSF_COMPLETE_EPAYMENT_RESP>
```

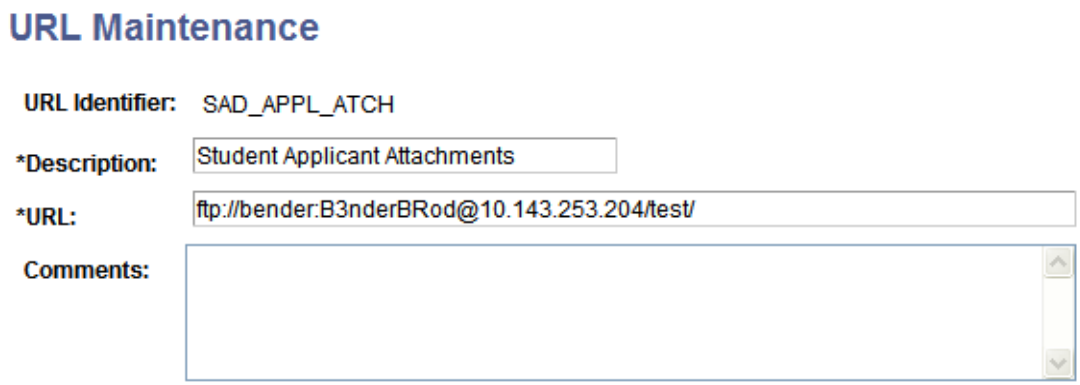
Fault Message: SSF_TRANS_FAULT_RESP

Refer to the SSF_TRANS_FAULT_RESP message example in the SSF_INITIATE_EPAYMENT section.

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:



Example of a URL Maintenance page

Setting Up Hosted Payment

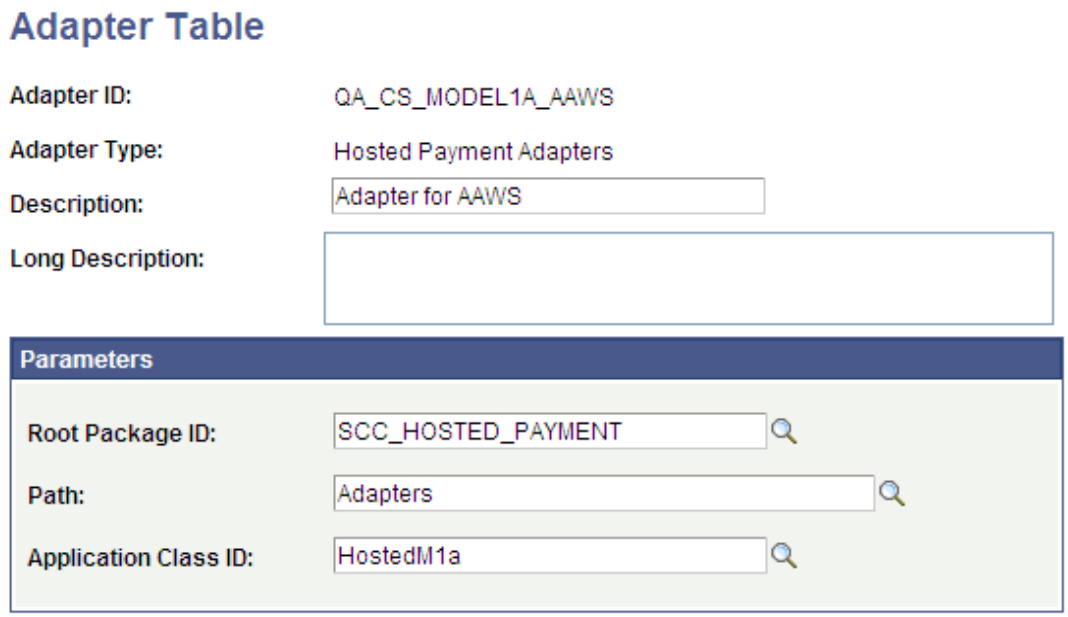
Application Fees are optionally required during application submission as configured in the Application Center Setup page. If fees are required to be paid then a prerequisite is to set up the hosted payment adapter and merchant which will facilitate the online payment.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Firstly, create a new HTML object in Application Designer. The HTML object is used to store user interface specific HTML and JavaScript which will direct the self-service user interface during the payment integration flow. Refer to the User Interface Considerations section in this chapter 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 in the first step) is attached to the adapter itself.

Access the Adapter Table page *(*Set Up SACR, System Administration, Utilities, Adapter Registry*).*



Hosted Payment Adapter Setup

Next, attach the new HTML definition to the Hosted Payment Adapter settings. Access the Hosted Payment Adapter page (Set Up SACR, Common Definitions, Electronic Payments, Hosted Payment Adapter Setting)

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Hosted Payment Adapter | Parameters

Hosted Payment Adapter: QA_CS_MODEL1A_AAWS

