# ACGI - [Web Site Integration and Self Service Web Applications](https://acgisoftware.jira.com/wiki/display/DOC/Web+Site+Integration+and+Self+Service+Web+Applications)

# Overview

Association Anywhere includes a number of pre-built applications as well as programming and interface tools that allow you to integrate one or more websites with your Association Anywhere database.  This section of the documentation space addresses the applications, programming tools and additional techniques that are available to help you with your integration tasks.    
The tools and applications described in this section are intended primarily for the deployment of "Self Service Applications".  Like pay-at-the-pump, or like ATM machines, these self-service applications are designed to tap into your members and website visitors as a data entry resource, and allow them to interact with the database as your internal staff might.  Members can retrieve or enter information so that your staff don't have to.  Association anywhere provides pre-built applications called plug-ins, and API's that allow you to integrate your own applications.

NOTE:  A comprehensive listing of all APIs (i.e., stored procedures and XML web services) is provided in **Excel spreadsheet**.  This listing provides individual links to the documentation where available.  This approach is also replicated on the **Application Programming Interfaces (API) By Module** wiki page.

## The Big Picture: A Range of Integration Options balancing Cost and User Experience

Association Anywhere offers a range of potential ways to integrate your database with the web.  Some items are pre-built and others require more advanced technical skill sets to implement.

## Association Anywhere Website SSA Plug-In Concepts

Plug-Ins are small, simple, pre-built web applications that are included in the Association Anywhere suite of products.  Each performs a specific, discrete function common to member-only web sites.  The application's behavior is controlled by configuration settings in your Association Anywhere AMS, and can be "plugged in" to your existing web site with a minimal amount of configuration, by simply imbedding a link on your website to a URL.  Plug-ins are usually the best way to get real-time member interaction with your AMS at the lowest cost.   
Because they are discrete and modular functions, you may need to use two or three pre-built plug-ins in series to accomplish a task.  For instance, a purchase transaction (join, donate, register, etc) may consist of separate plug-ins used in series.  The first plug-in would require login or creating an account.  A second plug-in would be used to take the order information for the membership, the donation, event registration, etc. that is being purchased.  Finally, the 3rd plug-in would take the payment.   
If your main objective is to consolidate all of the transaction into a streamlined, single page, then you would need to do custom programming and use API's to achieve that result.  You would not be able to use plug-ins.

## Application Programming Interface (API) Concepts

Association Anywhere includes a variety of different types of application programming interfaces (APIs).  Different APIs are better suited for different types of tasks.  Multiple options are provided so that you can use the right tool for the job.

* **HTTP-relay** - An integration technique for use by your website or 3rd party applications where information can be passed to or retrieved from association anywhere via web form posts.  The information passed via this technique is validated and posted to the database, and then the browser is relayed on to one of 2 pages, depending on success or failure validity of the information that was posted.   Information can be passed over HTTP via either the GET method (as with a link) or POST method (as with a form post with input fields and hidden variables on a web form).  This method is often useful for a low cost method of integration with only a minimal amount of programming complexity.
* **Oracle Stored Procedure** - An integration technique for more robust and complex integration projects that requires your website or 3rd party application to establish a secure database connection to the underlying Oracle database and invoke stored procedures in the native language used to develop Association Anywhere.  This technique provides the most reliable and fastest performing integration for web sites and 3rd party applications that require programming as well as real time integration with Association Anywhere.  The Oracle stored procedures are function calls that are oriented around the common data interchange points pertinent to integrating with websites.  Sample code for invoking the Oracle Stored Procedures via programming languages can be found at 9.2 Sample Code for Stored Procedure Calls using PHP
* **Pre-Built SOAP WebServices**  - there are a number of SOAP based web services, built around the same functional interface points as the Oracle Stored procedure APIs, but are accessed as a SOAP web service rather than via a database connection. In general, the SOAP based web services are older, less popular and are not the current direction of ACGI API development
* **Pre-Built RESTful WebServices** - there are a number of RESTful web services which are increasingly the method of choice for most integration projects.  Use of these web services requires integrators to work with our mutual client, to establish an integrator account and password, and to provide development and production IP addresses to whitelist.

With the purchase of certain additional licenses, additional contractual arrangements, and demonstrated technical competency, the following integration options can also be made available upon request to clients with sufficient technical capabilities.

