



Welcome to Microsoft Live@edu

Building Live@edu Services into a School Portal

The information contained in this document relates to pre-release software products and services that may be substantially modified before its first commercial release. Accordingly, the information may not accurately describe or reflect the software product when first commercially released. THIS GUIDE IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY AND MICROSOFT CORP. MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS GUIDE OR THE INFORMATION CONTAINED IN IT.

© 2010 Microsoft Corp. All rights reserved.

Table of Contents

Executive Summary.....	4
A Connected Campus.....	5
Portal features considerations.....	6
Authentication	6
Mail and Calendar Web Parts	8
IT Admin and support staff tasks	9
Demo Portal Features	11
AD Authentication.....	11
Live Services Authentication using SSO toolkit.....	12
Seamless experiences EWS Managed API.....	14
Overview of EWS Managed API	14
Autodiscover Service.....	16
Impersonation.....	17
Web Parts.....	17
Mail WebPart	18
Calendar WebPart.....	19
Adding Events to a Student’s Calendar.....	20
Integration with PowerShell	22
Overview of PowerShell.....	22
Working with C# Outlook Live wrapper	22
Summary	25
Glossary of Acronyms	26
References	27

Executive Summary

The Live@edu offering from Microsoft provides a hosted e-mail solution and access to other online tools available for schools to create a “connected campus” environment. The primary advantages of this offering are:

- Provide a co-branded hosted Microsoft Exchange solution at no cost with Microsoft Outlook Live.
- Prepare your students for the real world with Microsoft tools.
- Help keep your students’ data private and promote online safety.
- Online storage on Windows Live SkyDrive to help students manage their studies, and share and store information.
- Simplify online collaboration and document sharing with Office Live Workspace.
- Give your university a reliable and easy-to-manage Microsoft solution with enhanced security.

In this paper we will introduce you to the various SDKs available that will help you build a comprehensive school portal that combines the school’s existing applications and services with the various services provided by Live@edu.

A Connected Campus

Live@edu is a flexible platform offering education institutions a rich set of familiar web-based applications that are regularly upgraded and added to, and hosted on Microsoft's reliable infrastructure; these services help relieve education institutions of the burden and cost of infrastructure maintenance and design. The platform consists of the capability for schools to provision Windows Live IDs and get a hosted email solution based on Outlook Live using a school's existing domain space. Windows Live IDs also enable students to access a variety of other services that allow them to collaborate with other students in real-time, and stay connected more easily with friends and family on campus and after they graduate.

In a connected campus environment, the various Outlook Live services are combined with the school portal for a seamless experience for users. We have developed a school portal to incorporate and access various Live Services seamlessly.

The school portal was designed taking three primary end users in an education institution into consideration. The first end user is a student who has his email mailbox setup in Outlook Live and who accesses other services like SkyDrive. The second end user is a professor who has his email mailbox setup on an on-premise Exchange Server. The third end user is administrator who performs all administrative tasks using Outlook Live. The portal has been designed to allow each user to easily access it for his or her specific needs.

Portal features considerations

We focused on three key areas to demonstrate integrating Live Service into the school portal:

- Authentication for users to access information and for personalization
- Email and calendar web parts for users to view their latest messages and schedule for the day
- Common IT administrator and support staff tasks

Authentication

A school portal will have some form of authentication for users to access information. It can be custom authentication, authentication using some Directory Server, or another method. In the school portal, Active Directory authentication is used to authenticate users. The SSO toolkit is used to access Live Services once the user is authenticated. This demonstrates how you can give an integrated experience to the users (Student, Faculty and Admin). The SSO toolkit is SDK provided by Microsoft Corporation to access Live Services bypassing Windows Live Login. Later, we will discuss in detail on how to use SSO tool kit to authenticate users to Live Services (Outlook Live, SkyDrive, etc.).

A screen shot of the school authentication page/ login page using AD authentication is shown below.

CONTOSO UNIVERSITY

SCHOOLS OFFICES ABOUT CONTOSO ADMISSIONS & AID SEARCH

Welcome to Contoso University

With a rich history and a dedication to the pursuit of excellence, Contoso University offers unique learning experiences across a broad spectrum of academic and social environments.

[More about CU](#)

EVENTS

May 15 Steve Ballmer Speaks to Law School

May 18 Rick Steves on Campus

[Event Calendar](#)

UNIVERSITY NEWS

Summer Graduation Ceremony 2009
The graduation ceremony for full-time and evening graduate students is scheduled for Saturday, June 13, 2009 at 7:00pm. You can find information about the ceremony on the Graduation website.

Legend's Breakfast Series Rolls Out for 2009
Come meet the some highly successful luminaries in both business and the arts and hear them speak. Please visit the Breakfast Series website for more information on the schedule and dates for the

[Capital Markets Discussion...](#)

SIGN IN

User Name
lisa_john

Password

[Sign In](#) [Sign Up](#)