HTML Definition: QA_CS_HPAY_AAWS

Third Party URL: QA_CS_TP_URL

Record Name:

Field Name:

Customize | Find | View All | First 1 of 1 Last

Return URL Type	Weblib Function	
		Main Content

Hosted Payment Adapter page

The parameters page should be set up according to your third-party payment provider's requirements.

Hosted Payment Adapter | Parameters

Hosted Payment Adapter: QA_CS_MODEL1A_AAWS

Description: Token-based Integration

Outbound Parameters — Customize | Find | First 1-3 of 3 Last

ePayment Property	Third Party Parameter Name
transactionSource	TICKET_NAME
merchantID	UPAY_SITE_ID
requestToken	TICKET

Inbound Parameters — Customize | Find | First 1-5 of 5 Last

Third Party Parameter Name	ePayment Property
session_identifier	sessionID
UPAY_SITE_ID	merchantID
EXT_TRANS_ID	merchantRef
acct_number	ccNum
acct_number	bankAcctNum

Parameters page

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

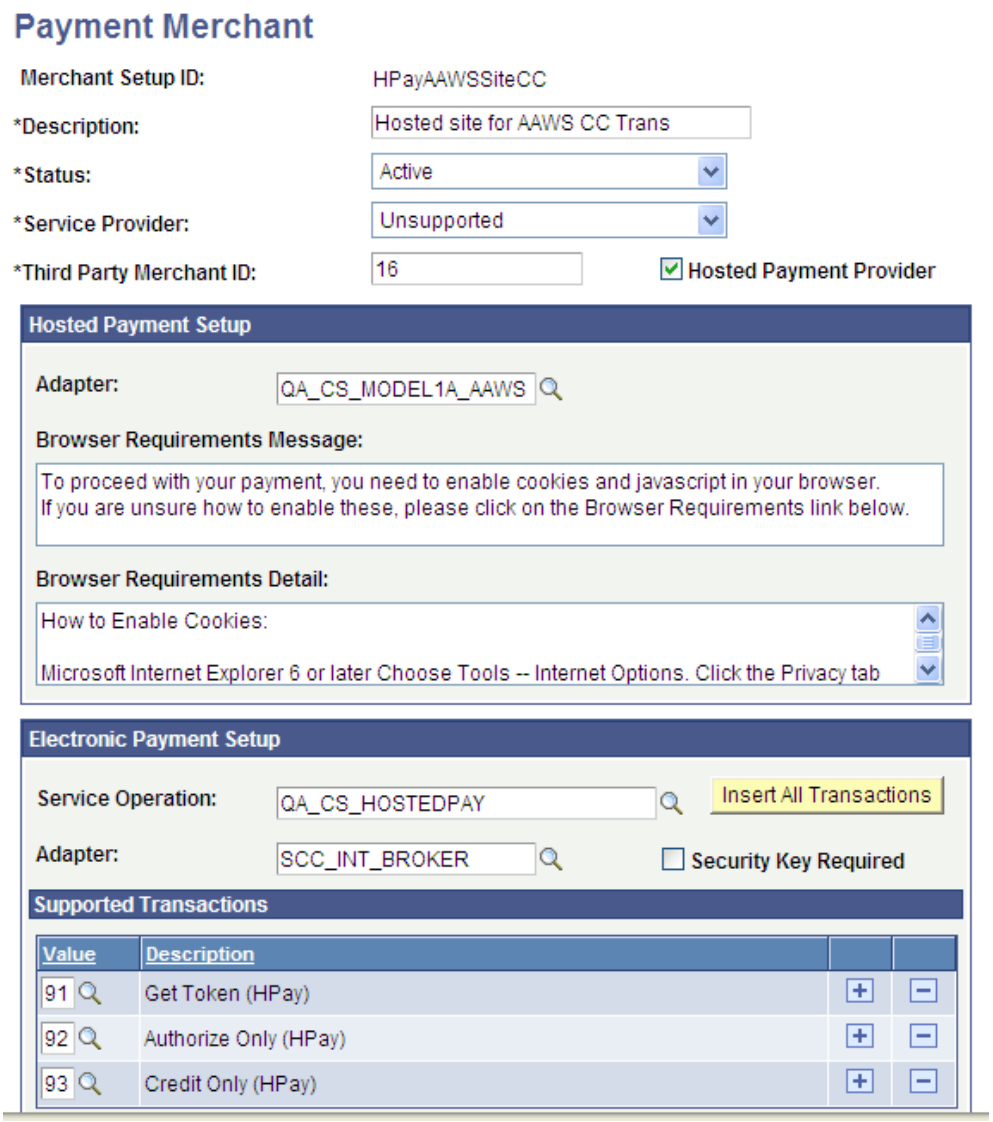
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

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).

Access the Payment Merchant page (Set Up SACR, Common Definitions, Electronic Payments, Payment Merchant).



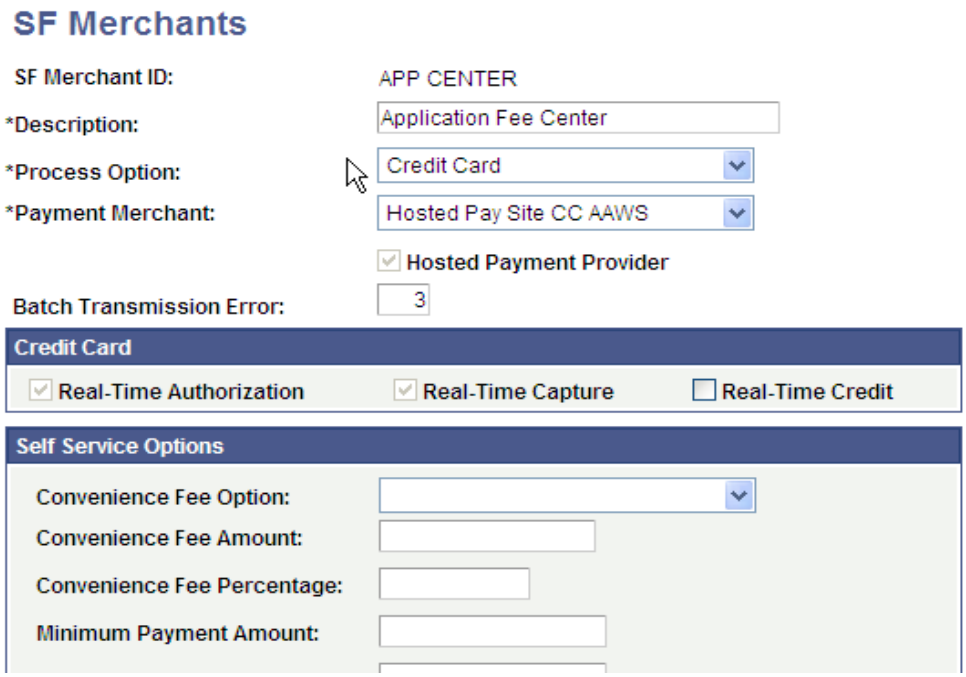
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Payment Merchant 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.

Access the SF Merchants page (*Set Up SACR, Common Definitions, Self Service, Student Financials, SF Merchants).*



SF Merchant page

Note: Convenience fees are not currently supported for online application fee payment.

