Plus2 Simplified Web Service Interface

Release:	V2 p4.3	Author:	Stephen Nichols
MWS version	For Version 2.6.2186	Owner:	Gladstone Health & Leisure
Date:	26th January 2015	Reviewers:	Tom Vian

Contents

General Information 3
Document History 3
Introduction 4
Scope 4
Technical Caveat 4
Glossary 4
Authentication Methods 5
Introduction 5
IIS 5
UserId and Password 5
Host Address 5
SQL user 5
Licensing 6
Services Available 6
Bookings.asmx 8
General.asmx 11
Install.asmx 14
Login.asmx 15
Members.asmx 16
Payments.asmx 20
Subscriptions.asmx 21
Deployment 22
Secure Intranet 22
Deployment responsibilities 23

General Information

Document History

Version	Date	Author	Notes
1.0	10/07/2007	AC	Original version
1.1	21/07/2007	AC	Second Draft
1.2	22/07/2007	AC	Amended after review by GS
2.0	12/11/2009	BM	Restructured and updated to include new methods
2.1	26/01/2011	SN	Updated to include new methods
3.0	06/10/2011	TV	Updated to include the breakdown in web services and new methods
3.1	27/10/2011	TV	Added SetAllSettings & GetAllSettings and updated SetSettings & GetSettings.
3.2	30/11/2011	SN	Updated to include new methods, update method descriptions.
4.1	10/07/2012	TV	Updated to include new methods included in MWS 2.4
4.2	20/03/2013	TV	Updated to include new methods up to MWS 2.4.38
4.3	26/01/2015	UH	Updated to include new methods up to MWS 2.6.2186

Introduction

Scope

This document contains information regarding the functionality of the WSI and the authorisation methods it provides. It does not contain information about configuring IIS and using IIS to secure web-services.

Technical Caveat

The WSI is developed using .Net 3.5 and has a dependency on this. When the WSI is installed it will require .Net 3.5 to be installed if it isn't already present. In addition the WSI will only be tested and supported using IIS versions 6.0, 7.0 and 7.5

Any client wishing to use the WSI must be able to use ASP.Net Xml web services.

Glossary

WSI – Web service interface. The collection of the web methods available for accessing the Plus2 database.

IIS – Internet Information Services.

SOAP – Simple Object Access Protocol.

Remote Host – The machine that is connection to the WSI.

SQL – Structured query language.

Webmethod – A method available through a web service.

int – A .Net data type.

String – A .Net data type.

Authentication Methods

Introduction

The functionality provided can result in returning information from the Plus2 database set. Identifying that the client requesting the information has authorisation to do so is important in certain deployment situations (i.e. over the internet) to avoid malicious parties using the WSI to tamper with data.

There are several methods provided by the WSI to enhance security. It is the customer's responsibility to implement these along with any others they require. On a side note, consider that each additional layer of security will take extra time to execute - so the higher the security - the slower the WSI will perform.

IIS

It's not within the scope of this document to describe how to secure web services using IIS. However suggested methods include: SSL (https) with X509 certificates, basic authentication (username and password) and permitting access by IP addresses.

UserId and Password

An additional SOAP (simple object access protocol) header has been added to the *web methods* that are exposed by the WSI. This contains two elements userId and password, when accessing the webmethod the *userId* and *password* specified will be checked against those in a WSI configuration file. Customers that are not concerned with security or would prefer to use the IIS equivalent can disable this option in a WSI configuration file. Figure 1 below illustrates the additional SOAP header.

```
<soap:Header>
  <AuthHeader xmlns="http://www.gladstonemrm.com/memberwebservice/">
    <UserId>string</Username>
    <Password>string</Password>
    <AppName>string</AppName>
  </AuthHeader>
</soap:Header>
```

Figure 1 - Additional SOAP element

The example C#.Net code below shows how easy it is to implement this in the client:

```
WebServicereference.MemberWebService wsi = new WebServicereference.MemberWebService();

WebServiceInterface.AuthHeader authHeader = new WebServiceInterface.AuthHeader();
authHeader.UserId = "UserId";
authHeader.Password = "clientPassword";
authHeader.AppName = "ThirdParty";

wsi.AuthHeaderValue = authHeader;
```

Host Address

When connecting to the WSI the *REMOTE_HOST* in the *HttpContext server variables* will be checked against an allowed list in a WSI configuration file (can be set to enabled or disabled). Customers may prefer not to use this method if they are allowing clients to connect from unknown IP addresses or are not concerned with security.

Some of the web methods enforce that the remote host is the localhost IP, this is explained in the next section.

SQL user

Customers may wish the WSI to have its own SQL user for accessing the Plus2 database. This provides the advantage of being able to lock down the permissions that SQL user has. If malicious parties were able to bypass all the other security measures having a separate SQL user with relatively low

permissions may limit their capabilities. Additionally it provides a means for the database administrator (DBA) to monitor that SQL user in isolation.

