Repstor affinity™

Extensibility Guide

Product Version 3.7.1
December 2017

Contents

Introduction

The Repstor products can be tailored as required to individual customer requirements and solutions. Extensibility is available at a number of levels, from basic configuration to full custom extensions.

This document describes various extensibility points within the Repstor affinity product. It should be read in conjunction with the Repstor affinity installation and configuration guide.

This document assumes familiarity with the Repstor affinity product, with Outlook and with SharePoint solutions.

Configuration within Custom Solutions

There are a number of features of Repstor affinity that are categorized as configuration, but are key elements of any customized solution. These are fully documented in the Repstor affinity installation and configuration guide.

- Central Repositories list – used to help Repstor affinity interact with a server-side solution. The Central Repositories list is a dynamic server side list that gives the Repstor client information around what repositories should be synchronized – and how they are synchronized.
- Property and view synchronization. Many repositories support synchronizing properties and displaying them in views. These properties can be displayed in the Outlook list view, and within searches.

Outlook as an Extensibility Platform

Repstor affinity stores its information in an Outlook message store. This means that the Affinity repositories, folders, and content are all accessible within the standard Outlook object model. Standard Outlook customization solutions can be used to tailor the Repstor message store. This guide will describe the properties to examine, and any special considerations when using Outlook solutions. Technologies like *Visual Studio Tools for Office,* and the *Outlook View Control* can be used as part of any Repstor affinity solution.

Repstor Protocol Handler

Repstor implements a protocol handler which is compatible with the SharePoint stssync protocol handler. The Repstor protocol handler adds some additional functionality that can be used to provide Repstor-specific functionality.

The protocol handler URLs can be used on any web page in order to perform operations against Repstor affinity.

The standard protocol handler (installed by Outlook) will add SharePoint document libraries into Outlook. Affinity replaces this protocol handler to redirect document libraries to be synchronized in Outlook by Repstor affinity. The Repstor protocol handler can be used to add other non-SharePoint repository types into Outlook.

Example standard stssync command:

stssync://sts/?ver=1.1&type=documents&cmd=add-folder&base-url=http://oink-sid5/subsite&list-url=/Shared%20Documents/&guid={27d2281d-0c14-4e00-8ab1-203124d09e7f}&site-name=subsite&list-name=Shared Documents&folder-url=/joe/bloggs&folder-id=2

Standard stssync parameters for Repository Synchronization

Name	Description
Base-url	The URL of the site
List-url	The list component of the URL
Guid	SharePoint Guid of the list
List-name	Name of the list
Folder-url	URL associated with sub folder to be synchronized
Site-name	Name of the SharePoint site
Cmd	Add-folder – no other commands are supported.
Type	Documents – no other types are supported by Repstor. Other types (like calendar) will be forwarded to the previous stssync handler that Repstor's version replaced.

Repstor custom stssync functionality

Repstor adds some additional stssync commands that can be called from any web page.

Name	Value and example setting
Stssync://roam/add=<encodedurl>	Add a repository with the given URL. The URL will be any URL displayed in the browser when viewing a SharePoint document library, list of sub-folder. You can also prefix the URL with the repository type (e.g. :::SharePoint:::) and a parent folder setting (e.g. :::Docs:Private:::) Stssync://roam/add=https%3A%2F%2Frepstor.sharepoint.com%2Fsites%2Fdemo2013%2FShared%2520Documents%2FForms%2FAllItems.aspx%3FRootFolder%3D%252Fsites%252Fdemo2013%252FShared%2520Documents%252FInformation%2520Requests%26FolderCTID%3D0x01200037B24B4E371074439FB41D298FA24C57%26View%3D%257B422CF3A2-112E-4025-AD61-BA20599BF749%257D stssync://roam/add=%3A%3A%3ASharePoint%3A%3A%3A%3A%3Amy%20docs%3Atest%3A%3A%3Ahttps%3A%2F%2Frepstor.sharepoint.com%2Fsites%2Ftest%2Ffergus%2Fdoc%2520lib%2520with%2520doc%2520sets%2FForms%2FAllItems.aspx
Stssync://roam/reqrepos=<encodedurl>	Set the central repository URL of the affinity system to the encoded URL. Stssync://roam/reqrepos=https%3A%2F%2Frepstor.sharepoint.com%2Fsites%2Fdemo2013%2FLists%2FCentral%2520Repositories%2FAllItems.aspx
Stssync://roam/refresh-reqrepos	Refresh the users central repository list (perhaps after it is known that the server has added a new entry to it)
Stssync://roam/ext=<myinfo>	Pass <myinfo> to the RepstorExt_RunExtensionAction extension for more processing
stssync://roam/add=<urllocation>#filter:<prop internal name>|<propval>:::<repos display name>:::	Synchronize a SharePoint document library filtered to a particular document or set of documents stssync://roam/add=:::SharePoint:::https://repstor.sharepoint.com/sites/Matters/00278/Documents#filter:_dlc_DocId|REPST-1292866892-119:::REPST-1292866892-119:::

Repstor-config command

As part of the Repstor affinity installation, a small executable “Repstor-Config.exe” is placed in the Repstor affinity installation folder. It is used by the stssync protocol handler. Any stssync protocol string can be passed to the Repstor-Config command to perform the equivalent action (for example, in client side batch files).

Example:

Repstor-config.exe “stssync://roam/add=<encodedurl>”

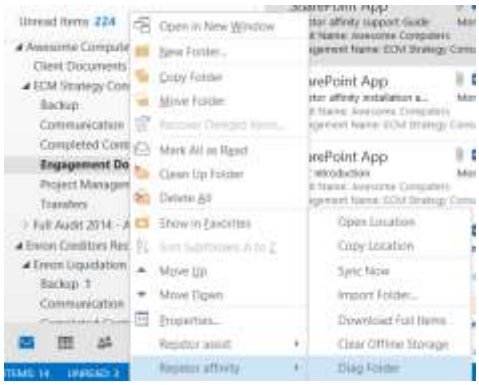
Repstor affinity Property Model

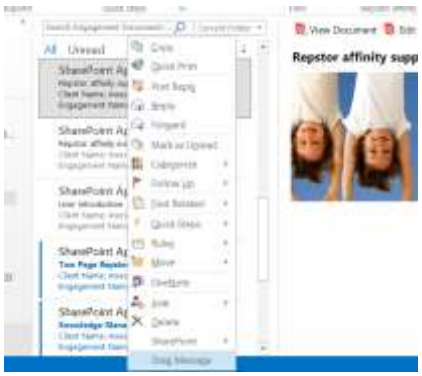
Repstor affinity synchronizes properties from the server repositories into the folders within the Outlook message store. Many of these properties are used to display information and property fields to the user. Some properties are internal and can be used as part of a custom solution.

Repstor will store properties at the folder and message level, and will store additional properties at the Repository level (the top level folder of a synchronized repository). Occasionally properties will be stored at a global level.

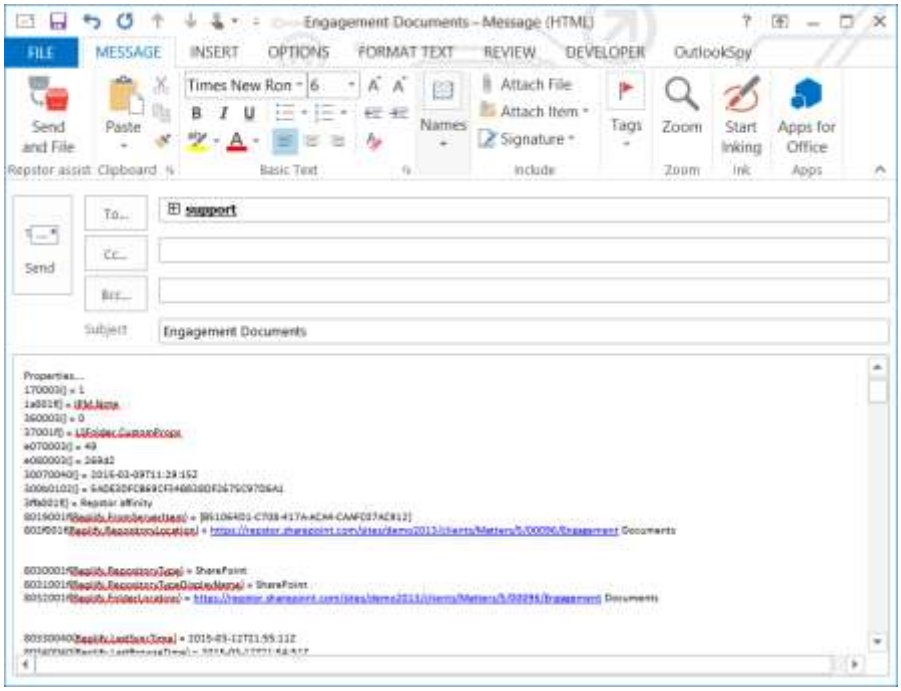
The utility OutlookSpy can be used to view properties stored within Repstor affinity. See http://www.dimastr.com/outspy/home.htm.