* **Direct Database Access** - This technique includes READ ONLY database access to some of the main database tables in a simplified database model.  Website programmers can then write custom queries to read information from the database directly.  This technique can be very useful particularly for publication of directories, for populating a 3rd party single sign on / LDAP system, or for website integration models where standalone data is replicated because a decoupled model is preferred.  Direct database access can be used for either standalone or real-time integration and is a better tool for fast performance and large data transfers.
* Do It Yourself (DIY) Webservices using **Query as a Web Service (QaaWS)** - Using Query as a Web Service from the Business Objects suite of tools, you can build your own SOAP web services to pull data from the database, and use them in web applications that consume web services.   These are well suited to website integration tasks and provide a high degree of code generation to save you programming time.  They are poorly suited to large data transfers, and with web services overhead they do not perform as fast as direct database access APIs.

[NTCA Single Sign On Information](https://acgisoftware.jira.com/wiki/display/NTCA/NTCA+Single+Sign+On+Information)

NTCA Core Team Technical Notes and Project Information

This is the information for NetStrategies implementation.

Once Live, the AMS will be the authentication source for both the SSA and the password-protected areas of the NTCA, FRS, PAC (formerly TECO) and Rural Telecom websites. To end users, it will be seamless, meaning if they log into the SSA, they don't have to relogin to the website and vice versa.

In order to preserve the Login IDs of current website users, those current website users were converted into AA whenever possible (specifically they were tied to an ID in MPACT using various keys like company ID & email address then converted into AA). This resulted in a number of duplicate records that will need to be merged. However current website users will retain their Login IDs. Everyone must reset their passwords because they were hashed and irretrievable in the old system. AA generated generic passwords for the users, which they do not know.

NTCA's web host, NetStrategies,(NS)  is using APIs from ACGI to authenticate. NS sends the login and password submitted by the user and the API returns access roles (or a flag indicating the user is not known).  NS then has logic to determine what the user can see/do based on those SSA User Roles.

The APIs are environment specific and permission is granted by IP address as well as a user name and password. ACGI does not allow a prod and dev access from the same IP address, so in Nov 2015, the dev websites was moved to new dev partitions with their own IPs. As of 12/2015, they are listed below. Essentially NTCA, PAC & RT share a prod IP.  FRS has it's own IP address. And all 4 sites use the same DEV IP.

In 2016 TECO rebranded as NTCA PAC. URLs were changed. AA SSA user roles for TECO access were not changed; roles outlined below remain in effect.

In 2016 a new Sucure firewall was added to the NTCA website. This new firewall does not affect AA SSA setups.

**AA SSA User Roles - used to control access to SSA menus and functionality, as well as by the website vendor to grant access to certain areas & sites. *(note: these are the active roles, used by NS and SSA as-of GoLive. There may be more listed but unused. These are set up under Admin > SSA > User Roles and the logic is set up under User Roles SQL)***

* Guest - Anyone with a login; all users have this role and additional roles are layered on top of it
* Member - individuals tied to an active member company via an employment relationship (Inherited Membership)
* CO\_ADMIN - Based off their Company Admin Y/N Attribute
* TECO\_ACCESS - used to grant access to the TECO website (Based off their company TECO\_YEARS\_GRANTED =  the current year AND the individual attribute TECO\_SOLIC\_CLASS = Y) - this is not yet working 12/2015 and is based on Attribute TECO\_MEMBER
* RT\_ACCESS - used to grant access to the Rural Telecom content (based on an active RT membership)
* TECO\_BOD - used for special areas on TECO website for Board of Directors documents (based on BOARD\_TECO Committee membership)
* STAFF - used to designate NTCA staff; set up but not used nor tested thoroughly

ROLES Used at Login for AMS authentication

* + - MEMBER
    - NONMEMBER
    - CO\_ADMIN
    - CO\_ADMIN\_NONMEMBER
    - STAFF
    - GUEST
    - RT\_ACCESS
    - TECO\_ACCESS
    - TECO\_BOD
    - HAS\_RENEWAL
    - ASSOC\_MEMBER

|  |  |  |
| --- | --- | --- |
| **Server** | **Environment** | **IP** |
| [www.ntca.org](http://www.ntca.org/) | Prod |  |
| [dev.ntca.org](http://dev.ntca.org/) | Dev |  |
| [www.ntcapac.org](http://www.ntcapac.org/) | Prod |  |
| [dev.ntcapac.org](http://dev.ntcapac.org/) | Dev |  |
| [www.frs.org](http://www.frs.org/) | Prod |  |
| [dev.frs.org](http://dev.frs.org/) | Dev |  |
| [www.ruraltelecom.org](http://www.ruraltelecom.org/) | Prod |  |
| [dev.ruraltelecom.org](http://dev.ruraltelecom.org/) | Dev |  |