The WSI will ship with the standard Plus2 SQL user encrypted in the WSI configuration file.

Licensing

A licence key must be installed on the webserver hosting the WSI. If no valid licence is installed, the web methods throw an exception when called. The licence key will be installed by Gladstone engineers when the WSI is installed, and will allow access to the web methods for an agreed period of time, typically a year. There are a number of different licence options available, further information can be provided if necessary.

Services Available

This section lists and describes all the web methods available. The WSI is continuously enhanced, and further web methods will be added periodically.

The general ethos is that the client application should assume that the web methods may throw an exception if there is a problem performing the task requested. For example if the Plus2 database server is being rebooted then the web methods cannot reach the Plus2 database and would throw an exception.

Every call to the web methods will be recorded in the Plus2 *event log*. This is the only available method of non-repudiation and must be accepted by the customer as an adequate method during contractual agreements. The event log isn't available through the WSI but can be reported on via the Plus2 product or the Plus2 database directly. Figure 2 shows this relationship as a use case diagram.

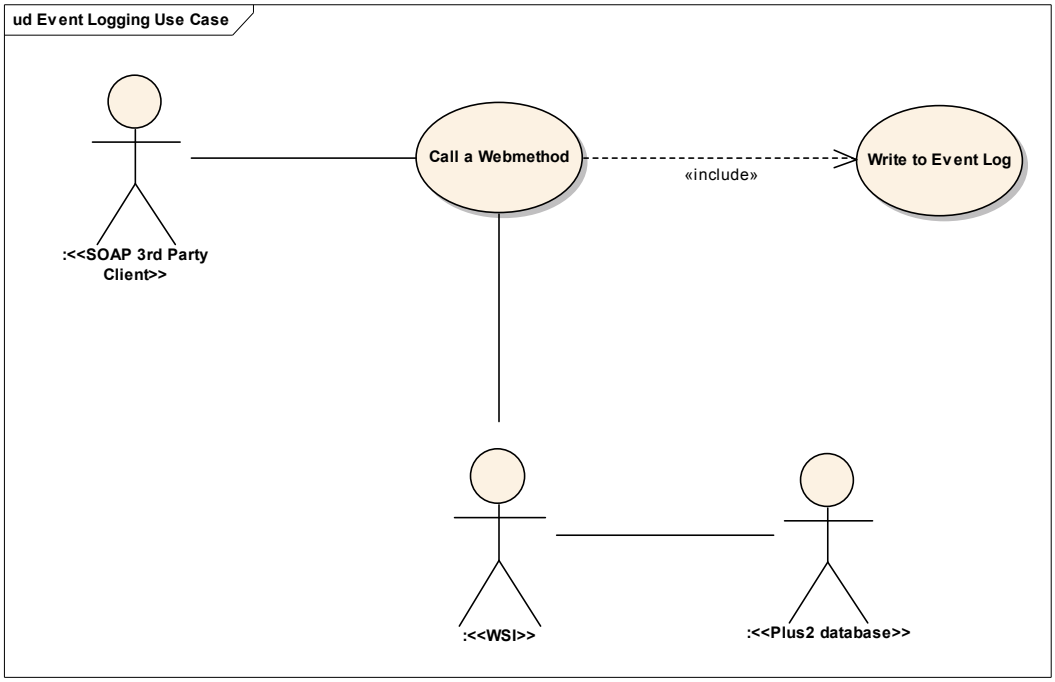

Figure 2 - Event Logging Use Case

It is expected that information about certain fields in the Plus2 database will be known by the client application. For example the default *status* value for new members and the status value for expired or 'dead' members.

The web methods are broken down into seven distinct web services:

- Bookings
- General
- Install
- Login
- Members
- Payments
- Subscriptions

Bookings.asmx

CancelBooking(int memberId, int bookingId, bool returnUsage)

This will cancel a booking for a member, if business rules allow a booking to be cancelled. If there is more than one member associated with the booking, then only the specified member is removed from the booking, and the booking remains for the other members. If the booking was paid for with a usage subscription, the parameter "returnUsage" determines whether or not the usage should be added back on to the member's subscription.

This method is due to be deprecated in favour of CancelBookingWithDetails().

CancelBookingWithDetails (int memberId, int bookingId, Boolean returnUsage)

This will cancel a booking for a member, if business rules allow a booking to be cancelled. If there is more than one member associated with the booking, then only the specified member is removed from the booking, and the booking remains for the other members. If the booking was paid for with a usage subscription, the parameter "returnUsage" determines whether or not the usage should be added back on to the member's subscription. This method provides additional reason reporting than that of CancelBooking.

GetActivitiesAtSite (obj request)

Call this method to get the details of activities, classes and courses at one or more sites.

GetActivityDetails(string activityId)

Call this method to get the details of a given activity, including the activity group it belongs to and resource products it is linked to.

GetActivityGroupDetails(obj request)

Call this method to get the details of one or more activity groups.