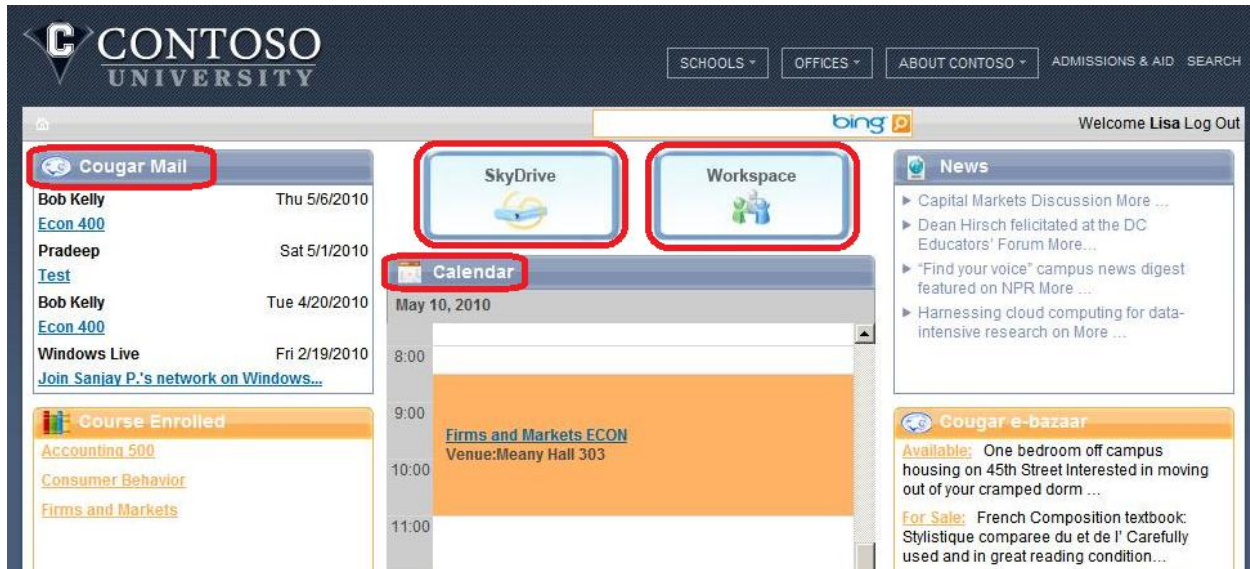
Gateways for..

[Faculty & Staff](#)

[Parents](#)

[Students](#)

After the user is authenticated into the school portal using AD authentication, SSO toolkit is used to login the user into Live Services. Below is a screenshot of the student home page where live services are integrated into the page. Check the highlighted areas for examples of live services.



The following live services can be accessed from the page:

- Outlook Live
- SkyDrive
- Workspace

Mail and Calendar Web Parts

Capturing a quick snapshot of user's Mail and Calendar data in a user's personal page in a school portal is a common and desirable functionality. We have built Mail and Calendar Web Parts for reading and rendering Mail and Calendar information from Outlook Live/ On-Premise Exchange Server using EWS Managed API. EWS Managed API is SDK provided by Microsoft to access Exchange resources (Mailbox, Calendar, etc.) from Outlook Live. We will get into more details in the following sections on how to build a Web Part and use EWS Managed API to access Mail and Calendar data from Outlook Live.

Below is a screenshot of a student home page rendering mail and calendar data of the student using the EWS Managed API.

The screenshot displays a student's personal page on the Contoso University portal. The page features a navigation bar with links for 'SCHOOLS', 'OFFICES', 'ABOUT CONTOSO', 'ADMISSIONS & AID', and 'SEARCH'. The main content area is divided into several sections:

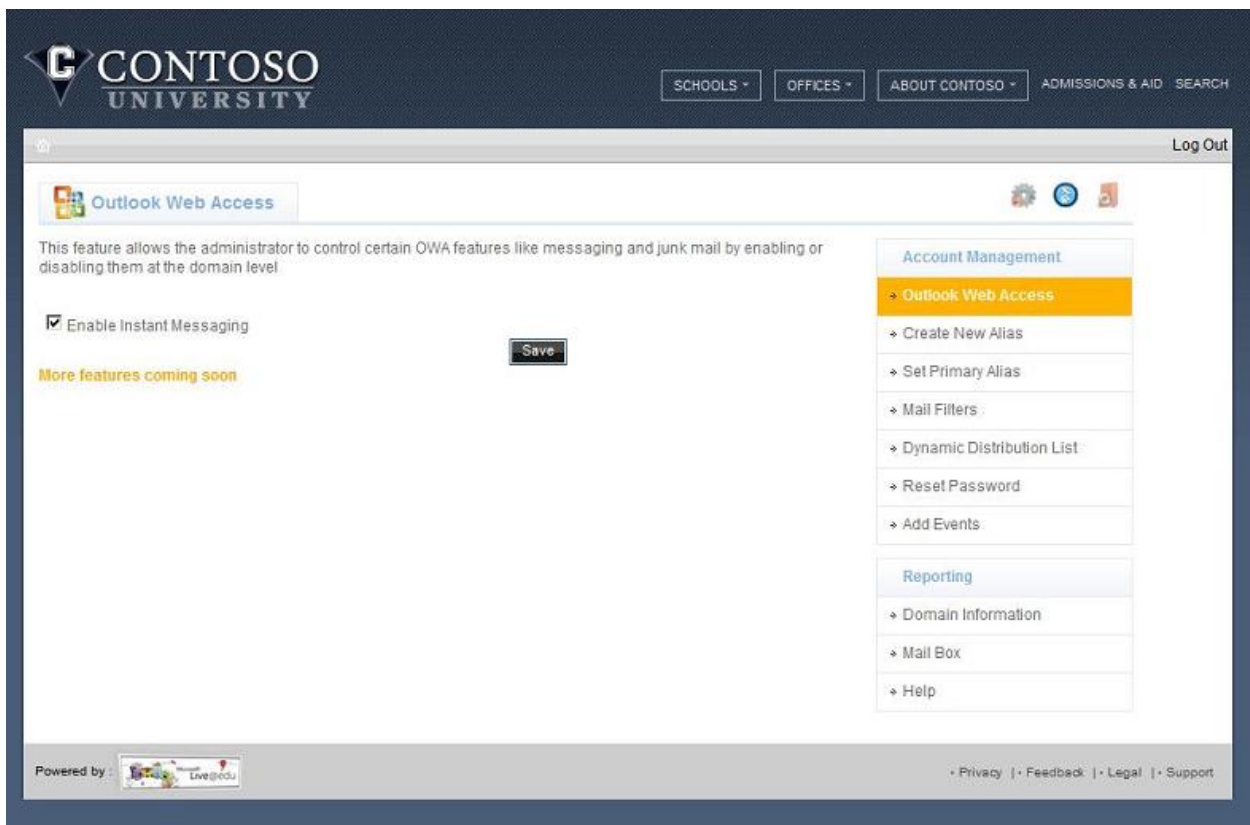
- Cougar Mail:** A list of five email items, including messages from Bob Kelly, Pradeep, and Windows Live, with dates ranging from 5/6/2010 to 2/19/2010.
- Course Enrolled:** A list of three courses: Accounting 500, Consumer Behavior, and Firms and Markets.
- Cougars Events Calendar:** A list of three events: 'Responding to Hate Crime' (8:00PM-7:00PM), 'Director's Duties in a Time' (7:00PM-8:00PM), and 'Annual Ethics Seminar' (8:00PM-9:00PM).
- Calendar:** A daily calendar view for May 10, 2010, showing two events: 'Firms and Markets ECON' (9:00-11:00 AM, Venue: Meany Hall 303) and 'Consumer Behavior MARKETING' (13:00-14:00, Venue: Balmer 413).
- News:** A list of four news items, including 'Capital Markets Discussion', 'Dean Hirsch felicitated at the DC Educators' Forum', and 'Hamessing cloud computing for data-intensive research'.
- Cougar e-bazaar:** A section with three items: 'Available: One bedroom off campus housing on 45th Street', 'For Sale: French Composition textbook', and 'Free free free: Cougar basketball tickets available'.
- Cougar Happenings:** A list of three events: 'Marketing Club offers free pizza for your ideas', 'Rick Steves discusses Iran trip', and 'Microsoft CEO Steve Ballmer addresses Law Students'.