**Information given to NetStrategies for testing/development:**

a. ACGI login page - <https://online.ntca.org/ntcadevssa/ssaauthmain.login_check>

b. ACGI registration page. [https://online.ntca.org/ntcadevssa/censsacustmast.insert\_page?](https://online.ntca.org/ntcadevssa/censsacustmast.insert_page)

c. ACGI password reset page. <https://online.ntca.org/ntcadevssa/ssaauthmain.forget_password?p_referrer=&p_context=CEN>

**APIs used:**

to be fully documented after GoLive

Authentication API -- censsawebsvclib.authentication

Package -- censsawebsvclib

Procedure -- authentication

Description -- Authenticates user using various methods (e.g., Username/password, ID/name)

XML/REST Web Service --

INPUT --

<?xml version="1.0"?>

<authentication-request>

<integratorUsername></integratorUsername><!-- REQUIRED if bypassing IP whitelisting, provided by ACGI -->

<integratorPassword></integratorPassword><!-- REQUIRED if bypassing IP whitelisting, provided by ACGI -->

<cust-id></cust-id>

<last-nm></last-nm>

<alias></alias>

<username></username>

<password></password>

<session-id></session-id>

</authentication-request>

OUTPUT --

<?xml version="1.0"?>

<authentication>

<authenticated>true</authenticated>

<authentication-message>Might be for a successful or a failed authentication</authentication-message>

<authentication-error-id>11</authentication-error-id>

<session>

<session-id>Example Hash Code Value</session-id>

<session\_id>Example Hash Code Value</session\_id>

<breadcrumb-session-id>Example Breadcrumb Session ID Value</breadcrumb-session-id>

<login-serno>1234567</login-serno>

<roles>

<role>MEMBER</role>

<role>GUEST</role>

</roles>

</session>

<customer>

<cust-id>A999999999</cust-id>

<cust-type>I</cust-type>

<name>

<display-name>John Doe</display-name>

<last-name>Doe<last-name>

<first-name>John</first-name>

<company-name>That Big University</company-name>

<title-name>Professor</title-name>

</name>

<cust-email>test@hotmail.com</cust-email>

</customer>

<memberships>

<membership><!-- can repeat -->

<member>true</member>

<status>ACTIVE</status>

<subgroup-ID>ABC</subgroup-ID>

<subgroup-type>NA</subgroup-type>

<subgroup-name>Human-Readable Value</subgroup-name>

<class-code>SAMPLE CODE</class-code>

<subclass-code>SAMPLE CODE</subclass-code>

<level-of-service>Sample Description of Level of Service</level-of-service>

<end-of-service-date>2011-12-31</end-of-service-date>

<paid-through-date>2010-12-31</paid-through-date>

<inheritedFromCustId></inheritedFromCustId>

</membership>

</memberships>

<subscriptions>

<subscription><!-- can repeat -->

<package-code>SAMPLE CORE PACK CODE</package-code>

<package-name>Sample Core Package Name</package-name>

<benefit-of-membership>true</benefit-of-membership>

<associated-subgroup-ID>ABC</associated-subgroup-ID>

<start-date>2011-12-31</start-date>

<end-of-service-date>2011-12-31</end-of-service-date>

<paid-through-date>2010-12-31</paid-through-date>

</subscription>

</subscriptions>

</authentication>

Authentication API -- censsawebsvclib.get\_valid\_roles\_xml

Package -- censsawebsvclib

Procedure -- get\_valid\_roles\_xml

Description -- Gets list of all possible SSA roles

XML/REST Web Service --

INPUT --

<?xml version="1.0" encoding="UTF-8"?>

<roleRequest>

<filter></filter>

<integratorUsername></integratorUsername>

<integratorPassword></integratorPassword>

</roleRequest>

OUTPUT --

<?xml version="1.0" encoding="UTF-8"?>

<role-list>

<role></role> \*\* Can Repeat

<status></status> \*\* SUCCESS or FAILURE

<message></message> \*\* Only included when status = FAILURE

</role-list>

eStore Purchases

eStore API -- ecssawebsvclib.get\_purchased\_products\_xml

Package -- ecssawebsvclib

Procedure -- get\_purchased\_products\_xml

Description -- Serves as an XML alternative to return the listing of products purchased by a specified customer