You can also display properties that are part of a message or folder by right clicking the item with CTRL and SHIFT held down. This will also appear if when Repstor diagnostics are switched on.





These options display the properties in a newly created message:



Folder Properties

When Repstor affinity synchronizes a folder it will synchronize:

- A standard set of properties used as part of the synchronization
- A set of custom properties that exist on the repository folder

Standard Repstor folder properties (any repository type)

Property Name	Type	Description and Example Value
0x3001001F	String	Folder name
Replify.FromServerItem	String	Unique GUID of the item within the repository. {D1E8B739-6CD0-42CF-A40E-1ADF9F9297C4}
Replify.FromServerShortItemId	String	Short ID of the item within the repository. 5835
Replify.RepositoryLocation	String	URL to the repository https://repstor.sharepoint.com/Shared Documents
Replify.RepositoryType	String	Repository type being synchronized. SharePoint, WorkSite, FileShare, SkyDrive
Replify.FolderLocation	String	URL to the folder within the repository – this format is used when you select “open folder” from the affinity folder context menu. https://repstor.sharepoint.com/Shared Documents/Releases/2.4.0/Documentation
Replify.LastSyncTime	DateTIme	Last time a synchronization of this folder was performed. (Used to queue folder synchronizations) 16:18, 05/08/2014
Replify.LastBrowseTime	DateTIme	16:18, 05/08/2014 Last time the user browsed this folder. (Used to expire folders)
Replify.SyncIssueFolderMessage	String	Long message displayed in the folder properties page to give information on the last synchronization of this folder. Synchronization started: 05/08/2014 17:18 Folder: https://repstor.sharepoint.com/Shared Documents/Releases/2.4.0/Documentation Uploading ChangesSkipping edit doc... Checking for deleted items Downloading new or changed items Downloading sub folders Synchronization completed: 05/08/2014 17:18

Replify.LastSyncDetailsFolderMessage	String	Short message displayed in folder “Sync Now” tooltips. Folder Synchronization Complete No changes.
ROAM.NumSyncsSinceAvailabilityCheck	Number	Used internally to determine when next full check will be performed on the repository (will re-check permissions changes) 4
Etag	String	Etag of the folder. Not used currently. {D1E8B739-6CD0-42CF-A40E-1ADF9F9297C4},1
ROAM.PermMask2	String	Permission mask used to determine the user’s permissions on the folder. E.g. whether they can add and delete documents. 0x7fffffffffffffff
Replify.SPS.RequireCheckout	String	Does this document library require the user to explicitly check out or not. False
ROAM.HeaderDownloadType		
ROAM.ExpiredFolder		

Standard affinity properties (for SharePoint repositories)

SPFolderPath	String	Folder path within the document library. Shared Documents/Releases/2.4.0/Documentation
SPIsList	Boolean	False for document libraries, true for lists. False
SPListId	String	Guid of the list this folder is a part of. {25ACE67A-D27D-4522-BFA3-401D52303250}
SPListName	String	Name of the list this folder is part of. Shared Documents
SPWebUrl	String	URL of the SharePoint site this list is part of.

		https://repstor.sharepoint.com
ChangeToken	String	Token used to perform incremental synchronizations. Clear this to resync the entire folder. 1;3;25ace67a-d27d-452-bfa3-401d52303250;635428523123130000;29360654
NextPageToken	String	Used in batched incremental synchronizations.

Typical affinity properties (for SharePoint repositories)

All the other folder properties associated with the SharePoint folder are synchronized. These include custom folder properties that have been added. They are all of type "String". Properties are renamed from the field names in SharePoint according to the following rules.

- Any occurrence of the "_" character is replaced with "-" (underscore is replaced with single hyphen)
- Any occurrence of "-" is replaced with "----" (existing hyphens replaced with 4 hyphens)

See Appendix A for a set of example folder properties.

Document Properties

When Repstor affinity synchronizes a document it will synchronize:

- A standard set of properties that are used by Repstor as part of the synchronization
- A set of custom properties – providing full access to the properties of the SharePoint (or other repository item). These are stored in Outlook as string properties.
- It will then convert some of the custom properties into Outlook friendly properties that can be used in Outlook views and property pages. The converted properties are those that are listed as fields of the SharePoint document library or list.

Standard Repstor properties (any repository type)

Property Name	**Type**	**Description and Example Value**
0x0037001F	String	Document display name / subject
Replify.FromServerETag	String	The server etag associated with this document {0B7A462D-BC4C-45AA-A714-01491214007D},11
Replify.FromServerListId	String	The List Id where this document is stored. {25ACE67A-D27D-4522-BFA3-401D52303250}

Replify.FromServerItem	String	The Guid of this item within the server. {0B7A462D-BC4C-45AA-A714-01491214007D}
Replify.ItemContentType	String	Content Type of this item. Document
Replify.FromServerShortItemId	String	The SharePoint short item identifier of this document. 5840
Replify.FromServerVersion	String	The internal version number of this item. 9
Replify.FromServerFileName	String	The filename associated with this item. Extensibility Guide.docx
Replify.MessageIsUnchanged	Boolean	Boolean property to indicate whether this document has been changed in the client since it was synchronized. This property is changed automatically when any properties of the item change. This property is used to determine pending item changes. False
Replify.FormRegionClass	String	Message Class we want to assign to the item to override the outlook one and display our own icon/property page. IPM.Post.ROAM.Doc.SharePoint.docx.Document
Replify.BeforeFormRegionClass	String	Message Class the item had before we synchronized it. (The standard Outlook message class of the item) IPM.Document.Word.Document.12
ROAM.AddedByMe	Boolean	Document was added by current user, from this client. False
ROAM.DuplicateHash	String	Hash of document (or key properties if email). Used in duplicate detection.

		{A4E75CFC-2BD3-2E80-CFAA-2C284E2F263C}
ROAM.IsDuplicate	Boolean	Indicates if this item is detected as a duplicate or not. False
Replify.ChangedFields	String	Used to flag fields which have changed (see below)
To Adjust adding behavior		
ROAM.ForcePromptForProperties	Boolean	True if the user should be prompted for properties on addition
For items in error		
Replify.SyncIssueType	String	
Replify.SyncIssueMessage	String	
For items currently launched for editing by the user.		
Replify.DocumentLaunched	String	Set on documents to indicate that they are currently being edited by the user. 1
Replify.LaunchedFilePath	String	Launch path where an edited document is currently being edited. C:\Users\Fergus\AppData\Local\Repstor\Office 15 Profile - Outlook\C\00000001\Extensibility Guide.docx
Replify.LaunchedWriteTime	DateTime	Last time an edited document was written to disk. 17:49, 04/08/2014
Replify.LaunchedProcessKey	String	The process ID of the process currently editing the document. Affinity will detect this process closing in order to decide when to synchronize the edit to the server. 11112-
For items currently being synchronized		
ROAM.BeingSynced	Number	Item is currently undergoing synchronization

ROAM.OpenByProcess	Number	Item currently is part of an edit property dialog, or view document launch.

Typical affinity properties (for SharePoint repositories)

Repstor affinity will synchronize all properties related to an item regardless of the content type. All these properties are synchronized as string properties. Properties are renamed from the field names in SharePoint according to the following rules.

- Any occurrence of the "_" character is replaced with "-"
- Any occurrence of "-" is replaced with "—"

See Appendix A for a set of example document properties.

Repository Properties

When Repstor affinity synchronizes a repository it will store additional properties related to the overall repository (regardless of the sub folders). Repository properties will include the standard folder properties described above. Additionally at the repository level, the following properties will be synchronized.

Standard Repstor properties (any repository type)

Property Name	**Type**	**Description and Example Value**
Replify.SyncDIsabled	Boolean	Whether the synchronization for this whole repository is disabled or not
Replify.MainFolder	Boolean	True for a top level repository folder. (not set on sub folders, or on parent folders)
ROAM.RepositoryIsRequired	Boolean	A repository that cannot be removed by the user
ROAM.TemporaryRepository	Boolean	True if this repository should be cleared out at the end of the outlook session. Note this will not force synchronization of pending items, which will be lost.
Replify.ManuallyCreated	Boolean	True if the repository was added by the user (as opposed to added through central repositories)

Typical Repstor properties (SharePoint repository type)

Property Name	**Type**	**Description and Example Value**
CT-Defaults, CT-Info, CT-0x…	String	Content type information for the repository

RS-UG	String	User and group information for the repository.
SPListXML, SPViewsXML	String	Information on the list and view of the repository.

Changing Properties

Changing the property value itself within Outlook will not trigger a change within the server repository, it is also necessary to mark the property as changed within a special Replify.ChangedFields string property

Change tag format:

\n<fieldName>\t<basicName>\t<fieldType>\n

Where:

- **fieldName** is the internal field name of the SharePoint field. (ows-Title)
- **basicName** is the display name of the SharePoint field. (Title)
- **fieldType** is a number reflecting the type of field (it is only important that this is set to 12/13 for user and multi user fields respectively)
- **\n** is the newline character
- **\t** is the tab character

For example, to change the name of a document you would set:

\nows-BaseName\tName\t1\n

These values must be appended together for multiple field changes.

