Extended Delivery Service

Rooster 2.1 Automation SDK

Introduction

Rooster Extended Delivery Service is designed so that third party addins can interface with Rooster can operate in and on the Rooster environment. Addins can provide functions that Rooster itself does not provide.

Addins can be authored by the end (power) users, site administrators, third-party vendors, or just about anybody who wish to automate some process, add additional feature or simply enhance his or her Rooster experience.

Rooster's EDS uses Component Object Model to communicate with the clients (addins). Clients can be programs (written in any programming language) and scripts. The most transparent programming language for COM clients, for the Windows platform, is Microsoft Visual Basic, and the most common scripting languages are Python, Java Scripts, JScripts, VBS and others. Enclosed with the extended Rooster distribution are several sample clients written in Visual Basic and several scripts as well. (You can download extended distribution from the Rooster Home Page.)

Rooster's COM Automation server supports any number of clients. All the clients need to be running on the local machine. Support for remote (network) clients will be introduced in on the future releases, if there is sufficient interest.

Rooster's COM server can be turned on and off via Options (Extended Delivery Service Category). To switch Rooster's COM server simply check/uncheck Automation Enabled check-box. In addition, when enabled, Rooster's EDS interface is automatically registered (or unregistered if disabled) in the Running Object Table/Windows Registry.

Rooster EDS interface is exposed via Properties, Standalone Functions and Events.

1. Properties

Properties lets you access miscellaneous program, session or configuration related information. Information is requested by the addin, and is returned immediately.
[In the future releases, properties may also be used so the Rooster can obtain particular, client specific, information. There will also be more of this, "classic", properties.]

VersionMajor - Rooster series-number
VersionMinor1 - First minor number
VersionMinor2 - Second minor number
VersionBuild - Version build number

Subscriptions(0) - Array of feed aliases
Subscriptions(1) - Array of feed URLs
Subscriptions(2) - Array of feed names

ClientsCount - Number of currently connected addins

2. Standalone Functions

Standalone functions are primarily intended to start particular action. Following functions are currently available:

AboutRooster - Open Rooster About box
DisplayOptions([0-20]) - Open Options dialog
[Number refers to the zero indexed Category, from the top to the bottom.]

SubscriptionsImport([opml file path]) - Import subscriptions
SubscriptionsExport - Export subscriptions

OpenBoxes - Open the Boxes window
OpenFeeds - Open the Feeds window
OpenTasks - Open the Tasks window
DisplayNewsTicker - Open the News Ticker
OpenInbox - Open the Inbox
OpenTrash - Open the Trash

FeedValidate([feed url]) - Validate the feed
FeedPreview([feed url]) - Preview the feed
FeedSubscribe([feed url]) - Subscribe to a new feed

Windows(-1) - Close all open windows
Windows(0) - Restore all open windows
Windows(1) - Minimize all open windows

Windows(2) - Maximize all open windows
Windows(3) - Stretch all open windows
Windows(4) - Center all open windows

3. Events

Unlike the standard Methods and Properties, EDS Events are implemented in slightly different manner. This is how it works:

Rooster's COM addins need to apply for particular Event. There are 3 major events: On Display, On Request and On Delivery. When the event occurs, EDS fires a notification function, which is implemented by the event sink interface on the client side, with particular data passed via parameters of the (notification) function. For example:

VB Addin

```
'Declarations section
Private WithEvents edsObject As RoosterLib.EDS

<...>

'Connect
Set edsObject = CreateObject("Rooster.EDS.1")

<...>

'Apply for the event
edsObject.ApplyForDisplay edsObject

<...>

'This is the event notification function
Private Sub edsObject_DisplayingArticles(ByVal bsAuthor As String,
                                         ByVal bsHeadline As String,
                                         ByVal bsFeedName As String,
                                         ByVal bsDate As String,
                                         ByVal bsArticleBody As String)

'Article information automatically picked up from here. For example:
MyNewHeadline$ = bsHeadline

End Sub

<...>

'Call off On Display application (do not receive notifications anymore)
edsObject.CallOffDisplay edsObject

'Disconnect
Set edsApplication = Nothing

Notes:
```

```
- RoosterLib is the library, and EDS is the common name of Rooster's
application object.
- To add the reference to the Rooster Type Library, click Project/References,
and check "Rooster 1.0 Type Library" from the list.
```