The page is powered by Live@edu and includes links for Privacy, Feedback, Legal, and Support.

IT Admin and support staff tasks

It is important for administrators and IT support staff to be able to perform regular tasks/ common user requests (password resets, creating new Dynamic Distribution List, etc.) through the school admin portal. To demonstrate this requirement, we built an admin portal, Outlook Live PowerShell, to perform various admin related tasks on Outlook Live. In this document, we will discuss how the Outlook Live PowerShell wrappers are used. How we built the Outlook Live PowerShell wrappers is discussed in a different whitepaper available at <http://code.msdn.microsoft.com/psoutlooklive/>.

Below is a screenshot of an admin page where common tasks can be performed on the Outlook Live managed domain.



The architecture of the system and how all components interact with each other is expressed in Figure 1 below. School portals may have many specific distinguished features such as registration, enrollment systems, etc., but those are not related to Live@edu and therefore not discussed in detail.

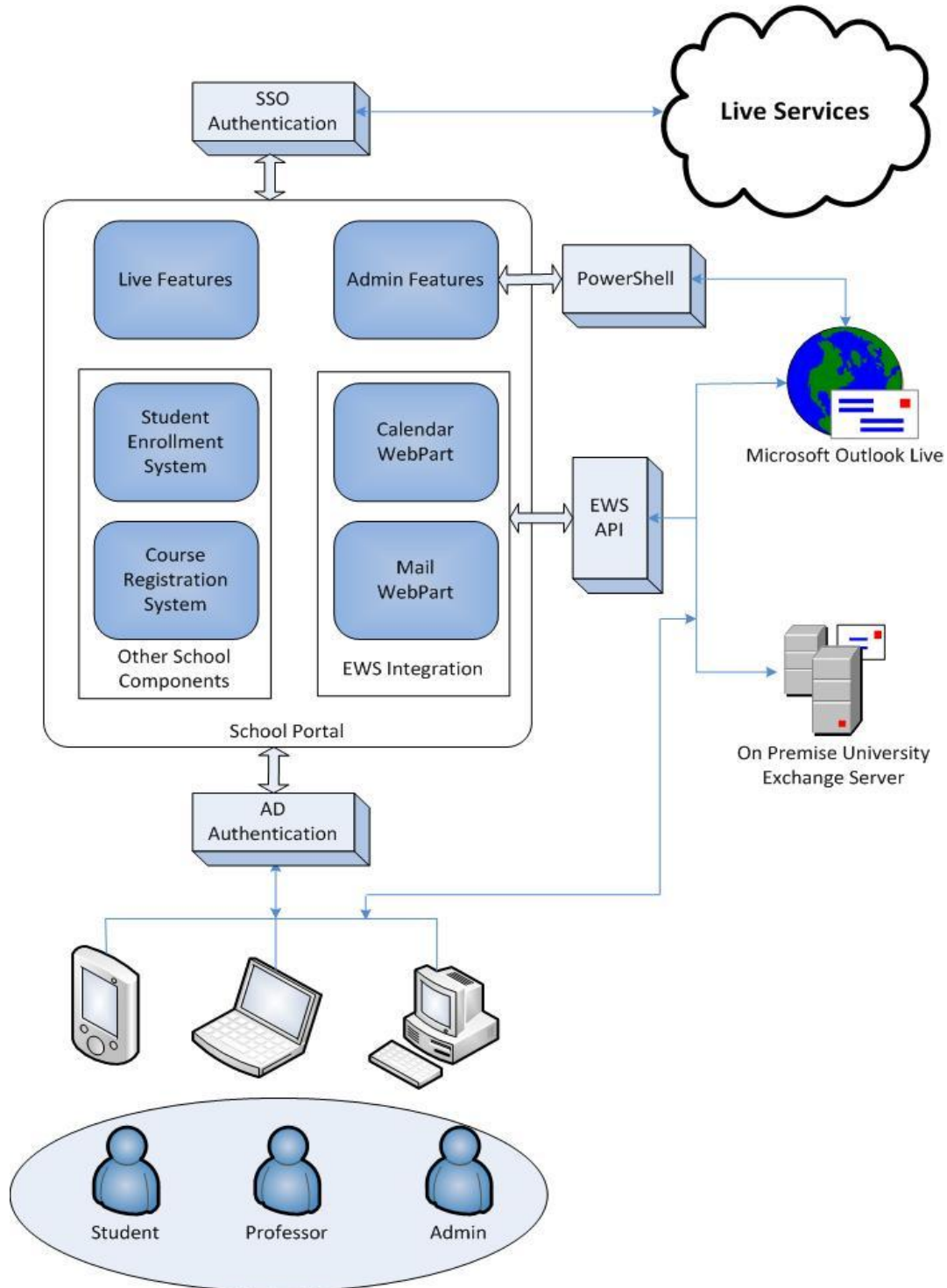


Figure 1: School Portal Overview

Demo Portal Features

This section covers various features in the school portal and their implementation details. The user is authenticated in to the portal using Active Directory (AD) authentication. Once authenticated, the user will be redirected to the correct page based on their user status (student/ faculty/ administrator).