Post Installation Troubleshooting Tips

This section discusses how to:

- Verify web service security settings.
- Publish the web services.
- Refresh the HCM Registry Cache.
- Verify Campus Solutions SOA framework setup entries.
- Avoid caching issues.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- Resolve Microsoft Internet Explorer 7 trusted sites issue.
- Resolve Adobe Flex domain security issue.

Verifying Web Service Security Settings

The service operations for the SAD_ADMISSIONS, SCC_USERREG, SCC_LOV, SSF_EPAYMENT_TRANSACTION, and SCC_SM_SERVICE services 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 as part of 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).
2. Click the Service Operation Security link to access the Web Service Access page**.**
3. On the Web Service Access page, verify whether the permission list has full access.

Publishing the Web Services

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

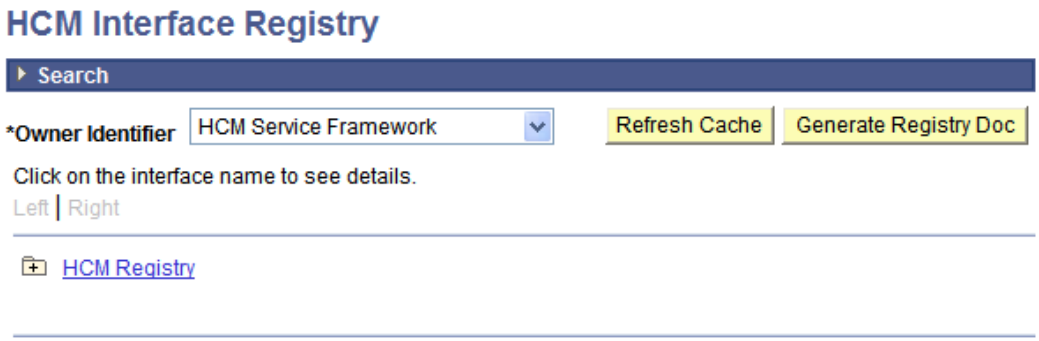
- SAD_ADMISSIONS
- SSF_EPAYMENT_TRANSACTION

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).



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

HCM Interface Registry page

2. Click the **Refresh Cache** button.

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:

Field	**Value**
Package Name	SCC_OLA
Path	Handlers
Application Class ID	Admissions

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

Field	**Value**
Package Name	SSF_EPAYMENT_TRANS
Path	HANDLERS
Application Class ID	Payments

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:

```
<!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:
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

```
.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>
```

Resolving Microsoft Internet Explorer 7 Trusted Sites Issue

When using Microsoft Internet Explorer 7, you may need to add your web server URL hostname to Microsoft Internet Explorer 7's trusted sites. If you do not add the URL to the trusted sites, you will receive a security error that starts like: *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.

Resolving Adobe Flex Domain Security Issue

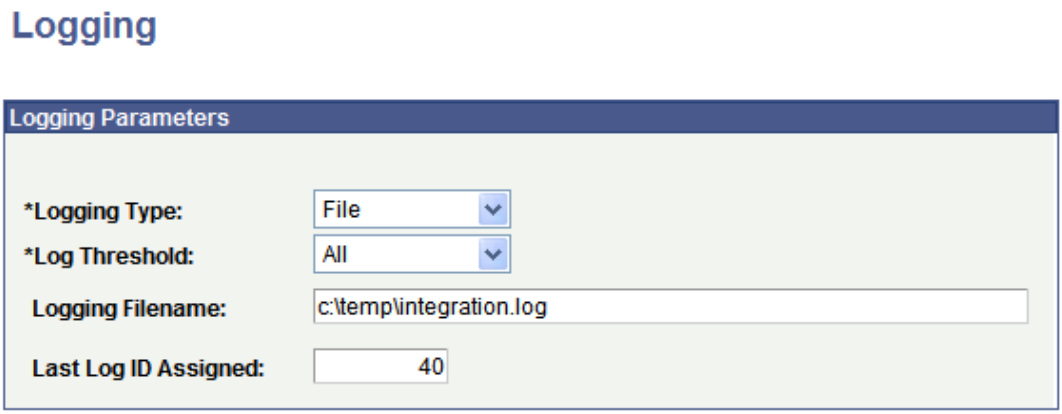
This issue is applicable if you are using Adobe Flex to develop an online application.

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.

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 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.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

User Interface Considerations

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

Note that in addition to user interfaces interacting with AAWS, a third party admission application can also interact with AAWS by transmitting the completed application data to AAWS for further processing.

Start: Unregistered User

Essentially all applicant users start in this state.

New User Registration

The user interface could use the SCC_USERREG_CREATEACCT service operation. NUR is a delivered CTM transaction.

Logon (Authentication)

The user interface could use the delivered SCC_USERREG_AUTHENTICATE service operation.

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 user interface 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.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A typical self-service, online-application user interface 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

Collecting application fees, while optional, is likely a common scenario that needs to be supported by the user interface. At some point after application creation (where the application center assigned to the application will be known to the user interface) but prior to application submission (when fees may be payable) the user interface 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 user interface. Options related to fee payment include whether:

- An application fee is required
- The applicant is given the option to *Pay Later*.
- The applicant can be given the option to request a *Waiver* for the fee.

Based on which fee options are configured, the user interface needs to react appropriately and present the applicant with only the valid options. This will in some cases alter the user interface page flow presented to the applicant.