C++ Addin

```
//Declarations section
IEDSPtr edsObject;

<...>

//Connect
::CoInitialize(NULL);
HRESULT hr = edsObject.CreateInstance("Rooster.EDS.1");

//Apply for the event
if (SUCCEEDED(hr)) edsObject->ApplyForDisplay(edsObject);

<...>

//This is the event notification function.
void CMyAddinEventSink::DisplayingArticles(LPCTSTR bsAuthor,
                                           LPCTSTR bsHeadline,
                                           LPCTSTR bsFeedName,
                                           LPCTSTR bsDate,
                                           LPCTSTR bsArticleBody)
{

//Article information automatically picked up from here. For example:
strMyNewHeadline = bsHeadline;

}

<...>

//Call off On Display application (do not receive notifications anymore)
edsObject->CallOffDisplay(edsObject);

//Disconnect
edsObject.Release();
::CoUninitialize();

Notes:
- Derive CMyAddinEventSink class from the CCmdTarget base class.
```

For C++ applications you need to implement event sink interface to receive notifications, and for VB applications it is sufficient to declare your WithEvents COM object as stated above. VB IDE will automatically create the function prototype for you.

During the session, single addin can apply (and call off the application) for any number of events, any number of times. Addin can apply for different types of events simultaneously. Addin needs to apply for single type of event (for example On Display) only once (unless it calls off the application).

Addin should call off the application (for particular event) before disconnecting from the server (for every type of event it applied for).

Addin should disconnect from the COM server before quitting.

if "Exit Rooster when the last addin disconnects" (Options/Exiting) check-box is checked, Rooster will automatically exit when the last addin quits. This applies only if Rooster session was started from that or another addin.

3.1 On Display Event

On Display event occurs when the article is opened from the newsbox summary. At that moment, all addins that are applied for this event will receive event notifications with the article information passed as function parameters.

ApplyForDisplay (...**)**
Apply the addin for On Display events.

DisplayingArticles (...**)**
When the article is opened, event is fired.

CallOffDisplay (...**)**
Call off the application for On Display events.

3.2 On Request Event

On Request event occurs when the user clicks "Pass to Addins" from the article-menu.

ApplyForRequest (...**)**
Apply the addin for On Request events.

RequestingArticles (...**)**
When the menu item "Pass To Addins" is clicked, event is fired.

CallOffRequest (...**)**
Call off the application for On Request events.

3.3 On Delivery Event

On Delivery event occurs when the articles are downloaded. There are two sub-types of delivery-events: On Delivery Royal & Get Articles.

3.3.1 On Delivery Royal

On Delivery Royal applies for all future downloads, for all subscribed feeds (royal stuff). Addin will receive the articles as long as it is applied for this event. Articles download may be initiated directly from the Rooster's main interface, or from the addin.

ApplyForDeliveryRoyal (...**)**
Apply to receive all the articles when the feeds are checked.

GettingArticles (...**)**
After being downloaded, articles are passed to the clients by the way of this function.

CallOffDeliveryRoyal (...)
Call off the application for Delivery Royal.

3.3.2 Get Articles

Unlike Delivery Royal, "Get Articles" is one-time event. When the function is called, Rooster will download, and pass the articles, for particular feed, to the addin. Feed alias is specified as parameter of "GetArticles" function.

GetArticles (...)
Download articles for particular feed. One time only.

GettingArticles (...)
Get the articles here. The same function is used by Delivery Royal.

You will notice there is no "call off" function as this is one-time event.

4. Data Types

Data types are different, depending on the client and the platform. The most common are String, Integer and Variant (VB), LPCTSTR, SHORT and VARIANT (C++) etc.