As a student, the user will be able to access Live@edu features such as SkyDrive without the need to login to Windows Live. This is achieved through SSO (Single Sign On) toolkit. Also, the student and the faculty will be able to view their mail and calendar information in the portal. EWS API's are used for accessing mail and calendar information from Outlook. Using SSO toolkit, users can be redirected to Outlook Live from the school portal without needing to sign in through Windows Live Login. Outlook Live PowerShell wrappers will allow an admin to perform administrative tasks on the domain.

AD Authentication

In this section, we will discuss how to authenticate users against Active Directory using Lightweight Directory Access Protocol (LDAP) and retrieve a user's Windows Live Id from the Active Directory Object.

As mentioned earlier, the portal we developed was designed to use AD authentication. This is to show how a school can have any form of authentication to authenticate and authorize users to the portal and then use SSO toolkit to authenticate to Live@edu services. We designed the school portal application to store Windows Live Id in AD user object's "ExtensionAttribute2". This will be input to SSO toolkit which will be discussed in the following section. Note that Windows Live Id can be stored and retrieved from anywhere as per your requirement.

Below is code snippet to authenticate users against AD and retrieve a user's Windows Live Id from the AD object.

```
string samAccountName = "lisa.andrews";
string adLdapPath =
"LDAP://<LDAPServer>/OU=contosouniversity.net,DC=contosouniversity,DC=local";
// Create DirectoryEntry object to authenticate user and retrieve user
// info.
DirectoryEntry directoryEntry = newDirectoryEntry(
    adLdapPath, samAccountName, "lisa.andrews");

// Bind to the native AdsObject to force authentication.
object obj = directoryEntry.NativeObject;

// Retrieve Windows Live Id from CustomAttribute2.
DirectorySearcher search = newDirectorySearcher(directoryEntry);
search.Filter = "(SAMAccountName=" + samAccountName + ")";
search.PropertiesToLoad.Add("customAttribute2");
SearchResult result = search.FindOne();
string windowsLiveId =
result.Properties["customAttribute2"][0].ToString();;
```

In the above code snippet, a **DirectoryEntry** object is created with the directory tree path, user name and password. **DirectoryEntry** object is used to bind to the native AdsObject to force authentication. A **DirectorySearcher** object is created to retrieve AD object properties. In this code snippet, **CustomAttribute2** is retrieved to get a user's Windows Live Id.

In the code published for portal at <http://code.msdn.microsoft.com/liveateduexplore>, refer to UserState.cs for AD authentication implementation.

Live Services Authentication using SSO toolkit

In this section, we will discuss SSO toolkit and how it is used to access live services without needing to enter in a user ID and password.

SSO toolkit is an SDK provided by Microsoft to access Live Services bypassing Windows Live Login. This solution is agnostic to university authentication infrastructure (e.g., Active Directory, LDAP, ADAM, etc.). Input to SSO toolkit is Windows Live ID and WLID-issued certificate. Before SSO toolkit can be used, your domain registered with Live@edu should be enabled for SSO.

Authenticating users to LiveServices is a two-step process.

- Get Short Lived Token (SLT)
- Use SLT to get re-direction URL

Below is code snippet to authenticate a user to Live Services.

```
// Thumb print of the SSO certificate.
string certThumbPrint = "aa aa ...";
// SSO Site ID. A unique ID is created when you request for SSO.
string siteId = "1111";
// Proxy server if the computer requires it for internet out.
string webProxy = "proxyserver";
int loginSeconds = 0;
// Live ID of the user to authenticate to Live services.
string windowsLiveUserId;
// URL of the live service to which user should be authenticated.
// Below is the URL to access Outlook Live service
string serviceUrl = "https://outlook.com/owa/";

// Get Short Lived Token (SLT).
LiveSLT ticket = newLiveSLT(certThumbPrint, siteId, webProxy,
loginSeconds);
shortLivedToken = ticket.GetSLT(windowsLiveUserId);
// Get redirect URL using SLT.
// Using this re-redirect URL, user can be automatically logged in to the
// requested Live service.
string redirectUrl = LiveRPS.CreateRedirectString(shortLivedToken,
serviceUrl);
```

LiveSLT and **LiveRPS** are classes in SSO toolkit SDK. In the above code snippet, Short Lived token (SLT) is retrieved using **LiveSLT** object. Certificate information, Site ID and the Windows Live ID of the user who should be authenticated to Live Services are passed as inputs. SLT is returned for the user. **LiveRPS** is used to retrieve authenticated service URL for the user. Input to **LiveRPS** is SLT and the URL for the service.

In the code published for portal at <http://code.msdn.microsoft.com/liveateduxplore>, refer to SSOHandler.cs for SSO implementation using SSO toolkit SDK.

SSO toolkit SDK can be downloaded from the Service Management Portal (<http://eduadmin.live.com>) or from Microsoft Connect (<http://connect.microsoft.com>).

Seamless experiences EWS Managed API

In this section, we will discuss EWS Managed API and scenarios where EWS Managed API is used to integrate Live@edu Outlook Live Service into school portals and provide integrated seamless experience to users. The school portal developed has mail and calendar Web Parts where we use EWS Managed APIs to access mail and calendar information.