A typical self-service, online-application user interface might perform the following actions:

- Collect the Academic Learning Area preferences that the applicant enters during the Create Application step. For example, the applicants are required to enter the academic career and program for which they wish to create an application.
- Use the preferences to create a request message for the SCC_LOV 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.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- 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:
    - If the *Pay Later* option is activated, then present the applicant with the choice to pay later (the applicant's choice is then used in the subsequent *Submit Application* step).
    - Store all the application fee and tendering configuration options in user interface storage for use in the subsequent *Present Fee Options* step.

Update Application

The online application's 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 user interface 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 SCC_GET_LOV service operation. 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 user interface 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.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- 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_ATTACH 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 user interface 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:
    - On a *Success* response there are various options:
        1. Present the user with a confirmation page. The confirmation page confirms that the application has been saved.
        2. Take the user back to the *Get Application Summary* page.
        3. Log out the user from the online application.
    - 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.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

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 user interface 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 user interface needs to cater for:
    - A *Success* response and fee payment *is not* required:
        1. Present the user with a confirmation page. The confirmation page confirms that the application has been submitted.
        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 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 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 or

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

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 (for example, CC and check) 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 user interface 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 service operation.
        - 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 operation.
        - Receive and process the response from the payment service operation.
        - 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 user interface 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 (for example, a database table).

          [[Bundle 27: Removed "cookie" from "for example" in previous sentence.]]
        - In the case of a HTML string response, the user interface can handle this in a number of ways. Some examples are:
            - Spawn a new child window to display the HTML string response.

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

        - Incorporate the HTML response into a parent window frame.

    3. The response style is configurable and customizable by use of dynamic HTML. There is sufficient flexibility for the user interface 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 user interface technology being employed.

- The end result of this activity is that the user will be presented with a browser page provided by the third 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 third party hosted payment provider (for example, the applicant will have been redirected to the third party site).

The applicant would then proceed to enter payment options and submit or cancel the payment. The third party provider would then redirect the applicant's browser to one of the 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 user interface should display the appropriate message to the applicant based on the Complete ePayment (SSF_COMPLETE_EPAYMENT) response message.

A typical self-service, online-application user interface 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:

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

- The user interface 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 user interface 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 user interface 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 user interface 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 Universally Unique Identifier (UUID) format, 36 character ISO standard.

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 user interface 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 user interface temporary storage area.

- Present the user with a *logged off* confirmation page.

Building the Hosted Payment HTML object

For more information about the hosted payment feature, refer to the *Electronic Payment Integration Developers Reference Guide*. This guide is available on My Oracle Support for PeopleSoft Enterprise Campus Solutions 9.0 Feature Pack 2.