GetActivityGroups(string siteId, bool webBookableOnly)

Call this method to get the list of activity group information for the particular site. This info can be filtered based on the web bookable activity group only.

GetAvailableBookableSlots(obj request)

This will return an array of bookable slot with price for a given activity between two times, for a given member. If memberId is 0, then the array will return the slots available to a non-member. Does not currently support classes or courses.

GetAvailableSlots(string activityId, DateTime startDateTime, DateTime endDateTime, int memberId)

This will return an array of bookable slots for a given activity or class between two times, for a given member. If memberId is 0, then the array will return the slots available to a non-member.

GetAvailableSlotsForProduct(string activityId, DateTime startDateTime, DateTime endDateTime, int memberId, string[] productIds)

This will return an array of bookable slots for a given activity between two times, for a given member and resource product(s). If the activity requires resource products to be chosen from one or more resource product groups, specify the chosen product ids in the productIds array. This might include a member of staff or a part of the leisure centre eg Court 1. To book badminton on court 1, enter the activityId representing badminton and the resource product id representing Court 1 in the productIds array. If memberId is 0, then the array will return the slots available to a non-member.

GetBookingSheet(BookingSheet BookingSheetRequest)

Call this method to return information on activities and reservations that are booked in the plus2 Booking sheet. There are four detail levels, SummaryMergedSlots, Summary, PublicDetail and PrivateDetail, which provide a different detail of information. This call can return a lot of

information and so it is advised to work with the operator of Plus2 to understand to correct time ranges to use with this method.

GetResourceProducts(string resourceGroupId)

Call this method to retrieve list of resource product Ids for a particular resource group.

This method is due to be deprecated in favour of GetProductsByProductGroups().

GetZeroPricedAvailableSlots(string activityId, DateTime startDateTime, DateTime endDateTime, int memberId)

Returns the same results as GetAvailableSlots, but includes only the slots the member can book for free.

GetZeroPricedAvailableSlotsForProduct(string activityId, DateTime startDateTime, DateTime endDateTime, int memberId, string[] productIds)

Returns the same results as GetAvailableSlotsForProduct, but includes only the slots the member can book for free.

ListBookableActivitiesClassesCourses(int memberId,string siteId, bool webBookableOnly)

This will return an array of bookable activities, classes and courses based on the information supplied. If memberId is 0, then the array will return the activities etc available to a non-member. If siteid is an empty string then slots are returned for all sites. More than one site can be supplied by separating their ids with a comma.

ListBookableActivitiesClassesCoursesByDate(int memberId,string siteId, bool webBookableOnly, DateTime[] dates)

Call this method to return all bookable activities, classes and courses. The dates parameter will be used to filter activities which start on any date in that array. The activities with slots which are available will be returned.

ListBookableActivitiesClassesCoursesByDateAndGroup(int memberId,string siteId, string activityGroupId, bool webBookableOnly, DateTime[] dates)

Call this method to return all bookable activities, classes and courses. The dates parameter will be used to filter activities which start on any date in that array.

Only activities belonging to the specified activity group will be returned. The activities with slots which are available will be returned.

ListZeroPricedActivitiesByDateAndGroup(int memberId,string siteId, string activityGroupId, bool webBookableOnly, DateTime[] dates)

Returns the same results as ListBookableActivitiesClassesCoursesByDateAndGroup, but includes only the slots the member can book for free.

ListZeroPricedBookableActivitiesClassesCoursesByDate(int memberId,string siteId, bool webBookableOnly, DateTime[] dates)

Returns the same results as ListBookableActivitiesClassesCoursesByDate, but includes only the slots the member can book for free.

MakeBooking(string activityId, DateTime startDateTime, int memberId)

This will make a booking for a member for the given activity at the given time. The method returns the booking id for a successful booking, or 0 if the booking could not be made.

MakeBookingForProduct (string activityId, DateTime startDateTime, int memberId, string[] productIds)

This will make a booking for a member for the given activity at the given time. The method returns the booking id for a successful booking, or 0 if the booking could not be made. Use this method (rather than MakeBooking) where a product must be specified from a resource group to correctly

identify the booking. If the activity requires resource products to be chosen from one or more resource product groups, specify the chosen product ids in the productIds array. This might include a member of staff or a part of the leisure centre eg Court 1. To book badminton on court 1, enter the activityId representing badminton and the resource product id representing Court 1 in the productIds array.

MakeZeroPricedBooking(string activityId, DateTime startDateTime, int memberId)

Returns the same results as MakeBooking, but only succeeds if the member can book the activity for free at that time.

MakeZeroPricedBookingForProduct(string activityId, DateTime startDateTime, int memberId, string[] productIds)

Returns the same results as MakeBookingForProduct, but only succeeds if the member can book the activity for free at that time.

RetrieveBookingPrice(string activityId, DateTime startDateTime, int memberId)

This will return the price which would be charged to a member booking an activity at a particular time.