Overview of EWS Managed API

The Exchange Web Services Managed API provides access to Exchange resources like e-mail, calendar, contacts, distribution lists, tasks, and Exchange folders. The EWS Managed API is a fully object-oriented API built on the EWS XML protocol; it provides an easy-to-learn, easy-to-use, and easy-to-maintain .NET interface to Exchange Web Services. Figure 2 shows the Exchange resources accessed by EWS API.

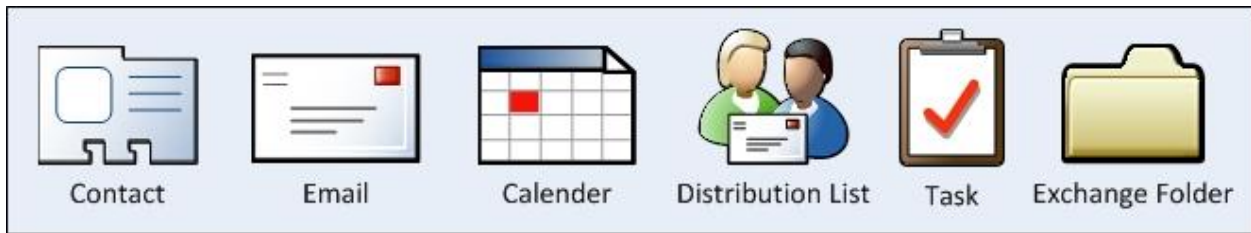


Figure 2: Exchange Resources

The EWS Managed API provides interfaces for managing the Exchange resources. Managing Exchange resources include features for creating, updating, retrieving, and deleting data in an object-oriented fashion. Each item type and folder type is exposed through a dedicated class.

The below diagrams illustrate the Item class hierarchy and Folder class hierarchy provided by EWS Managed API.

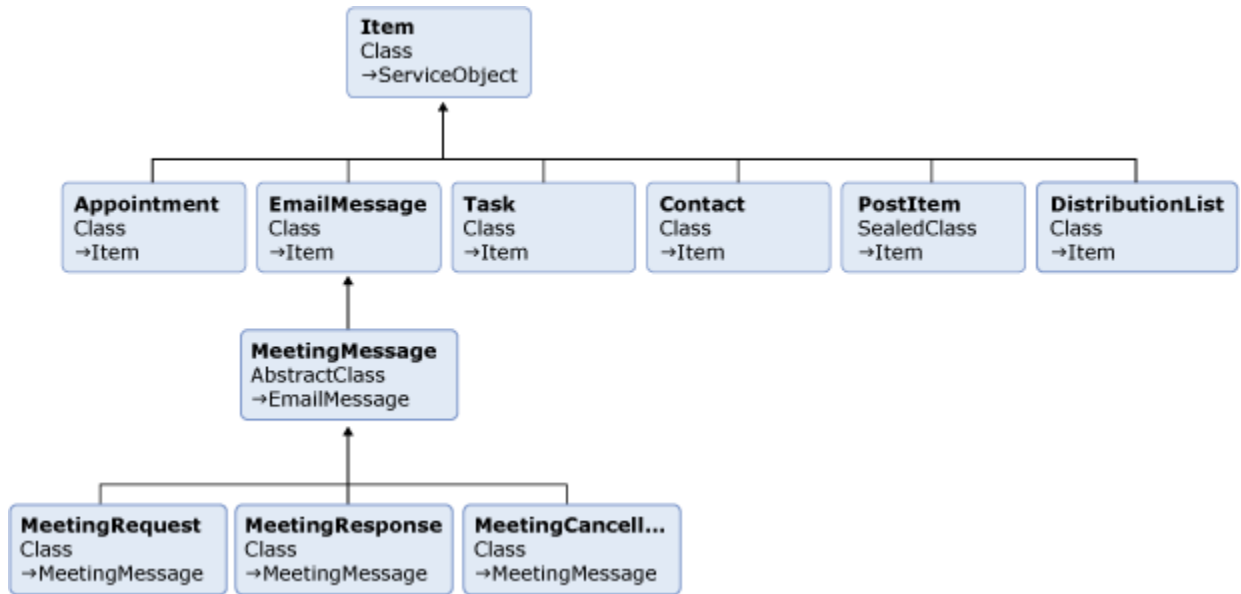


Figure 3: Item Class Hierarchies

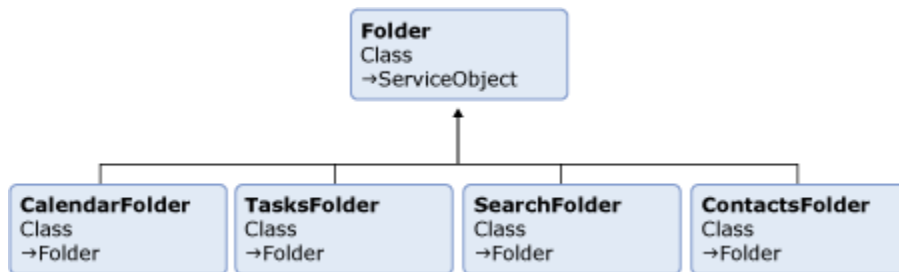


Figure 4: Folder class hierarchies

Autodiscover Service

To access Exchange resources using EWS Managed API, the first step is to create a service object. This can be done in two ways.

- Provide EWS end point
- Obtain EWS end point Using Autodiscover service

The pros and cons of both these methods and their implementation details are explained below.

Provide EWS end point

Below is a code snippet to instantiate a service object and provide a EWS end point.

```
// Create a service object to communicate with Exchange (Outlook Live).
ExchangeServiceservice = newExchangeService(ExchangeVersion.Exchange2010);
// Provide credentials.
service.Credentials = newWebCredentials(
    "admin@contosouniversity.com", "password");
// Exchange Web Service end point.
service.Url = newUri("https://sn1prd0202.outlook.com/EWS/Exchange.asmx");
```

As you can see, the EWS end point URL is assigned to the service. In this approach at the time of developing the application, the EWS end point should be known. Any change to the EWS end point URL will require a change and re-compile of the application or updating configuration files. This maintenance can be avoided using Autodiscover as discussed in the following section.