Using Properties in custom Outlook Solutions

The properties described above are available on the folder and message property sets that are used within Repstor affinities extensibility mechanisms. When using other technologies (like VSTO) it the message property sets are part of the Outlook message items, but the folder and repository properties are part of a folder associated message “LSFolder.CustomProps”.

User Interface Extensibility

Repstor 2.4 introduced new functionality for extending the Repstor affinity user interface with custom actions. It also introduced various extensibility points at key points within the product. This functionality is only available in Outlook 2010 and above.

The extensibility mechanism uses VBScript functions defined within an VBScript file starting with the name "Extensions" and having the extension "VBS" in a Scripts folder beneath the Repstor affinity installation directory. (The Scripts folder, and Extensions*.vbs file must be created)

Once the extensions file is in place, the extensions are enabled.

Extensions file should be named appropriately to the extension they are providing. For example Extensions – Sharing.vbs could provide extensions based on new sharing capability.

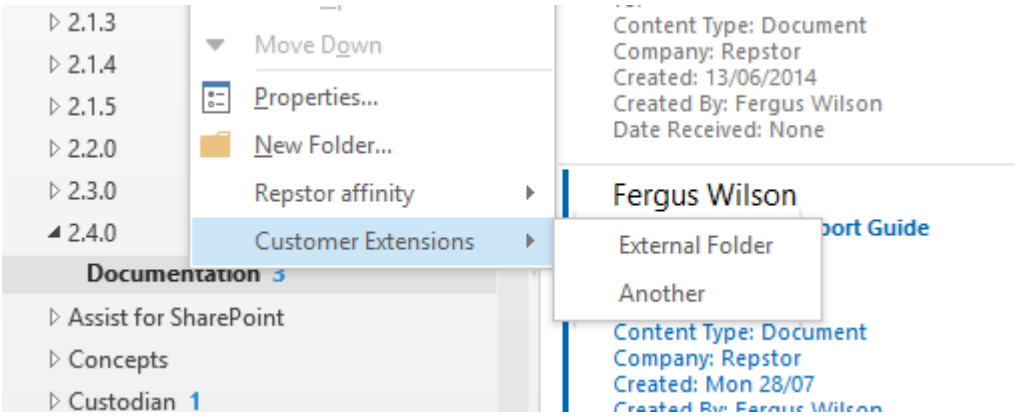
Function names should be used to avoid clashes between Extension files. Repstor will look for the main function names lists below, but there can only be one of each function name. To avoid clashes, you can also append the name of the extensions file onto the function.

e.g. instead of **RepstorExt_FolderExtName** as a function name you should use **RepstorExt_FolderExtName_Sharing** if the function is contained in an extension file called "Extensions – Sharing.vbs"

Custom Actions

Folder Actions

Folder actions appear on the context menu of folders within the Repstor affinity message store.



The name of parent menu for folder actions is given by the RepstorExt_FolderExtName function as shown below.

```
Function RepstorExt_FolderExtName()
	RepstorExt_FolderExtName = "Customer Extensions"
End Function
```

Since Repstor 3.1.2, the FolderExtName function can return a comma separated set of folder names, to allow for two different sets of folder extensions.

When the context menu is to be displayed, affinity calls on the function RepstorExt_FolderVisibleActions, and RepstorExt_NamedFolderVisibleActions to calculate which menu items should be displayed. The function can look at the properties of the selected folder in the PropMap parameter passed into it. See below for a description of the PropMap parameter. The value returned is a comma separated list of folder actions to be displayed in the menu.

```
Function RepstorExt_FolderVisibleActions(PropMap)
	RepstorExt_FolderVisibleActions = "External Folder,Another"
End Function
Function RepstorExt_NamedFolderVisibleActions(FolderGroup, PropMap)
	If (FolderGroup = "Workflow") Then
		RepstorExt_NamedFolderVisibleActions = "Folder Export,Archive"
	End If
	If (FolderGroup = "Collaboration") Then
		RepstorExt_NamedFolderVisibleActions = "Create Deal Room,Share"
	End If

End Function
```

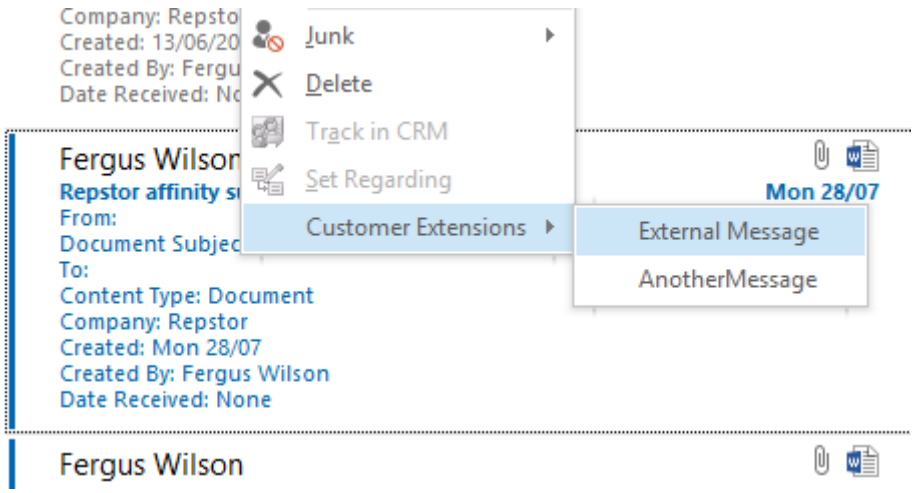
If a user selects one of the custom folder items, then the RepstorExt_FolderAction function is called. Note that Actions need to be unique across any folder groups:

```
Sub RepstorExt_FolderAction(Action, FolderPropMap)
	Set WshShell = CreateObject("WScript.Shell")
	WshShell.Run "http://www.repstor.com/" & "?" & URLEncode(Action)
End Sub
```

The folder action sub routine is passed the name of the action the user selected (as returned in the visible actions call), and the property map of the folder currently selected.

Message and Document Actions

Message and document actions work in a very similar way to the folder actions – they appear on the context menu of messages and documents within the Repstor affinity message store.



The name of parent menu for message actions is given by the RepstorExt_MessageExtName function as shown below.

```
Function RepstorExt_MessageExtName()
	RepstorExt_MessageExtName = "Customer Extensions"
End Function
```

Since 3.1.2, this can be a comma separated set of context menus. When the context menu is to be displayed, affinity calls on the function RepstorExt_MessageVisibleActions and RepstorExt_NamedMessageVisibleAction to calculate which menu items should be displayed. The function can look at the properties of the selected folder and the selected message in the FolderPropMap and MessagePropMap parameters passed into it. See below for a description of these parameters. The value returned is a comma separated list of message actions to be displayed in the menu.

```
Function RepstorExt_MessageVisibleActions(FolderPropMap, MessagePropMap))
	RepstorExt_MessageVisibleActions = "External Message,AnotherMessage"
End Function

Function RepstorExt_NamedMessageVisibleActions(ExtName, FolderPropMap,
MessagePropMap))
	If (ExtName = "Workflow") Then
		RepstorExt_NamedMessageVisibleActions = "Document Review,Submit as
Record"
	End If
	If (ExtName = "Collaboration") Then
		RepstorExt_NamedMessageVisibleActions = "Share"
	End If
End Function
```

If a user selects one of the custom message items, then the RepstorExt_MessageAction function is called.

The action names must be unique across all message extension groups.

```
Sub RepstorExt_MessageAction(Action, FolderPropMap, MessagePropMap)
	Set WshShell = CreateObject("WScript.Shell")
	dim strTitle
	strTitle = MessagePropMap.ItemStr("0x0037001F")
	If (FolderPropMap.HasKey("Replify.RepositoryType")) Then
		dim reposType
		reposType = FolderPropMap.ItemStr("Replify.RepositoryType")
		If (reposType = "WorkSite") Then
'			WshShell.Run "http://www.repstor.com/" & "?type=WS&action=" &
URLEncode(Action)  & "title=" & UrlEncode(strTitle)
		End If
		If (reposType = "SharePoint") Then
'			WshShell.Run "http://www.repstor.com/" & "?type=SP&action=" &
URLEncode(Action)  & "title=" & UrlEncode(strTitle)
		End If
	End If
End Sub
```

The folder action sub routine is passed the name of the action the user selected (as returned in the visible actions call), and the property map of the folder currently selected.

Global Actions

Global actions are always displayed in the ribbon bar. These actions cannot be hidden however the RepstorExt_IsButtonEnabled method can be optionally used to decide if they are enabled for the current selection.