XML/REST Web Service --

INPUT --

<?xml version="1.0" encoding="UTF-8" ?>

<ecord-request>

<vendorId>acgi</vendorId> <!-- provided by ACGI -->

<vendorPassword>acgipass</vendorPassword> <!-- provided by ACGI -->

<custId>62000367</custId>

<orderSerno></orderSerno> <!-- optional -->

<productType></productType> <!-- optional -->

</ecord-request>

OUTPUT --

<?xml version="1.0" encoding="UTF-8" ?>

<orders>

<order> \*\* can repeat

<productSerno></productSerno>

<productId></productId>

<productName></productName>

<length></length>

<width></width>

<height></height>

<weight></weight>

<activeFlag></activeFlag>

<internalOrderFlag></internalOrderFlag>

<firstAvailableDate></firstAvailableDate> \*\* MM/DD/YYYY

<defaultUnitCost></defaultUnitCost>

<showProductRelativeURL></showProductRelativeURL>

<imageThumbnail></imageThumbnail>

<imageFullsize></imageFullsize>

<orderDate></orderDate> \*\* MM/DD/YYYY

<orderStatus></orderStatus>

<orderSerno></orderSerno> \*\* The order\_serno

<productType></productType> \*\* The product template\_shortlist

<orderEmail></orderEmail>

<attributes> \*\* will not appear if none

<attribute>

<attribute-type></attribute-type>

<attribute-text></attribute-text>

</attribute>

</attributes>

</order>

<status>SUCCESS or FAILURE</status>

<message>Element will not appear if status = SUCCESS</message>

</orders>

eStore API -- ecssawebsvclib.get\_payment\_types\_xml

Package -- ecssawebsvclib

Procedure -- get\_payment\_types\_xml

Description -- Returns the payment types that are valid for a shopping cart during the check-out process

XML/REST Web Service --

INPUT --

<?xml version="1.0" encoding="UTF-8" ?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="paymentTypeRequest">

<xs:complexType>

<xs:sequence>

<xs:element name="integratorUsername" type="xs:string"/> <!-- Provided by ACGI -->

<xs:element name="integratorPassword" type="xs:string"/> <!-- Provided by ACGI -->

<xs:element name="custId" type="xs:string"/>

<xs:element name="cartId" type="xs:string"/>

<xs:element name="behaviorCode" minOccurs="0" maxOccurs="5"> <!-- Omitting behaviorCode results in all behaviorCodes being considered -->

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="ACH"/>

<xs:enumeration value="ACHRT"/>

<xs:enumeration value="CASH EQUIV"/>

<xs:enumeration value="CREDIT CARD"/>

<xs:enumeration value="VOUCHER"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

OUTPUT XSD --

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="paymentTypeResponse">

<xs:complexType>

<xs:sequence>

<xs:element name="paymentType" minOccurs="0" maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:element name="paymentTypeCd" type="xs:string" />

<xs:element name="description" type="xs:string" />

<xs:element name="behaviorCode" type="xs:string" />

<xs:element name="account" minOccurs="0">

<xs:complexType>

<xs:sequence>

<xs:element name="custAccountSerno" type="xs:unsignedLong" />

<xs:element name="preferredAccountFl">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="Y" />

<xs:enumeration value="N" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="accountName">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="60" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="accountType">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="CREDITCARD" />

<xs:enumeration value="ACH" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoStreet1">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="60" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoStreet2">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="60" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoCityNm">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="30" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoStateCd">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="3" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoStateNm">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="30" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoPostalCd">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="15" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoCountryCd">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="3" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoCountryNm">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="60" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="billtoProvinceNm">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="30" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="cardHolderName" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="60" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="cardNumber" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="32" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="cardExpireMonth" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:minLength value="2" />

<xs:maxLength value="2" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="cardExpireYear" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:minLength value="4" />

<xs:maxLength value="4" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="achAccountHolderName" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="60" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="achRoutingNumber" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="32" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="achAccountNumber" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:maxLength value="32" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="achAccountType" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="CHECKING" />

<xs:enumeration value="SAVINGS" />

</xs:restriction>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

<xs:attribute name="savedAccount" use="required">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="Y" />

<xs:enumeration value="N" />

</xs:restriction>

</xs:simpleType>

</xs:attribute>

</xs:complexType>

</xs:element>

<xs:element name="status">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:enumeration value="SUCCESS" />

<xs:enumeration value="FAILURE" />

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="message" type="xs:string" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