5. How to Enable/Disable Extended Delivery Service

To enable EDS via graphical interface:
1. Start Rooster
2. Click Options/EDS
3. Check "Enable Automation"
4. Hit OK

To enable EDS via command-line:
1. Open the Windows Explorer and browse to the Rooster program folder
2. Open the Command-prompt
3. Type: Rooster.exe /RegServer
4. Hit Enter

To disable EDS via graphical interface:
1. Start Rooster
2. Click Options/EDS
3. Uncheck "Enable Automation"
4. Hit OK

To disable EDS via command-line:
1. Open the Windows Explorer and browse to the Rooster program folder
2. Open the Command-prompt
3. Type: Rooster.exe /UnRegServer
4. Hit Enter

6. EDS Interface Specs

Short (Common) Name: EDS
Class Name: CEDS
Interface Name: IEDS
Events Interface Name: _IEDSEvents
Events Interface ID (IID):
{ 0xe54d5dcf, 0x02d8, 0x40ca, { 0x82, 0xc6, 0x60, 0xc4, 0x8c, 0x0f, 0x09, 0x9b } }

Object Lifetime: EDS object is valid as long as there are connected addins and the automation is enabled.

Dual Interface: EDS supports standard IUnknown and IDispatch interfaces.

7. Updates

EDS may not be updated with every release of Rooster. Future versions may contain not only the new features but the enhanced versions of current features. So please track the change log (of the Release Notes) for information on updates. If API is updated, clients of course need to be updated as well, in order to work with the corresponding Rooster release.

As you can see there are huge possibilities regarding the inter-program exchange and in which direction the EDS will go depends partially on you. If your business demands particular feature feel free to contact me and lay down your ideas.

EDS will be focused on, and will explore, primarily COM. Other types of API are also possible, but will not be implemented in the near future.

8. Your Own Addins

If you want to share your addins with others, feel free to send the URL's and I'll add them to the Rooster web site. This applies for both programs and scripts.

9. Usage & Examples

As mentioned earlier, EDS allows us to access and extend many of Rooster's basic functions. For example, let's say you maintain a web site. On the site you want to have a frame with the latest news about football player XYZ, or a movie star or particular news event from your county and so on. All you need to do is subscribe to particular feed(s), and set up you custom addin to receive On Delivery Royal event notifications. Inside the notification function just specify the search criteria (in our case "player XYZ", movie star name or a county name) and if you have a match, update or append the web site text. So once you set up the function, your web site will automatically be updated with the relevant information. It's so easy you can't beat it with a stick.
In the similar manner you can update the database or the news-ticker, create custom notifications and so on. (For the sample code please see edsOnDelivery example.)

Here is another example: You just downloaded the articles from your favorite news feed. One of the articles you wish to save in your e-mail storage. In the article window click Article/Pass To Addins, and your COM addin (which is applied for On Request events) will receive the article information. From there you can import the article to your e-mail client (is the client provides some type of API that supports importing of messages). If the client supports COM (as Office Outlook for example), that will make the task even easier. (For the sample code please see Feed2Outlook example.)

As a part of the Extended Rooster Distribution, this SDK and documentation is accompanied with several sample addins:

● edsOnDelivery shows how to apply for Royal Delivery and how to download and get articles for particular feed.

● edsOnRequest shows how to pass a single open article to the addin(s).

● edsOnDisplay demonstrates how to automatically obtain articles information when the articles are opened (from the Rooster's main interface).

● edsFunctions shows how to validate, preview and subscribe to a feed. It also demonstrates how to open Rooster's main windows and how to arrange or close windows. In addition, it shows how to facilitate Import and Export functions.

● edsProperties shows how to display subscriptions related information. List-boxes are populated with (subscribed) feed aliases, names and URLs.

● Feed2Outlook let's us copy (or download) articles to the Outlook mail storage.

Appendix - Launch on Startup

Additional feature was added at the late stage of Rooster 2 development.

It's the possibility to start collection of addins when the Rooster starts. This applies for programs (exe-files) and scripts (VBS, JScripts and Python scripts).

To add your program (or script) to the startup collection, start Rooster and click Tools/Addins. Click Add, browse and select the addin. Toggle Launch on Startup button for each addin you wish to be executed on startup.

Note. Command-line parameter "edsStartup" is passed when the addin is launched just as an indication it has been launched by Rooster, on startup, and that Automation is enabled. If the Automation is disabled, addins startup-options are suspended.