The ribbon groups are determined by a call to RepstorExt_RibbonButtonGroupName. From 3.1.2 this can return a comma separated set of ribbon group names.

```
Function RepstorExt_RibbonButtonGroupName()
	RepstorExt_RibbonButtonGroupName = "Repstor custodian,DMS"
End Function
```

The function RepstorExt_RibbonButtons or RepstorExt_NamedRibbonButtons can be used to determine what buttons to display.

```
Function RepstorExt_RibbonButtons()
	RepstorExt_RibbonButtons= "Create Matter"
End Function

Function RepstorExt_NamedRibbonButtons(groupName)
   if (groupName = "Repstor custodian") Then
	RepstorExt_NamedRibbonButtons = "My Matters,Create Matter,All Matters"
   End if
```

```
    if (groupName = "DMS") Then
        RepstorExt_NamedRibbonButtons = "Report"
    End if
End Function
```

Then the sub routine RepstorExt_CallRibbonButton is used when the user clicks on the button. All button names must be unique across ribbon groups. The folderPropMap argument has the general Outlook Folder properties, folderCustomProp has custom folder properties only when the selected folder is a Repstor affinity folder.

```
Function RepstorExt_CallRibbonButton(ButtonName, folderPropMap, folderCustomProp,
itemPropMap)
 ' Do something here
 ' itemPropMap is nothing if there's no message selected
 ' folderCustomProp is nothing if it's a non Repstor folder
End Function
```

The sub routine can be optionally implemented to disable the actions depending on the current folder or message selection. If this method is missing then all actions are always enabled.

```
Function RepstorExt_IsButtonEnabled(buttonName, folderPropMap, folderCustomProp,
itemPropMap)
        ' itemPropMap is nothing if there's no message selected
        if (itemPropMap is Nothing) then
                RepstorExt_IsButtonEnabled = false
                Exit function
        end if
        ' folderCustomProp is nothing if it's a non Repstor folder
        if (folderCustomProp is nothing) then
                RepstorExt_IsButtonEnabled = true
        else
                RepstorExt_IsButtonEnabled = true
        end if
end function
```

Property Page Extensibility

The property pages displayed on Repstor affinity messages can be configured to determine the exact property set to display at a time.

By default, the properties contained within an item's content type are displayed along with any error information.

The RepstorExt_PropertyPageVisibleProps function can be used to limit or expand the properties displayed at a particular time. If this function returns an empty string, then the default functionality is used – the properties of the content type will be displayed.

```
Function RepstorExt_PropertyPageVisibleProps(MessagePropMap)
	RepstorExt_PropertyPageVisibleProps = "ows-FileLeafRef,ows-Title,ows-Editor,ows-FileSizeDisplay,ows--dlc-DocId,ows--dlc-DocIdUrl,ows-MetaInfo"
End Function
```

If the fields are to be expanded beyond those within the item's content type, it is necessary to call another function to indicate to Repstor that additional properties may be included in the property page.

```
Function RepstorExt_PropertyPageAllFieldsNeeded(MessagePropMap)
	RepstorExt_PropertyPageAllFieldsNeeded = True
End Function
```

New Message Extensions

When a new item is added to Repstor affinity, affinity will calculate the properties based on a combination of defaults and existing properties. If any mandatory properties are not set, affinity will display a property dialog and ask the user to complete the properties.

The RepstorExt_NewMessage function gives a solution the opportunity to assign properties during this process, and avoid prompting the user for properties.

```
Sub RepstorExt_NewMessage(MessagePropMap)
	' do something based on content type
	dim strTitle
	strTitle = MessagePropMap.ItemStr("ows-ContentType")

	' Set a property
	MessagePropMap.ItemStr("ows-Title") = "my new title"

	' Set a prompt message to the user
	MessagePropMap.ItemStr("Replify.SyncIssueMessage") = "Make sure to fill all properties in!"
End Sub
```

The RepstorExt_PreUpload function is called just before upload. This is after all client side prompting and setting of properties has been completed. It provides an opportunity to change the name of the uploaded item.

```
Sub RepstorExt_PreUpload(IsNewItem, MessagePropMap)
	If (IsNewItem) Then
		fileName = MessagePropMap.ItemStr("Replify.FromServerFileName")

		pos = InStrRev(fileName, ".")
		baseFileName = Left(fileName, pos - 1)
		ext = Mid(fileName, pos + 1)

		curDate =  Year(Date) & "_" & Month(Date) & "_" & Day(Date) & "_" &
Replace(Time, ":", "_")
		MessagePropMap.ItemStr("Replify.FromServerFileName") = curDate & " " &
baseFileName & "." & ext
	End If
End Sub
```

In 3.7.1 Repstor affinity, new extensions were added to allow further control over messages and documents added by the user. These extensions are called at the very start of the process, before any prompts for properties have occurred.

The RepstorExt_UserAddedItems is called when one or more documents or emails is added to the system. It is called once for a single batch of documents. The MessagePropMap passed in is the first message in the batch. The function must return true if subsequent calls to RepstorExt_UserAddedItem are required.

```
Dim strAuthor

Function RepstorExt_UserAddedItems (FolderPropMap, MessagePropMap, Num)
	if (MessagePropMap.HasKey("Author")) Then
		RepstorExt_UserAddedItems  = false
		Exit Function
	end if
	strAuthor = repdocs.ShowPromptDialogWithCancel("Repstor", "Enter a value for
Author", bCancel)
	RepstorExt_UserAddedItems  = true

End Function
Sub RepstorExt_UserAddedItem (FolderPropMap, MessagePropMap)
	MessagePropMap.ItemStr("Author") = strAuthor
	' Should mark the author property as a changed field
End Sub
```

New Synchronized Message Extensions

When a new item is synchronized from a repository to Repstor affinity, affinity will store the properties of the item it retrieves from the source repository. There are two extensions possible. The first allows you to add new properties or adjust the properties of the item being synchronized.

The RepstorExt_NewSyncedMessage function gives a solution the opportunity to store additional properties during this process. This can be used to adjust properties for display in the Repstor property panels or for adding additional properties needed through extensibility.

```
Sub  RepstorExt_NewSyncedMessage ( FolderPropMap, MessagePropMap)
       ' Set a property
       MessagePropMap.ItemStr("NeedThisPropertyInOutlook") = "OurCustomProperty"
End Sub
```

The second extensibility point function RepstorExt_PostNewSyncedMessage is called after the item has been synchronized, and saved in the Repstor affinity store. This allows you to perform additional functionality

```
Sub RepstorExt_PostNewSyncedMessage  ( FolderPropMap, MessagePropMap)
       ' Perform some client functionality related to new messages
End Sub
```

New Synchronized Folder Extensions

When a new sub folder is synchronized from a repository to Repstor affinity, affinity will store the properties of the item it retrieves from the source repository. This allows you to add new properties or adjust the properties of the item being synchronized. This may be useful to give a folder a friendlier name (perhaps based on other properties than the main folder name in the repository). Care should be taken to when renaming is allowed on these folders. The renamed value will include the friendly name if the folder name is changed.

The extension is passed the folder property map (which normally includes the repository properties associated with the folder) and the actual folder property map. The actual folder property map gives access to the Outlook folder object, and can be used to change the display name of the folder, as shown in the example.

This extension must have the NewSyncFolderExt registry settings enabled before it will be called.

```
Sub RepstorExt_NewSyncedFolder(FolderPropMap, ActualFolderPropMap)
       dim strTitle
       strTitle = ActualFolderPropMap.ItemStr("0x3001001F")

       dim strId
```

```
        strId = FolderPropMap.ItemStr("ows-ID")

        ActualFolderPropMap.ItemStr("0x3001001F") = strTitle & " (" & strId & ")"
End Sub
```

When a new Parent folder (not a repository folder, but a virtual grouping folder above the repository folder) is created, you can call a different extension. This extension can be used to set additional properties that will display folder URL pages or allow right click Open Location values to be set.

An example is below:

```
Sub RepstorExt_NewSyncedParentFolder(FolderPropMap, ActualFolderPropMap)

        dim dispName
        dispName = ActualFolderPropMap.ItemStr("0x3001001f") ' folder name
    FolderPropMap.ItemStr("ROAM.FolderWebUrl") = "http://www.google.com?q=" &
dispName ' sets a url to be displayed in preview pane when folder is selected

        FolderPropMap.ItemStr("Replify.FolderLocationUrl") = "http://www.yahoo.com"
                        ' URL available as right click Repstor/Open Location
End Sub
```

New Document Menu Extensibility

Repstor also has the ability to display a drop down menu to allow a user to select a brand new document. This functionality is not normally available within the interface, but can be added as part of the extensibility.

```
Function RepstorExt_NewDocumentMenuItems(FolderPropMap)
     RepstorExt_NewDocumentMenuItems = "templatedocs,doc1,doc2,doc3"
End Function
```

For this functionality to work correctly, the template documents (doc1, 2, 3 in the above example) must exist in a document library synchronized with the name “templatedocs”, which has been synchronized to the hidden folder area of Repstor (parent folders group name: _ROAMHidden). This document library must be synchronized explicitly with full item downloads.