RetrieveBookingPriceForProduct(string activityId, DateTime startDateTime, int memberId, string[] productIds)

This will return the price which would be charged to a member booking an activity at a particular time. Use this method (rather than RetrieveBookingPrice) where a product must be specified from a resource group to correctly identify the booking. If the activity requires resource products to be chosen from one or more resource product groups, specify the chosen product ids in the productIds array. This might include a member of staff or a part of the leisure centre eg Court 1. To book badminton on court 1, enter the activityId representing badminton and the resource product id representing Court 1 in the productIds array.

General.asmx

CalculateSpecialDate(obj request)

This method returns a calculated date. It calculates the date value for the given Userfield of type 'Date' configured with special date function.

ClearSettingsCache()

Clears the server settings cache, forcing it to be fetched from DB next read.

GetAllSettings()

This method returns a WebServiceConfiguration object that is populated with the configuration values related to the applicationId defined in the WSI config file. These values are populated by default during install.

The WebServiceConfiguration object contains different objects to hold values for configuration related properties. These objects include PCID, Setting, BookableActivies, EventLog, MemberRecordAccess, MemberEditRecordAccess, MemberRecordCreateRequiredParts, MaxLength, LengthException, BookingConfiguration and AllowedLoginAttempts.

GetAttendanceByProductsGroup(string[] SiteIDs, string[] ProductGroups, string[] ExcludedDepartmentIds, DateTime StartDate, DateTime EndDate, Boolean IncludeNonMember, String UserfieldDescription)

Returns Sales details, for the given criteria, of when and where certain products were sold in order to provide analysis of attendance.

GetCorrespondenceCategories()

Returns a list of correspondence categories present in the Plus2 database.

GetEntryPointList(obj request)

This method returns details of Entry Points configured in the system. If the siteId is provided then it will return site specific Entry Point details whereas siteId as null will return Entry Point details for all sites.

GetLocalDateTimeForSite(string siteId)

Returns the current time on the database server, adjusted for the time difference between the database server's location and the site's location.

GetPriceLevelList()

Call this method to return a list of the PriceLevel objects present in the Plus2 database.

GetPriceLevels()

Call this method to return an array of all the available price level ids in the Plus2 database

GetProductGroupsByProducts(obj request)

This method returns the product groups for a given array of products.

GetProductsByProductGroups(obj request)

This method returns the products for a given array of product groups.

GetProductDetails(obj request)

Call this method to return details of a given product ID.

GetResourcesBySiteGroup(obj SiteGroupIds)

Returns an array of resources for given site groups. Resources usually define physical areas configured in the system e.g. a sports hall or a badminton court.

GetSettings()

This returns an array of Setting objects relating to the WSI from the Plus2 database. Each Setting object has a Name and a set of value parameters – StringValue, LongValue, DateTimeValue, DecimalValue. Each Setting will have only one of these values populated. The initial release of this method supports two settings – FutureBookingDays (LongValue), ClubDescription (StringValue). The results are dependent on the value of applicationId tag.This value is inserted into the database using SetSetting() and is populated with default values at the time of installation.

GetSiteDetails()

Call this method to return an array of details of the sites in the Plus2 database. These include site ID, Description, Latitude and Longitude.

GetSiteGroupsBySiteID(obj request)

This method returns an array of SiteGroups based on a given SiteId.

GetSiteIDs()

Call this method to return an array of the Site ids in the Plus2 database.

GetStatusList()

Call this method to return a list of the Status objects present in the Plus2 database.

GetStatuses()

Call this method to return an array of the Status ids present in the Plus2 database.

GetSubscriptionTypeGroups()

Call this method to return a list of all Subscription Type Groups in the Plus2 database.

GetSubscriptionTypes(List<string> subscriptionTypeGroupIDs)

Call this method to return a list of all Subscription Types in a given set of Subscription Type Groups in the Plus2 database.

GetSystemConfigurationList(string siteId, string id)

Call this method to return an array of the System Configuration objects present in the Plus2 database for a given site and system configuration id.

GetTitleList()

Call this method to return a list of the TitleDetail objects present in the Plus2 database. This includes the title description and gender of the title.

GetTitles()

Call this method to return an array of all the available titles in the Plus2 database. This should be used in conjunction with the CreateMember and Edit Member web methods to ensure a valid title is selected.

GetUserFields()

Call this method to return an array of the user fields in the Plus2 databaseSetAllSettings

SetAllSetting(WebServiceConfiguration configurationObject)

Before calling this method, call GetAllSettings to retrieve the current WebServiceConfiguration object from the Plus2 database. Edit values as required, then call SetAllSettings to save the changes back to the Plus2 database.

SetSettings(List<Setting> settings)

Before calling this method, call GetSettings to retrieve the current list of Setting objects from the Plus2 database. Edit values as required, then call SetSettings to save the changes back to the Plus2 database.

WriteEventLog(obj requestParameters)

Use this method to update a member's event log.