Autodiscover Service

Microsoft Exchange Server 2010 includes the Autodiscover service, which has been a key component since Exchange Server 2007. This service allows your clients to retrieve the URLs that it needs to gain access to the web services offered by Exchange.

```
// Create a service object to communicate with Exchange (Outlook Live).
ExchangeServiceservice = newExchangeService(ExchangeVersion.Exchange2010);
// Provide credentials.
service.Credentials = newWebCredentials(
    "admin@contosouniversity.com", "password");
// Autodiscover EWS end point.
service.AutodiscoverUrl(
    "professor@contosouniversity.com", delegate(string url) { returntrue;});
```

Explicitly providing the EWS end point URL is faster than using Autodiscover. However, if the endpoint changes down the line after the application is developed, the EWS end point URL should be updated which is work that can be avoided. We designed our portal by combining the advantages of both approaches. When the application starts, we do an Autodiscover and store the service URL returned and henceforth use the manual option on this URL.

Impersonation

Exchange Impersonation enables a caller to impersonate a given user account. This enables the caller to perform operations by using the permissions that are associated with the impersonated account, instead of the permissions that are associated with the caller's account. For instance in the school portal we developed, we use impersonation to display the email and calendar data of the logged in user in the mail and calendar Web Parts respectively.

The advantage of using impersonation is that we are not dependent on the user's credentials to integrate their mailbox information in to the school portal. The below code snippet demonstrates how to use impersonation.

```
ExchangeServiceservice = newExchangeService(ExchangeVersion.Exchange2010);
// Provide credentials of user that has impersonation rights enabled.
service.Credentials = newWebCredentials(
    "admin@contosouniversity.net", "password");
service.AutodiscoverUrl(
    "student@contosouniversity.net", delegate(string url) { returntrue;});

// Impersonate user professor@contosouniversity.net.
service.ImpersonatedUserId = newImpersonatedUserId(
    ConnectingIdType.SmtpAddress,
    "student@contosouniversity.net");
```

In the above code snippet, user admin@contosouniversity.net impersonates student@contosouniversity.net. Impersonation is a powerful feature which will lets users with the **ApplicationImpersonation** role to impersonate other users and perform actions on their behalf. **ApplicationImpersonation** is a management role available in Outlook Live.

Web Parts

We developed mail and calendar Web Parts to retrieve and display mail and calendar information for users (student/ faculty) in the school portal. Web Parts are ASP.NET server controls. To create a new Web Part, you must create an ASP.NET custom control. However, unlike standard ASP.NET controls, which are added to web form pages at design time, Web Parts are intended to be added to Web Part Zones on Web Part Pages at run time. The primary advantages of a Web Part are:

- They allow for personalization of page content and provide the flexibility to be moved or hidden to change the page layout.
- Web Parts allows users to export or import Web Parts settings for use in other pages. Web Parts retain the properties, appearance and the data across the pages when imported or exported. In our case, they can be easily integrated in to Microsoft SharePoint 2007.
- Web Parts can be assigned role-based access so you can determine which Web Parts can be shared by all or which should be hidden for certain roles. This helps us to provide customized content based on security.
- Web Parts can talk to each other. You can utilize the data from one Web Part to another Web Part for different purposes.

The code snippet below provides a simple example of how to display a “Hello World” string using Web Parts.

```
publicclassSimpleWebPart : WebPart
{
    privatestring displayText = "Hello World!";

    [WebBrowsable(true), Personalizable(true)]
    publicstring DisplayText
    {
        get { return displayText; }
        set { displayText = value; }
    }

    protectedoverridevoid Render(System.Web.UI.HtmlTextWriter writer)
    {
        writer.Write(displayText);
    }
}
```

Mail WebPart

In this section, we will discuss how the Mail Web Part control is built. There are two primary tasks this Web Part does.

- Retrieve the mail information from Outlook using EWS APIs
- Render Main information to the UI.

Fetch Mail items

In the Mail Web Part developed for the school portal, only the first five items from the user’s inbox are displayed. The code snippet provided below shows how this information is retrieved. Once the information is retrieved, iterate through the mail Items and extract the required information and finally render it to the UI.

```

// Bind to Inbox folder.
Folder inbox = Folder.Bind(service, WellKnownFolderName.Inbox);
// Fetch first 5 messages from Inbox.
FindItemsResults<Item> messages = inbox.FindItems(newItemView(5));

// Iterate through each mail and retrieve required information.
foreach (Item item in messages)
{
    EmailMessage message = (EmailMessage)item;

    // Load the properties we are interested in.
    message.Load(newPropertySet(newPropertyDefinitionBase[]
    {
        EmailMessageSchema.Subject, EmailMessageSchema.Body,
        EmailMessageSchema.From, EmailMessageSchema.DateTimeCreated
    }));

    // ... Rendering logic
}

```

As shown in the above code snippet, the first step to retrieve mail information is to create binding to the **Inbox** folder. When retrieving mail Items from the Inbox, you can specify the number of messages to retrieve. Once the messages are retrieved, one can iterate through the messages and perform a desired action which in our case is render mail data to UI.

In the code published for portal at

<http://code.msdn.microsoft.com/liveateduexplore>, refer to WebPartMailControl.cs in the LiveAtEduControls project for Mail Web Part implementation using EWS Managed API.

Calendar WebPart

In the Calendar WebPart developed for the school portal, all events for the current day for logged in users are displayed. To achieve this, similar to the Mail Web Part, we need to retrieve this information from Outlook using EWS Managed API.