Licensing Extensibility

There is one extensibility point that is called when a license is activated for the first time. This is

```
Sub RepstorExt_Activated()
     Repsup.DiagLog "We are off!"
End Sub
```

This extensibility point can be used to add new repositories or customize an installation on first use.

Outlook Startup and Message Store Extensibility

Extensions allow a custom solution to take action when Outlook starts, is closed, or if the Repstor affinity message store is added for the first time.

```
Sub RepstorExt_MessageStoreAdded ()
     repsup.ImportRepositoryList "u:\InitialReposList.tsv"
End Sub


Sub RepstorExt_OutlookOpening ()
     repsup.LogDiag "Outlook opened"
End Sub


Sub RepstorExt_OutlookClosing ()
     repsup.ExportRepositoryList "q:\MyReposList.tsv"
End Sub
```

Repository addition and removal Extensibility

Extensions can be called when repositories are added and removed through Repstor affinity.

The AddRepositoryExtension configuration must be enabled to receive the repository added event.

```
Function RepstorExt_RepositoryBeingAdded (FilterName, DefaultName, Location,
ParentGroup)
    ParentGroup = "DMS:" & ParentGroup
     RepstorExt_RepositoryBeingAdded = true
End Function
```

```
Function RepstorExt_RepositoryBeingRemoved ( NormalizedUrl )
     If (InStr(NormalizedUrl, "Policies") > 0) Then
         RepstorExt_RepositoryBeingRemoved = False
     Else
         RepstorExt_RepositoryBeingRemoved = True
```

```
    End If
End Function
```

Custom stssync Extensibility

This powerful extension lets you provide links on web pages that can call custom functionality you define within an extension. The stssync link is of the following format:

stssync://roam/ext=<str>

the <str> string is passed as a parameter to the extension function.

```
Function RepstorExt_RunExtensionAction (BSTR str)
    …
End Function
```

Detecting Usage

This extension call can be used to detect whether the user is clicking on any Repstor folders. It can be used to measure product usage.

```
Function  RepstorExt_FolderBrowse (FolderPropMap)
    …
End Function
```

Custom Field Validation

This extension call can be used to perform custom validation of fields on a property form.

```
Function   RepstorExt_CustomFieldValidation(FieldName, MessagePropMap)
    … ' check field
     RepstorExt_CustomFieldValidation = "You need to set a value of format XX/1
for this field"
End Function
```

Luggage Tag Filing

This extension call is used to determine the folder to file to, when a luggage tag is detected on an incoming message. This extension will override the standard functionality which looks for a luggage

tag using the quick file index. This functionality is only available when a license for Repstor assist is available.

```
Function  RepstorExt_GetAutoFileFolderForNewMessage(IncomingMessagePropMap, LuggageTag)
       RepstorExt_GetAutoFileFolderForNewMessage = "\\Repstor affinity\Folder\FileHere"
End Function
```

Applying properties to parent folders

This extension call is used to apply properties to parent folders when a child folder is synchronized. This can be used for example, to apply properties associated with a matter to the parent matter folder in affinity.

```
Function RepstorExt_ApplyReposDataToParent(FolderName, Url, ParentFolders, CentralReposUrl)
        RepstorExt_ApplyReposDataToParent = True
End Function
```

PropertyMap Interfaces

The property map interfaces give access to the properties on the folders and messages that were introduced earlier in this document. The interface is defined fully in Appendix B.

The interface is a read/write interface – properties can be set and retrieved. However, it is necessary to save properties that are set – so that Outlook saves the changes to the underlying message store. The NewMessage and the New message and folder synchronized extensions above are the only calls for which the saving of properties is guaranteed.

Extensibility Debugging

Care should be taken when developing extensions to Repstor. By default, errors raised in extension functionality are raised to the calling functionality, and in extreme cases could lead to Outlook crashing. It is possible to configure Repstor so errors are not raised by setting the DWORD registry setting: Extension.ThrowOnScriptError to 0.

By setting configuration entry BalloonForScriptErrors or turning on diagnostics, extension errors will be presented as balloon popups.

Errors within the scripts are always logged through the standard Repstor diagnostic log.

Extensibility Support Object

Within the extensibility scripts, the Repstor extensibility support object is available to obtain information and perform actions within the main affinity product. This can help with diagnostics, but can also provide more complete and seamless extensibility functionality.

See *Appendix C – Extension Support Object* for the definition of the support object.

The object can be used from any of the extensibility functions, and is effectively a callback into the Repstor product to either perform functions, or to provide information to the user.

```
Function RepstorExt_CallRibbonButton(ButtonName, …)
 Repsup.BalloonPopup "Ribbon Button", ButtonName & " has been pressed"
 Repsup.LogDiag "We have told the user that the button was pressed"
End Function
```

The functions available are:

LogDiag	Send a diagnostic message to the standard logs
LogError	Send an error message to the standard logs
MessagePopup	Display a popup message which must be OKed by the user
BalloonPopup	Display a balloon popup in the taskbar
AddReposWithUI	Add a new repository by prompting user to confirm details
AddReposNoUI	Add a new repository, without prompting user
RefreshRequiredRepositories	Refresh the current central repositories list
SyncFolderPath	Synchronize a specific folder according to Outlook folder path.
HasLicenseFor	Check if the current license covers some capability – currently: SPS_FILTER = 0, FILESHARE_FILTER = 1, CATEGORIZATION = 2, IMANAGE_FILTER = 4, TRIM_FILTER = 5, NEEDS_ACTIVATION = 6, CUSTODIAN = 7, CRM_LINK = 8, MINIMAL_CLIENT = 9, PROVISIONING_ENGINE = 10,

	REPSTOR_DRIVE = 11, HIGHQ = 12, MERIDIO = 13
SetUserStringConfig GetUserStringConfig SetUserDWORDConfig GetUserDWORDConfig	Set and get configuration of Repstor. Getting user configuration will get configuration from user registry, but will also check any local or group policy settings. Setting will only set the user's configuration.
SetUserProfileStringConfig GetUserProfileStringConfig	Set and get string configuration in the user registry, beneath an area specific to the current Outlook profile.
ShowBrowserDialog	Bring up a modal window displaying a URL. And check the dialog response.
ShowCustomDialog	Bring up a new style dialog, with a message, custom buttons with descriptions, and a default button
ShowPromptDialog	Prompt the user for some text
ShowPromptDialogWithCancel	Prompt the user for some text, return if cancel is pressed
ShowOptionsDialogWithCancel	Prompt the user for a selection of options. Return if cancel is pressed
ShowConfirmDialogWithCancel	Show yes/no dialog – return cancel if user selects No.
BrowseFoldersDialog	Launch a dialog asking the user to select a folder
RepstorBuildNumber	Return the Repstor build number
GetUserLang	Get the users language string e.g. ”en” or “de”
ImportRepositoryList	Import a repository list from a file path
ExportRepositoryList	Export the repository list to a path
PendingItemsCount	Return the number of items not synchronized to the server repository
ErrorItemsCount	Return the number of items that haven a error associated with them
LocalEditItemsCount	Return number of items currently being edited

RemoveRepository	Remove a repository
GetRepositoryList	Get the entire repository list
IsExistingRepository	Return if a URl is for an existing repository
FindExistingRepositoryPath	Find the folder path for a repository
FindFolderPathProps	For a given folder path, return the folder properties
JumpToFolder	Jump to a folder in Outlook
OpenFolderItem	Open an item from a folder
FindFolderByProp	Find a folder using a folder property (uses quick file index)
GetDigestValue	Get the SharePoint digest value to be used in SharePoint REST calls
UrlEncode	Encode a URL
UrlDecode	Decode a URL
CreateAndLaunchPrecedentInFolderPath	Look up the hidden precedent document library, and create a document from it within the given folder path
QueuePendingItemFolders	Queue folders which contain items still pending
TouchPendingItems	Mark items as changed that are pending items
GetFolderRepositoryMap	Get a repository map for a given folder property map
GetParentFolderMapForMessage	Get the parent map for a given message property map
GetParentFolderMapForFolder	Get the parent folder property map for a given folder property map
CopyItemToFolder	Copy a message from one folder to another
SavePropertyMap	Save the values of a changed property map
GetFilterTextForMsg	(Assist only) determine the text that is in a message and its attachments
LoadCustomRuleSet	(Assist only) Load or Create a custom ruleset

	within Assist
`CustomRuleSetAddText`	(Assist only) Add some text to a custom ruleset with the given tag
`CustomRuleSetGetTag`	(Assist only) Return the best tag from a ruleset for the given text.
`SetFieldByDisplayValue`	Set a field using a display value. This call will assign appropriate change values to the message
`GetFieldDisplayValue`	Get the current value for display for a given field
`GetFolderPath`	Get the folder path for a given folder property map
`GetMainWindowHandle`	Get the main window handle for the application

Other Property Page Customization Techniques

As well as the VBScript extensibility mechanism for changing the property page, there are a couple of other ways to change how the property page looks.

Property pages are displayed using Outlook's Form Region functionality. The standard property pages are contained on a form region called MultiItemFormRegion. For list items, and for documents as a separate property page there is also a CustomListAdj and CustomList formregion page respectively.