Install.asmx

The methods in this webservice do not require a licence or a valid username or password.
All the methods in this webservice can be invoked from a web browser at the URL http://localhost/<WebServiceApplicationName>/Install.asmx

GetIP()

This method returns the IP address of the connecting request. This method can be used to identify the client's IP address, so that it can be included in the WSI configuration file to allow access only to a limited set of client machines.

GetMwsUserLastActivityDate()

This method shows the time and date that MWS usernames last accessed the system. This can only be run from localhost and is intended to help provide information on the users and systems using MWS.

GetSystemInfo()

This method returns the current versions of MWS and Core stored proceedures. This requires connection to the database and will provide an analysis as to a whether MWS is installed correctly. This method can only be ran via localhost.

Login.asmx

CleanupUserChanges (string systemName)

Used in conjunction with Integration Manager. Call this method to delete committed user change records for a given system. Call this method infrequently – typically once a day, at a quiet time. Running it could cause database blocking.

CommitUserChanges (string systemName, int runId)

Used in conjunction with Integration Manager. Call this method to mark as committed user update records for a given system and run id. Call this method after calling GetUserChanges and successfully storing those updated records in a third-party system. If this method is not run, the next time GetUserChanges is run, it will include the same records as previously, plus any additional changed records.

GetUserChanges (string systemName, int lastRunId, ref int runId)

Used in conjunction with Integration Manager. Call this method to return a list of all users whose records in Plus2 have been edited since the last time this method was run for the given system, and since the last run with the given id. The current run is returned in the runId parameter. After these updates have been recorded in your third-party system, call CommitUserChanges, passing in the same systemName and runId. All users are returned until the first run of CommitUserChanges, thereafter only added, changed or deleted users are provided.

GetUserDetailById (string id)

Call this method to get the Plus2 user details for a given user Id.

LoginMember (IdentificationType identificationType, string identificationValue, ValidationType validationType, string validationValue, string defaultSite)

Call this method to validate the member login details and creates a login token that can be used by other applications for auto login. This method will call **ValidateMember()** internally. It returns the login result object that includes the login token information.

LoginUser (obj request)

Call this method to validate the member login details from security service and return a login result.

RetrieveUser (obj request)

Call this method to retrieve the user details from Plus2 or security service for a given user id.

ValidateMember (IdentificationType identificationType, string identificationValue, ValidationType validationType, string validationValue, string defaultSite)

Call this method to validate the member login details. It returns the login result information object.

ValidateUser (string UserID, string encryptedPassword, string[]accessGroupIds)

This method validates the user login details and returns a login result.

Members.asmx

AddAccountDetails(Account accountDetails)

This method enables adding, modifying and deleting members' accounts.

AddCorrespondence(obj Request)

This method adds a new correspondence record for the given member.

AddPicture(int MemberId, string base64EncodedImage)

Add a picture to the Plus2 database for a given memberID.

AmendLoyalty(int memberID, int loyaltyAmountChange)

Call this to amend the loyalty points of a member's record in the Plus2 database. The amendAmount can be positive or negative within the bounds of an int. The client must already know the memberID to use this webmethod.

CheckMember(string firstname, string lastname, DateTime birthDate)

Use CheckMember to find out if an existing member on the customer's system without a memberID exists within the Plus2 database. If they do their memberID will be returned and be aware of a possible limitation of this method - if there are two (or more) records in the Plus2 database with the same firstname, lastname and birthDate then an exception will be thrown. There isn't a way to determine the correct record without compromising the data of the other member. This member's memberID must be obtained by a manual business process rather than an automated one.

CheckMemberEmail(string email)

Use checkMemberEmail to get an array of members that use that email as either their work or home email.

CheckMemberID(int memberId)

Call this method to determine if a member record exists for the given id.

CheckMemberLogin(LoginIdentificationType LoginId, string idValue, LoginValidationType loginValidationType, string validationValue)

Call this method to check the member login information. If it is matched then it returns the member id. 0 will be returned if the member login is invalid or the member does not exist in the database.

CleanupMemberChanges(string systemName)

Used in conjunction with Integration Manager. Call this method to delete committed member change records for a given system. Call this method infrequently – typically once a day, at a quiet time. Running it could cause database blocking.

CommitMemberChanges(string systemName, int runId)

Used in conjunction with Integration Manager. Call this method to mark as committed member update records for a given system and run id. Call this method after calling GetMemberChanges and successfully storing those updated records in a third-party system. If this method is not run, the next time GetMemberChanges is run, it will include the same records as previously, plus any additional changed records.

CreateMember(Member newMember)

This webmethod will insert a new member into the Plus2 database. This will occur when a member is present on the customer's other system but without a memberID in their record. CreateMember calls CheckMember first in order to see whether a duplicate member exists. If so, CreateMember returns -1. If successful, the member id of the newly created member is returned.