OUTPUT --

<?xml version="1.0" encoding="UTF-8"?>

<paymentTypeResponse>

<paymentType savedAccount="Y">

<paymentTypeCd><![CDATA[0H]]></paymentTypeCd>

<description><![CDATA[Visa/MasterCard]]></description>

<behaviorCode><![CDATA[CREDIT CARD]]></behaviorCode>

<account>

<custAccountSerno><![CDATA[4022]]></custAccountSerno>

<preferredAccountFl><![CDATA[Y]]></preferredAccountFl>

<accountName><![CDATA[Saved Account One]]></accountName>

<accountType><![CDATA[CREDITCARD]]></accountType>

<billtoStreet1><![CDATA[123 Main Street]]></billtoStreet1>

<billtoStreet2 />

<billtoCityNm><![CDATA[Columbia]]></billtoCityNm>

<billtoStateCd><![CDATA[MD]]></billtoStateCd>

<billtoStateNm><![CDATA[Maryland]]></billtoStateNm>

<billtoPostalCd><![CDATA[21075]]></billtoPostalCd>

<billtoCountryCd />

<billtoCountryNm />

<billtoProvinceNm />

<cardHolderName><![CDATA[John Smith]]></cardHolderName>

<cardNumber><![CDATA[XXXXXXXXXXXX1111]]></cardNumber>

<cardExpireMonth><![CDATA[01]]></cardExpireMonth>

<cardExpireYear><![CDATA[2018]]></cardExpireYear>

</account>

</paymentType>

<paymentType savedAccount="Y">

<paymentTypeCd><![CDATA[0H]]></paymentTypeCd>

<description><![CDATA[Visa/MasterCard]]></description>

<behaviorCode><![CDATA[CREDIT CARD]]></behaviorCode>

<account>

<custAccountSerno><![CDATA[4023]]></custAccountSerno>

<preferredAccountFl><![CDATA[N]]></preferredAccountFl>

<accountName><![CDATA[Saved Account Two]]></accountName>

<accountType><![CDATA[CREDITCARD]]></accountType>

<billtoStreet1><![CDATA[123 Main Street]]></billtoStreet1>

<billtoStreet2 />

<billtoCityNm><![CDATA[Columbia]]></billtoCityNm>

<billtoStateCd><![CDATA[MD]]></billtoStateCd>

<billtoStateNm><![CDATA[Maryland]]></billtoStateNm>

<billtoPostalCd><![CDATA[21075]]></billtoPostalCd>

<billtoCountryCd><![CDATA[USA]]></billtoCountryCd>

<billtoCountryNm><![CDATA[United States of America]]></billtoCountryNm>

<billtoProvinceNm />

<cardHolderName><![CDATA[John Smith]]></cardHolderName>

<cardNumber><![CDATA[XXXXXXXXXXXX5555]]></cardNumber>

<cardExpireMonth><![CDATA[01]]></cardExpireMonth>

<cardExpireYear><![CDATA[2017]]></cardExpireYear>

</account>

</paymentType>

<paymentType savedAccount="N">

<paymentTypeCd><![CDATA[0I]]></paymentTypeCd>

<description><![CDATA[American Express]]></description>

<behaviorCode><![CDATA[CREDIT CARD]]></behaviorCode>

</paymentType>

<paymentType savedAccount="N">

<paymentTypeCd><![CDATA[0H]]></paymentTypeCd>

<description><![CDATA[Visa/MasterCard]]></description>

<behaviorCode><![CDATA[CREDIT CARD]]></behaviorCode>

</paymentType>

<status>SUCCESS</status>

</paymentTypeResponse>

# [Employment API - Get Employment Details XML - CENEMPWEBSVCLIB.GET\_EMPLOYERS\_XML](https://acgisoftware.jira.com/wiki/display/DOC/Employment+API+-+Get+Employment+Details+XML+-+CENEMPWEBSVCLIB.GET_EMPLOYERS_XML)

#### ****Web Service****

**CENEMPWEBSVCLIB**

#### ****Procedure****

**get\_employers\_xml**

#### ****Purpose****

The following web service procedure, cenempwebsvclib.get\_employers\_xml, serves as an XML alternative to cencustempselapi.get\_employers. Arguments are passed as an xml document with the name “p\_input\_xml\_doc”