Each form region uses a property panel (LSSImplePropertyGrid) to display the correct properties on the form with the right labels etc. This property panel can be removed and replaced with property fields directly on the form, or can be configured to add and remove properties.

Adding and Removing properties from the property grid.

The property grid has four properties which can be changed. Once changed, the form region can be re-saved to the Repstor affinity "FormRegion" folder.

Property	Description
FieldsToDisplayCSV	The list of fields that will always be displayed
FieldsToSkipCSV	The list of fields that will always be skipped
MaxFieldsToShow	Maximum number of fields to display on the form
ShowVisibleFields	Whether or not to show the visible fields in the content type. This defaults to true.

Fields displayed must either be part of the selected item's ContentType – or the RepstorExt_PropertyPageAllFieldsNeeded extension above used to include all fields in the property form display.

Replacing the Property Grid with direct fields

The property grid can be replaced with Repstor affinity properties provided directly on the form region. Repstor affinity properties are defined in a set of activex controls. The most common control is LSROAMSimpleField, but there are a variety of other controls for displaying the different types of SharePoint properties. Each control has a set of standard properties:

InnerName – the SharePoint property name to be displayed.

Advanced custom solutions can add new property fields that implement functionality around custom fields – or implement advanced user interfaces within the Repstor affinity property panel. Contact Repstor support (support@repstor.com) to find out more about that functionality.

Adding new Repository Types

Repstor affinity repository types use a standard plugin style mechanism based on the COM object model. It is possible to add new repository types without requiring a new version of Repstor affinity. Currently adding a new repository type is beyond the scope of this extensibility guide. Contact Repstor support (support@repstor.com) for help if you would like to support a custom repository type.

Example Extension File

```
Function Base64Encode(sText)
    Dim oXML, oNode

    Set oXML = CreateObject("Msxml2.DOMDocument.3.0")
    Set oNode = oXML.CreateElement("base64")
    oNode.dataType = "bin.base64"
    oNode.nodeTypedValue = Stream_StringToBinary(sText)
    Base64Encode = oNode.text
    Set oNode = Nothing
    Set oXML = Nothing
End Function

Function Stream_StringToBinary(Text)
  Const adTypeText = 2
  Const adTypeBinary = 1

  'Create Stream object
  Dim BinaryStream 'As New Stream
  Set BinaryStream = CreateObject("ADODB.Stream")
```

```
  'Specify stream type - we want To save text/string data.
  BinaryStream.Type = adTypeText

  'Specify charset For the source text (unicode) data.
  BinaryStream.CharSet = "us-ascii"

  'Open the stream And write text/string data To the object
  BinaryStream.Open
  BinaryStream.WriteText Text

  'Change stream type To binary
  BinaryStream.Position = 0
  BinaryStream.Type = adTypeBinary

  'Ignore first two bytes - sign of
  BinaryStream.Position = 0

  'Open the stream And get binary data from the object
  Stream_StringToBinary = BinaryStream.Read

  Set BinaryStream = Nothing
End Function

Function URLEncode(ByVal str)
 Dim strTemp, strChar
 Dim intPos, intASCII
 strTemp = ""
 strChar = ""
 For intPos = 1 To Len(str)
  intASCII = Asc(Mid(str, intPos, 1))
  If intASCII = 32 Then
   strTemp = strTemp & "+"
  ElseIf ((intASCII < 123) And (intASCII > 96)) Then
   strTemp = strTemp & Chr(intASCII)
  ElseIf ((intASCII < 91) And (intASCII > 64)) Then
   strTemp = strTemp & Chr(intASCII)
  ElseIf ((intASCII < 58) And (intASCII > 47)) Then
   strTemp = strTemp & Chr(intASCII)
  Else
   strChar = Trim(Hex(intASCII))
   If intASCII < 16 Then
    strTemp = strTemp & "%0" & strChar
   Else
    strTemp = strTemp & "%" & strChar
   End If
  End If
 Next
```

```
 URLEncode = strTemp
End Function

Function RepstorExt_FolderExtName()
	RepstorExt_FolderExtName = "Customer Extensions"
End Function

Function RepstorExt_FolderVisibleActions(PropMap)
	RepstorExt_FolderVisibleActions = "External Folder,Another"
End Function



Sub RepstorExt_FolderAction(Action, FolderPropMap)
'	Set WshShell = CreateObject("WScript.Shell")
'	WshShell.Run "http://www.repstor.com/" & "?" & URLEncode(Action)
End Sub

Function RepstorExt_MessageExtName()
	RepstorExt_MessageExtName = "Customer Extensions"
End Function

Function RepstorExt_MessageVisibleActions(FolderPropMap, MessagePropMap)
	RepstorExt_MessageVisibleActions = "External Message,AnotherMessage"
End Function

Sub RepstorExt_MessageAction(Action, FolderPropMap, MessagePropMap)
	Set WshShell = CreateObject("WScript.Shell")
	dim strTitle
	strTitle = MessagePropMap.ItemStr("0x0037001F")

	If (FolderPropMap.HasKey("Replify.RepositoryType")) Then

		dim reposType
		reposType = FolderPropMap.ItemStr("Replify.RepositoryType")

		If (reposType = "WorkSite") Then
'			WshShell.Run "http://www.repstor.com/" & "?type=WS&action=" &
URLEncode(Action)  & "title=" & UrlEncode(strTitle)
		End If

		If (reposType = "SharePoint") Then
'			WshShell.Run "http://www.repstor.com/" & "?type=SP&action=" &
URLEncode(Action)  & "title=" & UrlEncode(strTitle)
		End If
	End If
End Sub
```

```
Function RepstorExt_PropertyPageVisibleProps(MessagePropMap)
	If (MessagePropMap.HasKey("Replify.DocumentLaunched")) Then
		RepstorExt_PropertyPageVisibleProps = "ows-FileLeafRef,ows-Title,ows-
Editor,ows-FileSizeDisplay,ows--dlc-DocId,ows--dlc-DocIdUrl,ows-MetaInfo,ows--
CheckinComment"
	Else
		RepstorExt_PropertyPageVisibleProps = "ows-FileLeafRef,ows-Title,ows-
Editor,ows-FileSizeDisplay,ows--dlc-DocId,ows--dlc-DocIdUrl,ows-MetaInfo"
	End If

End Function

Function RepstorExt_PropertyPageAllFieldsNeeded(MessagePropMap)
	RepstorExt_PropertyPageAllFieldsNeeded = True
End Function

Function RepstorExt_NewMessage(MessagePropMap)
'	dim strTitle

'	strTitle = MessagePropMap.ItemStr("ows-ContentType")
'	MessagePropMap.ItemStr("ows-Title") = "Always the title"
'	Set WshShell = CreateObject("WScript.Shell")
'	WshShell.Run "http://www.repstor.com/" & "?" & URLEncode(strTitle)
	MessagePropMap.ItemStr("Replify.SyncIssueMessage") = "Make sure to fill all
properties in"
End Function

Function RepstorExt_RibbonButtons()
	RepstorExt_RibbonButtons= "Create Matter"
End Function

' itemPropMap is nothing if there's no message selected
' folderCustomProp is nothing if it's a non Repstor folder
Function RepstorExt_CallRibbonButton(ButtonName, folderPropMap, folderCustomProp,
itemPropMap)
	Set WshShell = CreateObject("WScript.Shell")
	WshShell.Run
"https://custodian.azurewebsites.net/Matter/Create?SPHostUrl=https%3A%2F%2Frepstor%
2Esharepoint%2Ecom%2Fsites%2Fcage"
End Function

' decide whether to enable the ribbon buttons
function RepstorExt_IsButtonEnabled(buttonName, folderPropMap, folderCustomProp,
itemPropMap)
	' itemPropMap is nothing if there's no message selected
	if (itemPropMap is Nothing) then
		RepstorExt_IsButtonEnabled = false
		Exit function
```

```
	end if
	' folderCustomProp is nothing if it's a non Repstor folder
	if (folderCustomProp is nothing) then
		RepstorExt_IsButtonEnabled = true
	else
		RepstorExt_IsButtonEnabled = true
	end if
end function

Function RepstorExt_NewDocumentMenuItems(FolderPropMap)
	Dim matterUrl, matterUrlBase64, spHostUrl, serviceUrl, url,
spHostUrlEncoded, serviceUrlEncoded, strPostData, formAction, form, clientId,
xmlDoc, precedentElems, xmlReq, odoc, inputs, input, precedentTitle
	Dim resultArray()


 matterUrl = FolderPropMap.ItemStr("Replify.FolderLocation")

	'DEBUG
	spHostUrl = "https://repstor.sharepoint.com/sites/devc"
	clientId = "7e2c1ea1-926f-4bbc-846f-747397c708d4"
	serviceUrl = "https://localhost:44307"

	matterUrlBase64 = Base64Encode(matterUrl)
	spHostUrlEncoded = URLEncode(spHostUrl)
	serviceUrlEncoded = URLEncode(serviceUrl)

	'Build URL
	url = spHostUrl & "/_layouts/15/AppRedirect.aspx?client_id="& clientId
&"&redirect_uri="& serviceUrlEncoded &"%2FAjax%2FGetPrecedentsForMatter%2F"&
matterUrlBase64 &"%3F%7BStandardTokens%7D%26SPHasRedirectedToSharePoint%3D1"

	'Send the HTTP request
	Set xmlReq = CreateObject("MSXML2.XMLHTTP")
	call xmlReq.Open("GET", url, False)
	call xmlReq.send()

	'Create the HTML document
	Set odoc = CreateObject("HTMLFILE")
	odoc.write xmlReq.responseText

	'Get the form object
	Set form = odoc.GetElementById("frmRedirect")
	If Not form Is Nothing Then

		'Get the URL from the form
		formAction = form.GetAttribute("action")
```

```
            'Get the post variables
            strPostData = ""
            Set inputs = form.getElementsByTagName("input")
            For Each input In inputs
                  strPostData = strPostData & input.GetAttribute("name") & "=" &
input.GetAttribute("value") & "&"
            Next

            strPostData = strPostData & "SPVisited=1"

            'Create a new HTTP Request to post the form
            Set xmlReq = CreateObject("MSXML2.XMLHTTP")
            With xmlReq
              .Open "POST", formAction, False
              .setRequestHeader "Content-Type", "application/x-www-form-
urlencoded"
              .send (strPostData)
            End With

            'Get the result in an XML document
            Set xmlDoc = CreateObject("Microsoft.XMLDOM")
            xmlDoc.loadXML(xmlReq.responseText)

            'Go through each precedent and get the title and add to result array
            Set precedentElems = xmlDoc.getElementsByTagName("FileLeafRef")

            Dim title
            Set fso = CreateObject("Scripting.FileSystemObject")
            Redim resultArray(precedentElems.length + 1)
            resultArray(0) = "precedentdocs"
            For i = 1 To precedentElems.length
                  title = precedentElems(i-1).Text

                  'Remove the file extension
                  resultArray(i) = Replace(title,"." &
fso.GetExtensionName(title),"")
            Next
      End If

      'Join the array into a comma separated list for returning
      RepstorExt_NewDocumentMenuItems = Join(resultArray, ",")
End Function
```