The HTML object is provided as a flexible means to allow a diverse range of user interface 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 third 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 user interface 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 third party payment provider:

```
<BODY onload="submitForm()">
 <form name=”myform” action="https://someserver/upay/web/index.jsp"
method="post">
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

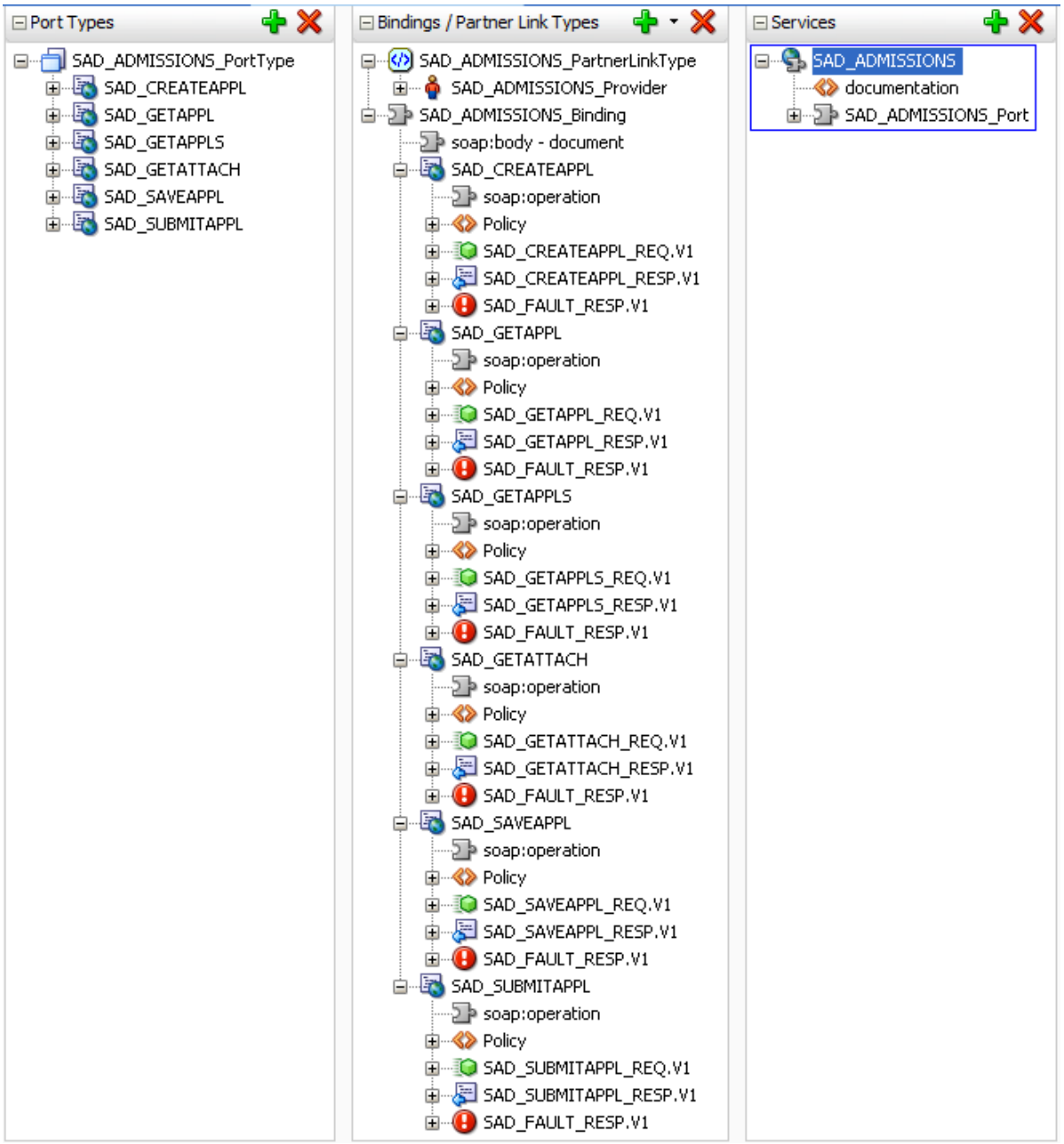
```
<input type="hidden" name="UPAY_SITE_ID" value="3">
<input type="hidden" name="TICKET" value="12345">
<input type="hidden" name="TICKET_NAME" value="12345">
</form>
<SCRIPT LANGUAGE="JavaScript">function submitForm()
{document.myform.submit();}
</SCRIPT>
</BODY>
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

AAWS Web Services Description Language

During setup, the system generates the Web Services Description Language (WSDL) for SAD_ADMISSIONS.

The following diagram lists the SAD_ADMISSIONS web service operations and their messages. The diagram also shows an example of a location from where an online application can access the SAD_ADMISSIONS web service.



Graphical Representation for SAD_ADMISSIONS WSDL

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Sample Code for an Online Application

The following example shows how an online application uses ADF to call the SAD_SAVEAPPL service operation:

1. Create the web service proxy using the WSDL URL. It will generate the proxy code (methods) to invoke the service operations.