Fetch Calendar items

CalendarFolder object is used to fetch calendar items. **CalendarView** object is used to specify the date range between which events are to be fetched. Following code snippet fetches all events for the current day.

```

// Bind to Calendar folder.
CalendarFolder calendar = CalendarFolder.Bind(service,
    WellKnownFolderName.Calendar);
// Retrieve appointments for the current day.
FindItemsResults<Appointment> appointments = calendar.FindAppointments(
    newCalendarView(DateTime.Today, DateTime.Today.AddDays(1)));

```

Now that we have all appointment details for the current day for a user, the next step is to iterate through each of these appointments and get the specific information such as subject, location, start and end time, etc. and render this info to UI.

In the code published for portal at

<http://code.msdn.microsoft.com/liveateduexplores>, refer to WebPartCalendarControl.cs in the LiveAtEduControls project for Calendar Web Part implementation using EWS Managed API.

Adding Events to a Student's Calendar

Public events, holidays, academic timetable, etc. can be added to a student's calendar. By adding school events directly to a student's calendar, it enables them to learn more about many of the exciting events taking place on the campus and be aware of what is happening around them. This scenario is enabled by impersonating a student and adding events to student's calendar.

In the school portal developed, the events are read from an xml file. Event information has attributes like name of the event, description, location of the event, and schedule of the event. The below XML shows sample data for event information.

```
<CalendarEvents>
<CalendarData>
<Subject>Cougars vs Nittany Lions</Subject>
<Body>
<BodyType>Text</BodyType>
<Text>Be there when the Cougars take on the Nittany Lions</Text>
</Body>
<StartDate>2010-05-24T18:00:00</StartDate>
<EndDate>2010-05-24T22:00:00</EndDate>
<Location>The Arena</Location>
<RequiredAttendees />
<OptionalAttendees />
<RecurrenceAppointment />
</CalendarData>
</CalendarEvents>
```

Event object used in the code snippet below is a custom data structure in the school portal application which contains information read from the above xml file. This data can be read from a database or any other data source in real world apps. The following code snippet shows how to impersonate each user and create an event in their calendar. **User** is another custom object in the school portal application.

```
// Create the service object.
ExchangeServiceservice = newExchangeService(ExchangeVersion.Exchange2010);
service.Credentials = newWebCredentials(
    "admin@contosouniversity.net", "password");
service.AutodiscoverUrl(
    "admin@contosouniversity.net", delegate(string url)
        { returntrue;});

// Iterate through each user and add the event
foreach (User user in users)
{
    // Impersonate the user.
    service.ImpersonatedUserId = newImpersonatedUserId(
        ConnectingIdType.SmtpAddress,
        user.WindowsLiveId);

    // Iterate through each event and add the event.
    foreach (Event event in events)
    {
        Appointment appointment = newAppointment(service);
        appointment.Subject = event.Subject;
        appointment.Body = newMessageBody();
        appointment.Body.Text = event.Description;
        appointment.Start =event.Start;
        appointment.End = event.End;
        appointment.Save();
    }
}
```

In the code published for portal at

<http://code.msdn.microsoft.com/liveateduxplore>, refer to AddEvents.cs in the LiveAtEduDemo project for adding events to a user's calendar using impersonation.

Integration with PowerShell

Administrators of a school's IT department may typically get requests from users (student, faculty, staff, management) to reset passwords, create new aliases add aliases, create Dynamic Distribution Lists, block mail from one user to another, and so on. In a connected campus, the administrator can perform all these tasks by logging in to the admin home of the school portal.

In order to achieve this we have developed a C# Outlook Live wrapper over PowerShell cmdlets. Whitepaper available at <http://code.msdn.microsoft.com/psoutlooklive/> explains in detail about the C# Outlook Live wrapper with code snippets on how to perform these common tasks from a portal.

Overview of PowerShell

Windows PowerShell is a command-line shell and scripting language that you can use to manage your organization (Outlook Live domain).

Windows PowerShell uses administrative tasks called cmdlets. Each cmdlet has required and optional arguments, called parameters, that identify which objects to act on or control how the cmdlet performs its task. You can combine cmdlets in scripts to perform complex functions that give you more control and help you be more efficient.

You can use Windows PowerShell on a local computer to connect to your Outlook Live organization and perform management tasks that aren't available or practical in the web management interface. For example, you can create Dynamic Distribution Groups, create or update many user accounts at a time, and script automated solutions.

Working with C# Outlook Live wrapper

To perform any administrative task on Outlook Live using PowerShell, a remote session to Outlook Live has to be created and imported. Creating and importing a session are heavily time consuming operations and because of that we designed the portal in a way where these operations are performed only when needed. To understand the design let us first define the pre-requisites.

OutlookLiveSessionRunspace

This object represents a PowerShell Runspace with Outlook session imported. This is the primary object upon which all requests are performed.

SessionRunspaceFactory

This is a factory class that creates an instance of the OutlookLiveSessionRunspace object.

SessionRunspaceSingleton

This is a custom singleton implementation for the OutlookLiveSessionRunspace object.

To perform any operation in PowerShell we need an instance of the OutlookLiveSessionRunspace object. As seen in Figure 5, every time a request for this object is received, the SessionRunspaceSingleton checks on the current instance of OutlookLiveSessionRunspace and sees if it is valid by performing a trivial, non-time consuming task. If the task is completed successfully it returns the current instance back to the caller else a new instance is created by calling the SessionRunspaceFactory and the new instance is returned back. By this design, we circumvent the need to create and import a session every time and at the same time guarantee that the session is always valid and has not expired.