Appendix A – Example Properties (SharePoint)

Example Document Properties

An example property set is shown below. Note that each document's properties will depend on the type of the document and on the SharePoint configuration.

Property Name	Example Value
ows-ContentTypeId	0x0101002C8B7E5E5A7AD544818E7C987310B217
ows-Title	Extensibility Guide
ows-ID	5840
ows-ContentType	Document
ows-Modified	2014-08-01T09:02:59Z
ows-Created	2014-06-13T18:11:49Z
ows-Author	Fergus Wilson
ows-Editor	Fergus Wilson
ows-owshiddenversion	9
ows-WorkflowVersion	1
ows—UIVersion	2560
ows—UIVersionString	5.0
ows—ModerationStatus	0
ows-SelectTitle	5840
ows-Order	27200.0000000000
ows-GUID	{E52CBBE0-080E-4502-AA1B-1DB136F34C0E}
ows-FileRef	Shared Documents/Releases/2.4.0/Documentation/Extensibility Guide.docx
ows-FileDirRef	Shared Documents/Releases/2.4.0/Documentation
ows-Last-x0020-Modified	2014-08-01T09:03:00Z
ows-Created-x0020-Date	2014-06-13T18:11:49Z
ows-FSObjType	0

ows-SortBehavior	0
ows-PermMask	0x7fffffffffffffff
ows-FileLeafRef	Extensibility Guide.docx
ows-UniqueId	{0B7A462D-BC4C-45AA-A714-01491214007D}
ows-ProgId	
ows-ScopeId	{7D308A85-3398-4E1F-B8E8-F13AEA56EEAA}
ows--EditMenuTableStart	Extensibility Guide.docx
ows--EditMenuTableStart2	5840
ows—EditMenuTableEnd	5840
ows-LinkFilenameNoMenu	Extensibility Guide.docx
ows-LinkFilename	Extensibility Guide.docx
ows-LinkFilename2	Extensibility Guide.docx
ows-ServerUrl	/Shared Documents/Releases/2.4.0/Documentation/Extensibility Guide.docx
ows-EncodedAbsUrl	https://repstor.sharepoint.com/Shared%20Documents/Releases/2.4.0/Documentation/Extensibility%20Guide.docx
ows-BaseName	Extensibility Guide
ows-MetaInfo	ttttttt:SW|
ows—Level	1
ows—IsCurrentVersion	1
ows-ItemChildCount	0
ows-FolderChildCount	0
ows-File-x0020-Size	2006321
ows-CheckedOutUserId	
ows-IsCheckedoutToLocal	0
ows-VirusStatus	2006321
ows-CheckedOutTitle	

ows—CheckinComment	Checked in by Repstor
ows-ParentVersionString	
ows-ParentLeafName	
ows-ParentUniqueId	{D1E8B739-6CD0-42CF-A40E-1ADF9F9297C4}
ows-StreamHash	
ows-DocIcon	docx
ows-FileSizeDisplay	2006321
ows-Edit	0
ows-Modified-x0020-By	i:0#.f|membership|fergus.wilson@repstor.com
ows-Created-x0020-By	i:0#.f|membership|fergus.wilson@repstor.com
ows-File-x0020-Type	docx
ows-SelectFilename	5840
ows-Combine	0
ows-RepairDocument	0

Example Folder Properties (SharePoint)

ows-ContentTypeId	0x0120002373F093BED2364B8772683A62F8D88E
ows-ID	5835
ows-ContentType	Folder
ows-Modified	2014-06-13T18:10:57Z
ows-Created	2014-06-13T18:10:57Z
ows-Author	Fergus Wilson
ows-Editor	Fergus Wilson
ows-owshiddenversion	1
ows-WorkflowVersion	1
ows—UIVersion	512

ows—UIVersionString	1.0
ows—ModerationStatus	0
ows-SelectTitle	5835
ows-Order	583500.000000000
ows-GUID	{37D5D67D-9C34-4ECD-9959-D0DD87708983}
ows-FileRef	Shared Documents/Releases/2.4.0/Documentation
ows-FileDirRef	Shared Documents/Releases/2.4.0
ows-Last-x0020-Modified	2014-07-31T09:31:55Z
ows-Created-x0020-Date	2014-06-13T18:10:57Z
ows-FSObjType	1
ows-SortBehavior	1
ows-PermMask	0x7fffffffffffffff
ows-FileLeafRef	Documentation
ows-UniqueId	{D1E8B739-6CD0-42CF-A40E-1ADF9F9297C4}
ows-ProgId	
ows-ScopeId	{7D308A85-3398-4E1F-B8E8-F13AEA56EEAA}
ows--EditMenuTableStart	Documentation
ows--EditMenuTableStart2	5835
ows—EditMenuTableEnd	5835
ows-LinkFilenameNoMenu	Documentation
ows-LinkFilename	Documentation
ows-LinkFilename2	Documentation
ows-ServerUrl	/Shared Documents/Releases/2.4.0/Documentation
ows-EncodedAbsUrl	https://repstor.sharepoint.com/Shared%20Documents/Releases/2.4.0/Documentation
ows-BaseName	Documentation
ows-MetaInfo	

ows—Level	1
ows—IsCurrentVersion	1
ows-ItemChildCount	5
ows-FolderChildCount	0
ows-File-x0020-Size	
ows-CheckedOutUserId	
ows-IsCheckedoutToLocal	0
ows-VirusStatus	
ows-CheckedOutTitle	
ows—CheckinComment	
ows-ParentVersionString	
ows-ParentLeafName	
ows-ParentUniqueId	{EF73F962-1DFC-4C98-AE5A-21E64BAF4E99}
ows-StreamHash	
ows-Edit	0
ows-SelectFilename	5835
Replify.ContentTypeId	0x0120002373F093BED2364B8772683A62F8D88E
ows-Combine	1
ows-RepairDocument	1