Sample code of generated proxy:

```
@WebServiceClient(wsdlLocation="http://<hostname>/PSIGW/PeopleSoftServiceListeningC
onnector/SAD_ADMISSIONS.1.wsdl",
targetNamespace="http://xmlns.oracle.com/Enterprise/HCM/services/SAD_ADMISSIONS.1",
  name="SAD_ADMISSIONS")
public class SAD_ADMISSIONS
  extends Service
{  private static URL wsdlLocationURL;
  private static Logger logger;
  static
  {  try    {      logger = Logger.getLogger("sadAdmissions.SAD_ADMISSIONS");
           URL baseUrl = SAD_ADMISSIONS.class.getResource(".");
           if (baseUrl == null)  {
        wsdlLocationURL =
SAD_ADMISSIONS.class.getResource("http://<hostname>/PSIGW/PeopleSoftServiceListenin
gConnector/SAD_ADMISSIONS.1.wsdl");
        if (wsdlLocationURL == null)   {
          baseUrl = new File(".").toURL();
          wsdlLocationURL =
              new URL(baseUrl,
"http://<hostname>/PSIGW/PeopleSoftServiceListeningConnector/SAD_ADMISSIONS.1.wsdl"
);
        }
      }     else      {
                if (!baseUrl.getPath().endsWith("/")) {
         baseUrl = new URL(baseUrl, baseUrl.getPath() + "/");
}                wsdlLocationURL =
            new URL(baseUrl,
"http://<hostname>/PSIGW/PeopleSoftServiceListeningConnector/SAD_ADMISSIONS.1.wsdl"
);
      }
    }    catch (MalformedURLException e)
    {      logger.log(Level.ALL,
          "Failed to create wsdlLocationURL using
http://<hostname>/PSIGW/PeopleSoftServiceListeningConnector/SAD_ADMISSIONS.1.wsdl",
          e);
    }
  }
//...
//...
//...
  @WebEndpoint(name="SAD_ADMISSIONS_Port")
  public SAD_ADMISSIONS_PortType getSAD_ADMISSIONS_Port()
  {    return (SAD_ADMISSIONS_PortType) super.getPort(new
QName("http://xmlns.oracle.com/Enterprise/HCM/services/SAD_ADMISSIONS.1",

"SAD_ADMISSIONS_Port"),
                                                        SAD_ADMISSIONS_PortType.class);
  }
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

```
//...
}

  @WebMethod(operationName="SAD_SAVEAPPL", action="SAD_SAVEAPPL.v1")
  @SOAPBinding(parameterStyle=ParameterStyle.BARE)
  @Action(input="SAD_SAVEAPPL.v1", fault =
      {
@FaultAction(value="http://xmlns.oracle.com/Enterprise/HCM/services/SAD_ADMISSIONS.
1/SAD_ADMISSIONS_PortType/SAD_SAVEAPPL/Fault/SAD_FAULT_RESP.V1",
          className=sadAdmissions.SAD_FAULT_RESPV1.class) },
output="http://xmlns.oracle.com/Enterprise/HCM/services/SAD_ADMISSIONS.1/SAD_ADMISS
IONS_PortType/SAD_SAVEAPPL_RESP.V1")
  @WebResult(targetNamespace="http://xmlns.oracle.com/Enterprise/HCM/services",
    partName="parameter", name="SAD_SAVEAPPL_RESP")
  public SADSAVEAPPLRESP
sadSAVEAPPL(@WebParam(targetNamespace="http://xmlns.oracle.com/Enterprise/HCM/servi
ces",
      partName="parameter", name="SAD_SAVEAPPL_REQ")
    SADSAVEAPPLREQ parameter)
    throws sadAdmissions.SAD_FAULT_RESPV1;
```

2. Use the proxy methods to invoke the service operation:

```
//Create the Application request
        SADSAVEAPPLREQ reqSaveAppl = new SADSAVEAPPLREQ();
        //Construct the Application request by setter methods created by web
service proxy

        //Create the Application response
        SADSAVEAPPLRESP respSaveAppl = new SADSAVEAPPLRESP();

        try {
            SecurityPolicyFeature[] securityFeatures =
                            new SecurityPolicyFeature[] { new
SecurityPolicyFeature("oracle/wss_username_token_client_policy") };
            sAD_ADMISSIONS = new SAD_ADMISSIONS();
            SAD_ADMISSIONS_PortType sAD_ADMISSIONS_PortType =
sAD_ADMISSIONS.getSAD_ADMISSIONS_Port(securityFeatures);

            Map<String, Object> requestContext = ((BindingProvider)
sAD_ADMISSIONS_PortType).getRequestContext();

SAD_ADMISSIONS_PortClient.setPortCredentialProviderList(requestContext);
            // Add WS-Security header username & password if we currently
authenticated (logged on)
            requestContext.put(BindingProvider.USERNAME_PROPERTY, username);
            requestContext.put(BindingProvider.PASSWORD_PROPERTY, password);

            // invoke the sadSAVEAPPL method created by we service proxy
            respSaveAppl = sAD_ADMISSIONS_PortType.sadSAVEAPPL(reqSaveAppl);

            // pass the response (respSaveAppl) to the response handler

        }
        catch (SOAPFaultException sfe) {
            //handle SOAP Fault
        }
        catch (Exception e) {
            //handle any other exception
        }
```

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.