CreateOrUpdateMember(Member inputMember)

This method provides access to either member creation or member updating, the decision as to which is appropriate will be based on the input member if the memberid is already known it will be quicker to use editmember if the memberid is not known submit the input member with and id of 0 and the code will decide if a update or insert is needed based on first name, last name and date of birth

CreateValidMember(Member newMember, bool isDuplicateAllowed)

This method works in the same way as CreateMember, but the duplicate checking is extended. If isDuplicateAllowed is true, a member can be created even if another member in the database matches certain key fields. If isDuplicateAllowed is false, a member can be created only if no duplicate records are found when searching by first name, last name, login id, date of birth, email address and postcode (if these fields are entered).

EditMember(Member updatedMember)

This web method should be used if a member in the customer's system has some details updated (e.g. change of address). If the member doesn't have a memberID Checkmember can be used to get it. Then either a blank Member object can be populated or RetrieveMember can be used to return a populated Member object that can be amended and returned to EditMember.

EditMember can be used to update the member's cardID, for example if their old card was lost or stolen then a flag will be available in the Member object to indicate whether the old cardID should be *hotlisted*.

GetAttendanceHistory(int memberId, DateTime startDateTime, DateTime endDateTime, string siteId, int rowCount)

This will return an array of the attendance records for a member between the startDateTime and endDateTime. If siteId is specified, only records for that site are returned. The number of rows returned is limited in size by the rowCount parameter

GetCorrespondence(obj FilterDetails)

Call this method to retrieve correspondence from a given set of members. The response will contain a number of correspondence objects, depending on the data stored in the system, with a SuccessFailureWithMessage object.

GetMemberBookings(int memberId, bool futureOnly)

This will return an array of the bookings made by a member. If futureOnly is true, only bookings starting in the future will be returned.

GetBookingsWithProducts(int memberId, DateTime earliestStart)

Returns an array of bookings a member has made, including any booked activity's resource product groups. If futureOnly is true, only bookings starting in the future will be returned.

GetMemberChanges(string systemName, int lastRunId, ref int runId)

Used in conjunction with Integration Manager. Call this method to return a list of all members whose records in Plus2 have been edited since the last time this method was run for the given system, and since the last run with the given id. The current run is returned in the runId parameter. After these updates have been recorded in your third-party system, call CommitMemberChanges, passing in the same systemName and runId.

GetMemberChangesPaged(string SystemName, int LastRunID, int PageSize)

Similar to GetMemberChanges, but allows the calling application to specify page size. A page size of 500 is suggested.

GetMemberPicture(int memberId)

Get a picture from the Plus2 pictures database for a given memberID.

GetMemberRelationList(string memberId, string sort, int rowIndex, int maxPageCount, out int totalCount)

Call this method to determine if a member in the Plus2 database is linked to any other member.

GetMemberSearch(string firstName, string lastName, string emailAddress, string homeTelephone, string mobileTelephone, string workTelephone, string postcode, int memberId, DateTime birthDate, string sort, int rowIndex, int maxPageCount, out int totalCount)

Call this method to determine if a member record exists in the Plus2 database for the details specified.

GetMemberSubscriptions(int memberId)

Call this method to return an array of the subscriptions held by a member.

GetScreenProfileList(string profileId)

Return a list of required fields for a given Screen Profile in the Plus2 database.

GetTemplateMember()

Call this method to return a valid template member that you can then amend and use when calling CreateMember.

MemberSearchByCardID(string cardID)

Search for a member by their card id. The method returns 0 if there is no match or 1 if there is a match and throws an exception if multiple records are found.

MemberSearchByUserField(string userFieldName, string UserFieldValue)

Find a member by a user field name and corresponding value.

RecordAttendance(obj request)

This method will record an attendance for a given member. An attendance can be recorded against either a booking or an entry point. A SuccessFailureWithMessage is returned.

RegisterMemberForThirdPartySystem(obj request)

This method allows members to be register against a given 3rd party system. The results of GetMemberChanges or GetMemberChangesPaged will be filtered by the given 3rd party system and will return updates for these members.

RetrieveLoyalty(int memberID)

Call this with memberID as a parameter to return the loyalty points for the member in the Plus2 database.

RetrieveMember(int memberID)

This webmethod should be called when the client wishes to retrieve a member record from the Plus2 database. A Member object (class available through WSI) will be returned. This is likely to be used is in conjunction with EditMember. The Plus2 memberID should be passed as a parameter.

UnHotlistCard(string cardID)

This webmethod should be called when the client wishes to remove a card from the hotlist of invalid cards, stored within the Plus2 database.

UnRegisterMemberForThirdPartySystem(obj request)

This method allows members to be unregistered from a given 3rd party system. The results of GetMemberChanges or GetMemberChangesPaged will be filtered by the given 3rd party system will no longer return updates for these members.

VerifyMember(MemberFilter filters)

Call this method to determine if the member already exists in the DB and return all members with matching records against the supplied parameters.

Payments.asmx

AddBillingAccount(obj request)
This method inserts or updates a member's billing account.

AddSubHolderBillingAccount(obj request)
This method inserts, updates or deletes a sub-member's billing account.