#### Input Document

<?xml version="1.0"?>

<employer-request>

    <vendor-id></vendor-id>

    <vendor-password></vendor-password>

    <cust-id></cust-id>

    <org-type></org-type>                   \*\* Optional

    <preferred-only></preferred-only>       \*\* Optional  Y/N

    <current-only></current-only>           \*\* Optional  Y/N

</employer-request>

#### Output Document

<?xml version="1.0" encoding="UTF-8" ?>

<employment-list>

   <employment>             \*\* can repeat

      <emp-serno></emp-serno>

      <employer-id></employer-id>

      <employer-name></employer-name>        \*\* CDATA Wrapped

      <employee-id></employee-id>

      <employee-name></employee-name>

      <title-nm></title-nm>

      <title-nm-long></title-nm-long>

      <title-cd></title-cd>

      <function-cd></function-cd>

      <start-date></start-date>  \*\* MM/DD/YYYY - CDATA Wrapped

      <end-date></end-date>       \*\* MM/DD/YYYY - CDATA Wrapped

      <hrs-week></hrs-week>

      <emp-remark></emp-remark>

      <preferred></preferred>     \*\* Y/N/NULL

      <create-date></create-date> \*\* MM/DD/YYYY - CDATA Wrapped

      <create-by></create-by>

      <last-change-date></last-change-date> \*\* MM/DD/YYYY - CDATA

      <last-change-by></last-change-by>

      <checksum></checksum>

   </employment>

   <status>SUCCESS or FAILURE</status>

   <message>Element will not appear if status = SUCCESS</message>

</employment-list>

Customer Attributes

Central Customer API (Customer Attributes) -- censsawebsvclib.get\_attributes\_xml

Package -- censsawebsvclib

Procedure -- get\_attributes\_xml

Description -- Provides an XML alternative to return a list of address records for the corresponding customer ID

XML/REST Web Service --

INPUT --

<?xml version="1.0"?>

<custAttributeDtlInfoRequest>

<integratorUsername></integratorUsername> <!-- REQUIRED, provided by ACGI -->

<integratorPassword></integratorPassword> <!-- REQUIRED, provided by ACGI -->

<custId></custId> <!-- REQUIRED -->

<attributeType></attributeType> <!-- OPTIONAL -->

<attributeGroup></attributeGroup> <!-- OPTIONAL -->

</custAttributeDtlInfoRequest>

OUTPUT --

<custAttributes>

<attribute>

<custId></custId>

<type></type>

<typeDescription></typeDescription>

<code></code>

<codeDescription ></codeDescription >

<cvar></cvar>

<nvar></nvar>

<dvar></dvar> <!-- Format YYYY-MM-DD -->

<createDate></createDate> <!-- Format YYYY-MM-DD -->

<createBy></createBy>

<lastChangeDate></lastChangeDate> <!-- Format YYYY-MM-DD -->

<lastChangeBy></lastChangeBy>

<lockCode></lockCode>

</attribute>

<status></status> <!-- (Returns SUCCESS or FAILURE) -->

<message></message>

</custAttributes>

Customer Directory

Central Customer API (Company Directory) -- censsawebsvclib.get\_directory\_list\_xml

Package -- censsawebsvclib

Procedure -- get\_directory\_list\_xml

Description -- Returns a listing of customers and their specified directory contact details based on the data passed in request