Appendix B - PropertyMap interface

The following is the definition of the ILSPropertyMap interface. Not the HasKey function is only available for string fields.

```
Interface ILSPropertyMap : Idispatch{
	[propget, id(1), helpstring("property ItemStr"), defaultcollelem] HRESULT ItemStr([in] BSTR key, [out, retval] BSTR* pVal);
	[propput, id(1), helpstring("property ItemStr"), defaultcollelem] HRESULT ItemStr([in] BSTR key, [in] BSTR newVal);
	[propget, id(2), helpstring("property ItemNum")] HRESULT ItemNum([in] BSTR key, [out, retval] ULONG* pVal);
	[propput, id(2), helpstring("property ItemNum")] HRESULT ItemNum([in] BSTR
```

```
key, [in] ULONG newVal);
	[propget, id(3), helpstring("property ItemDateTime")] HRESULT
ItemDateTime([in] BSTR key, [out, retval] DATE* pVal);
	[propput, id(3), helpstring("property ItemDateTime")] HRESULT
ItemDateTime([in] BSTR key, [in] DATE newVal);
	[propget, id(4), helpstring("property PropertyCount")] HRESULT
PropertyCount([out, retval] ULONG* pVal);
	[propget, id(5), helpstring("property HasKey")] HRESULT HasKey([in] BSTR
key, [out, retval] VARIANT_BOOL* pVal);
	[propget, id(6), helpstring("property KeyAt")] HRESULT KeyAt([in] ULONG
index, [out, retval] BSTR* pVal);
	[propget, id(7), helpstring("property ItemBool")] HRESULT ItemBool([in] BSTR
key, [out, retval] boolean* pVal);
	[propput, id(7), helpstring("property ItemBool")] HRESULT ItemBool([in] BSTR
key, [in] boolean newVal);
	[propget, id(8), helpstring("property ItemDouble")] HRESULT ItemDouble([in]
BSTR key, [out, retval] DOUBLE* pVal);
	[propput, id(8), helpstring("property ItemDouble")] HRESULT ItemDouble([in]
BSTR key, [in] DOUBLE newVal);
	[propget, id(9), helpstring("property ItemCurrency")] HRESULT
ItemCurrency([in] BSTR key, [out, retval] VARIANT* pVal);
	[propput, id(9), helpstring("property ItemCurrency")] HRESULT
ItemCurrency([in] BSTR key, [in] VARIANT newVal);
	[id(10), helpstring("set multiple ItemStr")] HRESULT ItemStrs([in] VARIANT
vSafeArray);
	;
```

Appendix C – Extension Support Object

The following is the definition of the Extension Support Object interface. This is available within extensibility functionality by calling into the “repsup” object.

```
interface ILSExtensionSupport : IDispatch
{
	HRESULT LogDiag([in] BSTR msg);
	HRESULT LogError([in] BSTR msg);
	HRESULT MessagePopup([in] BSTR title, [in] BSTR msg);
	HRESULT BalloonPopup([in] BSTR title, [in] BSTR msg);
	HRESULT AddReposWithUI([in]BSTR filterName,
		[in]BSTR defaultName,
		[in]BSTR location,
		[in, optional]VARIANT parentGroup,
		[in, optional]VARIANT openInDrive);
	HRESULT AddReposNoUI([in]BSTR filterName,
		[in]BSTR defaultName,
		[in]BSTR location,
		[in, optional]VARIANT parentGroup,
		[in, optional]VARIANT headerDownloadType,
		[in, optional]VARIANT openInDrive,
		[out,retval] BSTR *pAnyError);
	HRESULT RefreshRequiredRepositories();
	HRESULT SyncFolderPath([in]BSTR FolderPath);
```

```
	HRESULT HasLicenseFor([in]int capability, [out,retval] VARIANT_BOOL
*pIsLicensed);

	HRESULT SetUserStringConfig([in] BSTR setting, [in] BSTR value);
	HRESULT GetUserStringConfig([in] BSTR setting, [out,retval] BSTR *pValue);
	HRESULT SetUserDWORDConfig([in] BSTR setting, [in] LONG value);
	HRESULT GetUserDWORDConfig([in] BSTR setting, [out,retval] LONG *pValue);
	HRESULT SetUserProfileStringConfig([in] BSTR setting, [in] BSTR value);
	HRESULT GetUserProfileStringConfig([in] BSTR setting, [out,retval] BSTR
*pValue);

	HRESULT ShowBrowserDialog([in] BSTR url, [out, retval] BSTR *pDialogResult);
	HRESULT ShowCustomDialog([in] BSTR title, [in] BSTR msg, [in]BSTR buttonsCSV,
[in, optional]VARIANT defaultButtonName, [in,optional]VARIANT showCancel, [out,
retval] int *pDialogResult);
	HRESULT ShowPromptDialog([in] BSTR title, [in] BSTR bstrPromptCaption, [in,
optional]VARIANT varPromptInitialText, [out, retval]BSTR * pbstrPromptFinalText);
	HRESULT ShowPromptDialogWithCancel([in] BSTR title, [in] BSTR
bstrPromptCaption, [in, optional]VARIANT varPromptInitialText, [out, optional] VARIANT
*pbIsCancelled, [out, retval]BSTR * pbstrPromptFinalText);
	HRESULT ShowOptionsDialogWithCancel([in] BSTR title, [in] BSTR
bstrPromptCaption, [in] BSTR bstrOptionsPSV,[in, optional]VARIANT
varPromptInitialText, [out, optional] VARIANT *pbIsCancelled, [out, retval]BSTR *
pbstrPromptFinalText);
	HRESULT ShowConfirmDialogWithCancel([in] BSTR title, [in] BSTR
bstrPromptCaption, [out, retval] VARIANT *pbIsCancelled);
	HRESULT BrowseFoldersDialog([in] BSTR defaultFolder, [out, retval] BSTR
*pSelectedPath);
	HRESULT RepstorBuildNumber( [out, retval] BSTR *pVersionNumber);
	HRESULT GetUserLang( [out, retval] BSTR *pLocale);

	HRESULT ImportRepositoryList( [in] BSTR filePath);
	HRESULT ExportRepositoryList( [in] BSTR filePath);

	HRESULT PendingItemsCount( [out,retval] int *pItemsCount);
	HRESULT ErrorItemsCount( [out,retval] int *pItemsCount);
	HRESULT LocalEditItemsCount( [out,retval] int *pItemsCount);

	HRESULT RemoveRepository([in] BSTR reposNormalizedUrl);

	HRESULT GetRepositoryList([out,retval] BSTR *pReposList);

	HRESULT IsExistingRepository([in] BSTR reposNormalizedUrl, [out,retval]
VARIANT_BOOL *pIsRepository);

	HRESULT GetDigestValue([in] BSTR spWebUrl, [out,retval] BSTR *pDigestValue);

	HRESULT UrlEncode([in] BSTR spUrl, [out,retval] BSTR *pEncodedUrl);

	HRESULT UrlDecode([in] BSTR spUrl, [out,retval] BSTR *pDecodedUrl);

	HRESULT FindExistingRepositoryPath([in] BSTR reposNormalizedUrl, [out,retval]
BSTR *outlookFolderPath);

	HRESULT FindFolderPathProps([in] BSTR outlookFolderPath, [out,retval] IDispatch
**ppFolderProps);

	HRESULT  CreateAndLaunchPrecedentInFolderPath([in] BSTR outlookFolderPath, [in]
BSTR precedentName);

	HRESULT QueuePendingItemFolders();
```

```
        HRESULT TouchPendingItems();

        [propput, bindable, requestedit, id(38)]
        HRESULT Application([in]IDispatch * lpApp);

        HRESULT FindFolderByProp([in] BSTR folderPropName,[in] BSTR folderPropValue,
[in] BSTR folderExtraTokens, [in,optional] VARIANT onlyWritable,[out,retval] BSTR
*pOutlookFolderPath);

        HRESULT JumpToFolder([in] BSTR folderPath);

        HRESULT OpenFolderItem([in] BSTR folderPath, [in] BSTR propName, [in] BSTR
propValue, [out,retval] VARIANT_BOOL *pSucceeded);

        HRESULT GetFolderRepositoryMap([in] VARIANT spFolderPropMap, [out, retval]
VARIANT *pReposFolder);

        HRESULT GetParentFolderMapForMessage([in] VARIANT spMessagePropMap, [out,
retval] VARIANT * pParentFolder);

        HRESULT GetParentFolderMapForFolder([in] VARIANT spFolderPropMap, [out, retval]
VARIANT * pParentFolder);

        HRESULT CopyItemToFolder([in] VARIANT spMessagePropMap, [in] BSTR
bstrOutlookPath);

        HRESULT SavePropertyMap([in] VARIANT spMessagePropMap);

        HRESULT GetFilterTextForMsg([in] VARIANT spMessage, [out,retval] BSTR
*pbstrFilterText);
         HRESULT LoadCustomRuleSet([in] BSTR bstrRuleSetName, [in] int expiryDays);
        HRESULT CustomRuleSetAddText([in] BSTR bstrRuleSetName, [in] BSTR bstrTagName,
[in] BSTR bstrText);
        HRESULT CustomRuleSetGetTag([in] BSTR bstrRuleSetName, [in] BSTR bstrText,
[out, retval] BSTR *pbstrCat);

        HRESULT SetFieldByDisplayValue([in] VARIANT spMessagePropMap, [in] BSTR
bstrFieldName, [in] BSTR bstrValue);
        HRESULT GetFieldDisplayValue([in] VARIANT spMessagePropMap, [in] BSTR
bstrFieldName, [out, retval] BSTR *pbstrValue);

        HRESULT GetFolderPath([in] VARIANT spFolderPropMap, [out, retval] BSTR *
pbstrFolderPath);

        HRESULT GetMainwindowHandle([out, retval] LONG *pValue);
};
 }
```