CreditMember(obj request)
This method allows specified sales and payments to be credited back to a member's account.

GetMemberBillingAccountSummary(obj request)
This method retrieves information for a member's billing account.

GetMembersInvoiceDetails(obj request)
This method enables a member's invoices to be retrieved according to given search criteria.

GetNextOnlineAuthID (string paymentSiteID)
Get the next sequential id to be used in the PayInvoice method.

GetUnpaidDebt (int[] memberIds, bool includeLinkedMembers)
This webmethod returns all unpaid invoices for the array of member ids selected, and (if required) their linked members. The invoices can then be marked as paid using the PayInvoice method.

PayInvoice (int salesInvoiceId, int memberId, int onlineAuthId, string paymentSiteId)
Call this method to mark a Plus2 sales invoice as paid for a member.

This method is due to be deprecated in favour of PayInvoices().

PayInvoices (int[] salesInvoiceIds, int memberId, int onlineAuthId, string paymentSiteId)
Call this method to mark a Plus2 sales invoice as paid for a member. Allows a list of sales invoices to be paid in one call.

PrepareSalesInvicePayment(obj request)
Call this method when integrating with Payment Manager. The method returns a URL within Payment Manager for completing payment of sales invoices.

RetrieveProductPrice(obj request)
This method gets the prices for a number of specified products for a given member.

SellProductToMember(obj request)
This method raises a sale for a number of specified products for a given member.

TopUpMembersBillingAccount(obj request)
This method enables top ups to be applied to a member's cashless billing account.

Subscriptions.asmx

AssignSubTypeToMember(int memberId, string subTypeId, DateTime startDate, string siteId)

Call this to assign a subscription to a member in the Plus2 database. An unpaid debt is left on the member's account in Plus2. If payment has been received you can then call SetMemberSubPaid.

AssignSubTypeToMemberDate(int memberId, string subTypeId, DateTime startDate, string siteId, DateTime earliestEndDate)

This method behaves exactly as AssignSubTypeToMember, but allows the Plus2 Earliest End Date to be specified.

AssignSubTypeToMemberWithDetails(obj requestDetails)

This method allocates an unpaid Subscription to a member from the given start date. The earliest end date and renewal stop date can also be set. It returns the relevant Subscription and Invoice Details.

DecrementUsagesOnSub(int memberId, int subscriptionRef, int usageCount, string transactionId)

Call this method to decrement the number of usages for a given subscription and member by a certain amount, returning the number of usages remaining. You may use this method to increment the usages on a subscription if WSI configuration allows it. The transactionId must be a unique id which cannot be re-used. This is designed to prevent the webmethod being called repeatedly to represent a single transaction.

GetAllAvailableSubs(int memberId, string siteId, bool webBookableOnly)

Call this to retrieve an array of SubscriptionType objects available to be purchased. Set memberId as 0 to see the subscriptions which non-members can purchase. SiteId can be a single site or comma-separated list of site ids.

This method is due to be deprecated in favour of GetAvailableSubsByStartDate().

GetAvailableSubsByStartDate(obj request)

This method give the same results as GetAllAvailableSubs with additional price information and next collection date.

SetMemberSubPaid(int memberId, string subscriptionRef, bool payAll, int externalAuditId)

Call this to set a subscription to the paid status in the Plus2 database. If you set pay all to true it will pay off all the invoices assigned to this subscription. If this is false it will pay off only those that are due.

StopSubscriptionRenewal()

This method sets the renewalStopDate allowing start of day procedures to correctly stop the subscription. It is recommended that the StopSubscriptionRenewal request parameter AdhereToEarliestEndDate is set to true. It returns a SuccessFailureWithMessage

Deployment

The deployment can take two main forms - from a secure intranet - to an internet facing solution through a DMZ. The key features of both are:

The WSI server must have IIS version 6.0 or higher.

The WSI server must be able to access the database server via port 1433.

The WSI will be running in ASP.net 2.0 with .Net Framework 3.5 installed.

The WSI does not necessarily need a dedicated server but this will be recommended.

Secure Intranet

Figure 3 shows a secure intranet deployment. All elements are on the same local area network (LAN). The SOAP client can reach the customers other database system (e.g. smart card system).