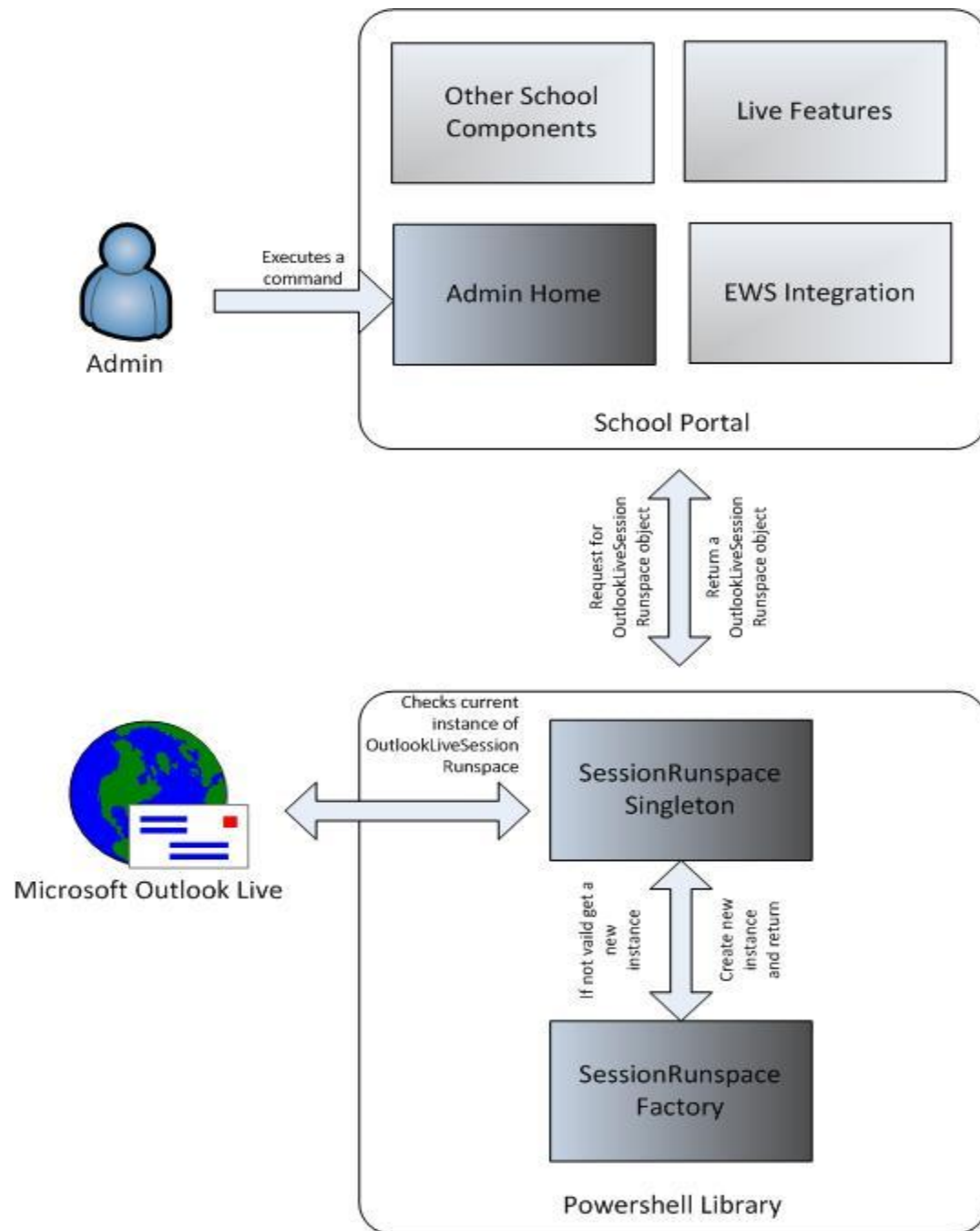


Figure 5: Step by step PowerShell execution

A whitepaper available at <http://code.msdn.microsoft.com/psoutlooklive/> explains in detail about the C# Outlook Live wrapper with code snippets.

In the code published for portal at <http://code.msdn.microsoft.com/psoutlooklive/>, refer to PowerShell project for implementation details.

Summary

In this document we discussed different features considered for developing the school portal. We also discussed in detail how to build a comprehensive school portal with Live@edu services. We discussed using SSO toolkit for providing integrated Single Sign on experience when accessing live services. We discussed EWS Managed API and how to use it to retrieve mail and calendar information. We also discussed how to expose admin functionality through the portal using PowerShell.

These different scenarios and features should get you started with integrating Live@edu services in to your school portal or building a new school portal with Live@edu services.

Glossary of Acronyms

- AD – Active Directory
- SSO – Single Sign On
- SDK – Software Development Kit
- SLT – Short Lived Token
- EWS – Exchange Web Services
- API – Application Programming Interface

References

- Active Directory Authentication from ASP .NET
 - [http://msdn.microsoft.com/en-us/library/ms180890\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/ms180890(VS.80).aspx)
- Single Sign On (SSO)
 - <http://www.microsoft.com/liveatedu/single-sign-on.aspx?locale=en-US&country=US>
- SSO toolkit SDK download:
 - from Service Management Portal (<http://eduadmin.live.com>)
 - from Microsoft Connect (<http://connect.microsoft.com>)
- Exchange Web Services overview
 - <http://msdn.microsoft.com/en-us/library/dd877045.aspx>
- EWS Managed API overview
 - [http://msdn.microsoft.com/en-us/library/dd633709\(EXCHG.80\).aspx](http://msdn.microsoft.com/en-us/library/dd633709(EXCHG.80).aspx)
- EWS Managed API download
 - <http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=c3342fb3-fbcc-4127-becf-872c746840e1>
- Install and Configure Windows PowerShell
 - <http://help.outlook.com/en-us/140/cc952756.aspx>
- Outlook Live help
 - <http://help.outlook.com/>
- Details on Windows PowerShell scripting
 - <http://technet.microsoft.com/en-us/library/bb978526.aspx>
- Reference to PowerShell cmdlets for Outlook Live
 - <http://help.outlook.com/en-us/140/dd575549.aspx>
- How to install and configure Windows PowerShell?
 - <http://help.outlook.com/en-us/140/cc952756.aspx>
- Portal Code
 - <http://code.msdn.microsoft.com/liveateduexplore>
- PowerShell Code
 - <http://code.msdn.microsoft.com/psoutlooklive/>