XML/REST Web Service --

INPUT --

<?xml version="1.0"?>

<custDirectoryRequest>

<integratorUsername></integratorUsername> <!-- REQUIRED, provided by ACGI -->

<integratorPassword></integratorPassword> <!-- REQUIRED, provided by ACGI -->

<directoryId><![CDATA[]]></directoryId> <!-- OPTIONAL -->

<subgroupId><![CDATA[]]></subgroupId> <!-- OPTIONAL -->

<classCode><![CDATA[]]></classCodeId> <!-- OPTIONAL -->

<firstName><![CDATA[]]></firstName> <!-- OPTIONAL -->

<lastName><![CDATA[]]></lastName> <!-- OPTIONAL -->

<companyName><![CDATA[]]></companyName> <!-- OPTIONAL -->

<cityName><![CDATA[]]></cityName> <!-- OPTIONAL -->

<stateCode><![CDATA[]]></stateCode> <!-- OPTIONAL -->

<provinceName><![CDATA[]]></provinceName> <!-- OPTIONAL -->

<postalCode><![CDATA[]]></postalCode> <!-- OPTIONAL -->

<distance><![CDATA[]]></distance> <!-- OPTIONAL -->

<countryCode><![CDATA[]]></countryCode> <!-- OPTIONAL -->

<email><![CDATA[]]></email> <!-- OPTIONAL -->

</custDirectoryRequest>

OUTPUT --

<?xml version="1.0" encoding="UTF-8" ?>

<custDirectoryRequestResult>

<customer> <!-- can repeat -->

<customerId></customerId>

<customerTypeCode></customerTypeCode>

<customerTypeDescr></customerTypeDescr>

<firstName><![CDATA[]]></firstName>

<lastName><![CDATA[]]></lastName>

<companyName><![CDATA[]]></companyName>

<displayName><![CDATA[]]></displayName>

<cityName><![CDATA[]]></cityName>

<stateCode><![CDATA[]]></stateCode>

<subgroup> <!-- can repeat -->

<subgroupId><![CDATA[]]></subgroupId>

<subgroupTypeCode><![CDATA[]]></subgroupTypeCode>

<subgroupTypeDescr><![CDATA[]]></subgroupTypeDescr>

</subgroup>

</customer>

<status></status> <!-- (Returns SUCCESS or FAILURE) -->

<message></message>

</custDirectoryRequestResult>

Company Directory API

Central Customer API (Company Directory) -- censsawebsvclib.get\_company\_list\_xml

Package -- censsawebsvclib

Procedure -- get\_company\_list\_xml

Description -- Returns a list of all companies in the system (i.e., customers with a customer type of 'C')

XML/REST Web Service --

INPUT --

<?xml version="1.0" encoding="UTF-8"?>

<companyRequest>

<integratorUsername></integratorUsername>

<integratorPassword></integratorPassword>

</companyRequest>

OUTPUT --

<?xml version="1.0" encoding="UTF-8"?>

<companyList>

<company> \*\* Can Repeat

<id></id>

<name></name> \*\* CDATA wrapped

<street1></street1> \*\* CDATA wrapped; omitted if null

<street2></street2> \*\* CDATA wrapped; omitted if null

<street3></street3> \*\* CDATA wrapped; omitted if null

<city></city> \*\* CDATA wrapped; omitted if null

<state></state> \*\* CDATA wrapped; omitted if null

<province></province>\*\* CDATA wrapped; omitted if null

<zip></zip> \*\* CDATA wrapped; omitted if null

<country></country> \*\* CDATA wrapped; omitted if null

</company>

<status></status> \*\* SUCCESS or FAILURE

<message></message> \*\* Only included when status = FAILURE

</companyList>

Events

Event API -- censsawebsvclib.get\_event\_info\_xml

Package -- censsawebsvclib

Procedure -- get\_event\_info\_xml

Description -- Provides an XML/REST alternative to return a listing of events based on criteria specified

XML/REST Web Service --

INPUT --

<?xml version="1.0"?>

<event-request>

<vendor-id>acgi/vendor-id>

<vendor-password>acgipass</vendor-password>

<cust-id>3200678</cust-id> <!-- Optional -->

<start-date></start-date> <!-- Optional -->

<end-date></end-date> <!-- Optional -->

<status></status> <!-- Optional (can include multiple as comma-delimited list e.g. ACTIVE,PROPOSED) -->

<category></category> <!-- Optional -->

<event-type></event-type> <!-- Optional -->

<item-type></item-type> <!-- Optional -->

<state-code>MD</state-code> <!-- Optional -->

<sponsor></sponsor> <!-- Optional -->

</event-request>

OUTPUT --

<?xml version="1.0" encoding="UTF-8" ?>

<event-list>

<event> <!-- can repeat -->

<id></id> <!-- some elements will be CDATA wrapped -->

<program-name></program-name>

<name></name> <!-- CDATA wrapped -->

<type></type>

<type-descr></type-descr> <!-- Event type description from the cen\_code\_genval where code\_ty = 'EVT\_EVENTTYPE' -->

<status></status>

<start-dt></start-dt> <!-- Format YYYYMMDDHH24MISS -->

<end-dt></end-dt> <!-- Format YYYYMMDDHH24MISS -->

<deadline-dt></deadline-dt> <!-- Format YYYYMMDDHH24MISS -->

<location-nm></location-nm> <!-- CDATA wrapped -->

<location-street1></location-street1> <!-- CDATA wrapped -->

<location-street2></location-street2> <!-- CDATA wrapped -->

<location-city></location-city>

<location-state></location-state>

<location-zip></location-zip>

<location-country></location-country>

<register-url></register-url> <!-- CDATA wrapped -->

<registration-status></registration-status>

<attribute-list>

<attribute> <!-- can repeat -->

<type></type>

<code></code>

<character-value></character-value>

<number-value></number-value>

<date-value></date-value> <!-- CDATA wrapped MM/DD/YYYY -->

</attribute>

</attribute-list>

<validRegistrationTypes>

<regType> <!-- can repeat -->

<type></type>

<descr></descr>

<defaultCl><defaultCl>

<waitlistingFlag><waitlistingFlag>

<staffFlag><staffFlag>

<regType>

</validRegistrationTypes>

</event>

<status>SUCCESS or FAILURE</status>

<message>Element will not appear if status = SUCCESS</message>

</event-list>

Event API -- censsawebsvclib.get\_eventreg\_info\_xml

Package -- censsawebsvclib

Procedure -- get\_eventreg\_info\_xml

Description -- Provides an XML/REST alternative to return the registration details for a specific customer and event