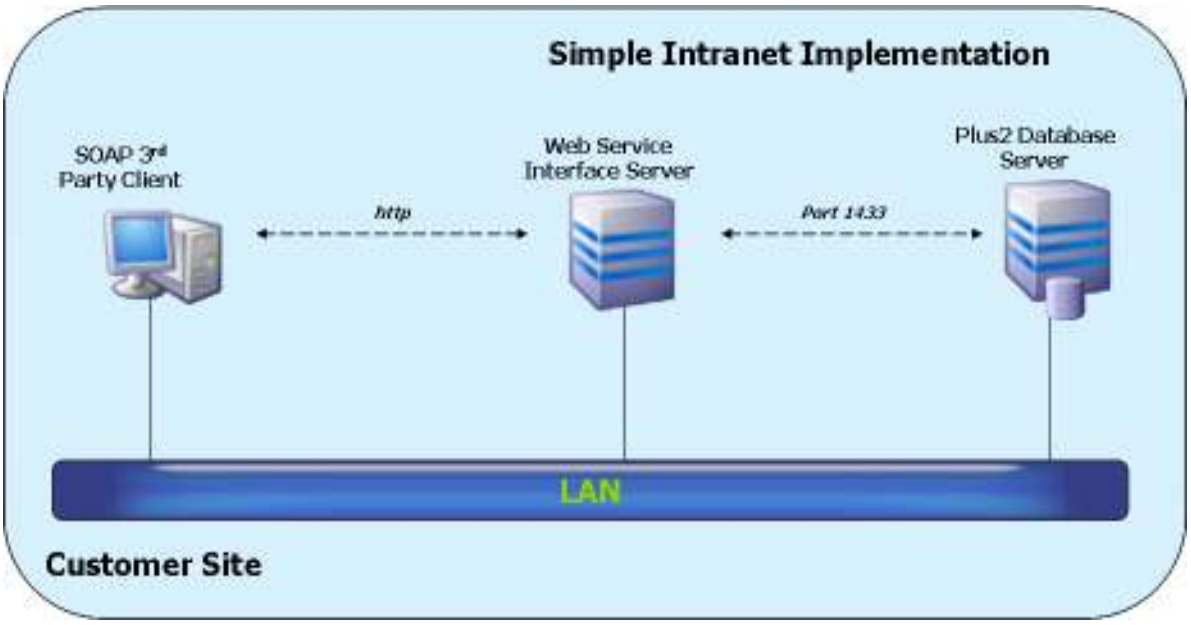

Figure 3 - Simple Intranet Deployment

Internet Facing

Figure 4 shows an internet facing deployment. The WSI server is not on the same LAN as the Plus2 database server; it accesses it though the more secure DMZ. Again - the SOAP client can reach the customers other database system (e.g. smart card system) as required.

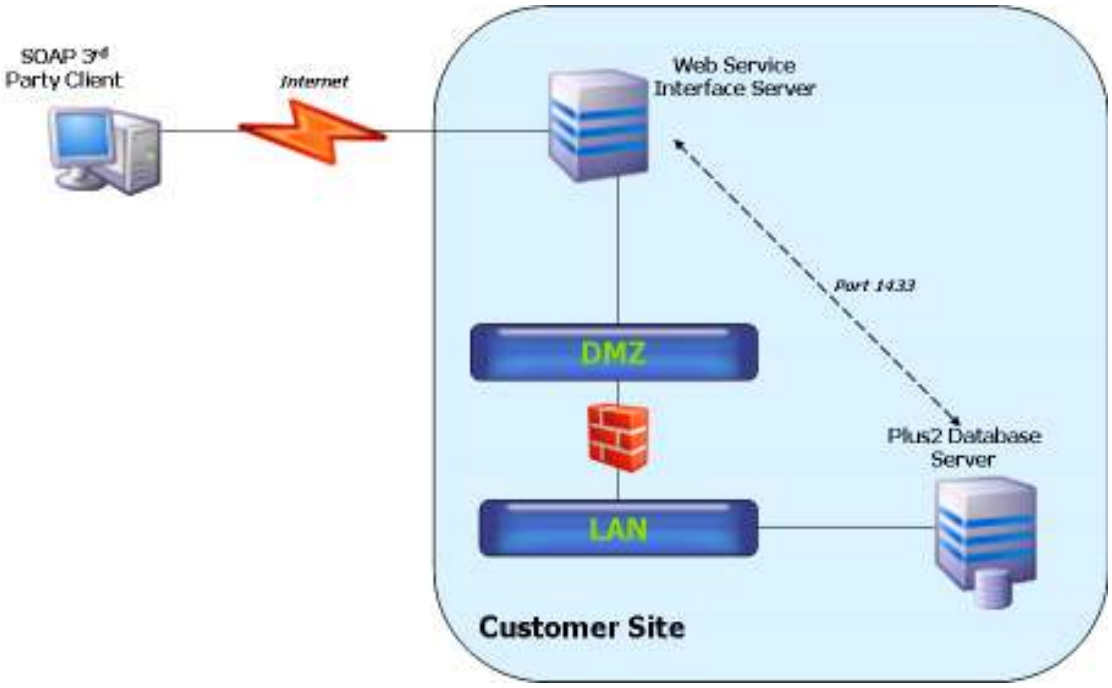

Figure 4 - Internet Facing Deployment

Deployment responsibilities

Gladstone engineers will be responsible for installing the WSI on the customer's environmentlocal. They will also be responsible for ensuring the WSI is working (via a simple test client on localhost).

The customer will be responsible for: advanced IIS configuration (e.g. SSL) and ensuring the WSI works across the internet by configuring their firewall etc.