XML/REST Web Service --

INPUT --

<?xml version="1.0" encoding="UTF-8"?>

<eventreg-request>

<vendor-id></vendor-id>

<vendor-password></vendor-password>

<cust-id></cust-id><!-- Optional, but should not be null if event-id is null -->

<event-id></event-id><!-- Optional, but should not be null if cust-id is null -->

<first-name></first-name><!-- Optional, used as an alternate lookup parameter -->

<last-name></last-name><!-- Optional, used as an alternate lookup parameter -->

<company-name></company-name><!-- Optional, used as an alternate lookup parameter -->

<email></email><!-- Optional, used as an alternate lookup parameter -->

<secondaries-require-ticket></secondaries-require-ticket><!-- Optional boolean input. Further explanation below -->

<regAttributes include="true" includeAll="true">

<!--

The following are examples of querying registration attributes by type or type+code.

Do not include the attr children if includeAll is set to "true"

<attr type="CELLPHONE" />

<attr type="FREE\_GIFT" code="TSHIRT" />

-->

</regAttributes>

<regItemAttributes include="true" includeAll="true">

<!--

The following are examples of querying registration item attributes by type or type+code.

Do not include the attr children if includeAll is set to "true"

<attr type="DIET\_RESTRICTION" />

<attr type="FREE\_GIFT" code="TSHIRT" />

-->

</regItemAttributes>

</eventreg-request>

OUTPUT --

<?xml version="1.0" encoding="UTF-8" ?>

<registrations>

<registration><!-- can repeat -->

<regi-serno></regi-serno>

<customer-id></customer-id>

<event-id></event-id>

<registration-date></registration-date><!-- MM/DD/YYYY -->

<registration-type></registration-type>

<registration-name></registration-name>

<representing></representing>

<billto-id></billto-id>

<promo-cd></promo-cd>

<purchase-order></purchase-order>

<prim-item-id></prim-item-id>

<prim-reg-status></prim-reg-status>

<total-charges></total-charges>

<total-payment></total-payment>

<balance></balance>

<event-nm></event-nm>

<program-nm></program-nm>

<primary-item-descr></primary-item-descr>

<event-start-dt></event-start-dt>

<event-end-dt></event-end-dt>

<location-name></location-name>

<location-street1></location-street1>

<location-street2></location-street2>

<location-city></location-city>

<location-state></location-state>

<location-zip></location-zip>

<location-country></location-country>

<location-country-descr></location-country-descr>

<first-name></first-name>

<last-name></last-name>

<company-name></company-name>

<email></email>

<evt-reg-street1></evt-reg-street1>

<evt-reg-street2></evt-reg-street2>

<evt-reg-street3></evt-reg-street3>

<evt-reg-city></evt-reg-city>

<evt-reg-state></evt-reg-state>

<evt-reg-zip></evt-reg-zip>

<evt-reg-country></evt-reg-country>

<regAttributes>

<regAttribute><!-- can repeat -->

<type><![CDATA[]]></type>

<typeName><![CDATA[]]></typeName>

<code><![CDATA[]]></code>

<codeDescr><![CDATA[]]></codeDescr>

<char><![CDATA[]]></char>

<number />

<date />

</regAttribute>

</regAttributes>

<items>

<item><!-- can repeat -->

<id></id>

<descr></descr>

<registration-type></registration-type>

<registration-status></registration-status>

<registration-date></registration-date>

<quantity></quantity>

<attended></attended><!-- Y/N/Null -->

<regItemAttributes>

<regItemAttribute><!-- can repeat -->

<type><![CDATA[]]></type>

<typeName><![CDATA[]]></typeName>

<code><![CDATA[]]></code>

<codeDescr><![CDATA[]]></codeDescr>

<char><![CDATA[]]></char>

<number />

<date />

</regItemAttribute>

</regItemAttributes>

</item>

</items>

<guests>

<guest><!-- can repeat -->

<firstNm></firstNm>

<middleNm></middleNm>

<lastNm></lastNm>

</guest>

</guests>

</registration>

<status>SUCCESS or FAILURE</status>

<message>Element will not appear if status = SUCCESS</message>

</registrations>