Magic xpa 2.3a Release Notes

OUTPERFORM THE FUTURE™

New Features, Feature Enhancements and Behavior Changes

Android™ and iOS™ – Spinner Display

The Android and iOS clients now display a spinner when the application is busy.

Android™ – Cache Folder

The Android client supports the 'temp' keyword in the ClientOSEnvGet() function to retrieve the cache folder (as in iOS).

Controls' SWFName Property

The value of the SWFName property of the controls can now be set with the value of Magic xpa's Control Name property if the Control Name property has a value and when the SpecialSwfControlNameProperty special setting is set to Y.

Fixed Issues

- 176703 – The Runtime engine crashed when calling a resident task twice and each time a different form was displayed.
- 276562 – The form was refreshed slowly when it had a Tab control with lots of controls that had a visible expression.
- 281569 – There was a memory corruption when a BLOB variable was updated twice within a single transaction, once with short data and a second time with longer data.
- 284486 – The wrong context menu was evaluated in runtime when right clicking (without clicking it first) on a control that had a context menu.
- 289735 – The project name was not displayed on the title bar when the AllowTesting setting in the Magic.ini file was set to Yes.
- 296741 – The focus did not move to a Push Button control outside the table when clicking on any record in the table.
- 304575 – The CLeft() and CTop() functions were not converted to CX() and CY() when converting a version 9.4 application directly to Magic xpa using the V9 converter and then migrating the application.
- 305024 – Duplicate nodes were displayed in a Rich Client Tree control if any parameters were passed to a subfom.
- 307581 – It was not possible to call Magic xpa using the .NET requester when the broker was defined with a port number.
- 309285 – Incremental locate on a Hebrew variable did not work properly when the typing was fast.
- 310719 – Magic crashed when the result of a Case function used in the Invoke UDP operation's Arguments was the Default value.
- 439895 – The MnuShow() function showed menus that were defined with the Visible property unchecked.

Magic xpa 2.3 – New Features, Feature Enhancements and Behavior Changes

Table Control – Multi-Mark Support in Online Tasks

Multi-marking is now supported for Table controls in Online tasks.

A row is marked by clicking on it. The **Marking Column** property is, therefore, obsolete and no longer supported.

Window Pulldown Menu Support for Online Applications

The Window pulldown menu is now supported and behaves similar to the V1.9x menu.

This menu will show all the windows defined with **Show in Window Menu = Yes**. The **More Windows** entry is, therefore, obsolete and no longer supported.

Label Control – Expression Support in Online Tasks

Expressions are now supported for the Label control in both Online and Rich Client tasks.

.NET Data View Control Binding in Online Tasks

It is now easier than ever to use .NET grid controls in Magic xpa.

A new property named **Dataview Control** was added to both the .NET control and the .NET control model so that you can define a .NET control as the dataview control. This definition means that the entire task data view will be assigned to the **DataSource** property of the .NET control.

You can now use any .NET control grid instead of the Magic xpa built-in Table control. Any modification made to the Magic xpa variables will be automatically reflected in the .NET control grid and vice versa.

Note that at this point this functionality is supported in Online tasks only and when the **Preload View** = **Yes**.

For more information on how to use .NET dataview control binding, see the .NET Tutorial sample installed with Magic xpa.

.NET Choice Controls' Data

It is now easier than ever to use .NET choice controls in Magic xpa.

A few new properties were added to the .NET control model so that you can define the .NET control properties for the data source, display member, and value member values.

After these properties are defined, the .NET control placed on the Form Editor will have the same properties as the built-in choice controls, so you can use the .NET control in the same way as you use any Magic xpa built-in choice control.

For more information on how to use .NET data binding, see the .NET Tutorial sample installed with Magic xpa.

.NET Data Binding – Behavior Change

When using a .NET control defined with data binding and updating the .NET control, Magic xpa automatically updates the Magic xpa variable with the .NET control value.

If after the update, the values in the Magic xpa variable and the .NET control are different, the control will be updated with the Magic xpa variable value (instead of keeping the typed value as in previous versions).

.NET in Batch Tasks Called from Rich Client Tasks

It is now possible to run .NET code in a Batch task called from a Rich Client task.

Note that:

- A client side .NET object (defined in a Rich Client task) will not be available on the server side (Batch task) and vice versa.
- It is not possible to send .NET variables as a parameter to a Batch task.

Generic Server Error Messages in Rich Client Applications – Behavior Change

At runtime, when there is an error serving a request (such as Server not found), a generic message containing only the error code will be shown (instead of showing a full message exposing the server address).

You can still have the previous functionality by specifying the 'N' value in the **DisplayGenericError** property in the execution properties.

List Box Control – Multi-Mark Support in Rich Client Tasks

Multi-marking is now supported for List Box controls in both Online and Rich Client tasks.

Edit Control – Vertical Alignment Support in Rich Client Tasks

Vertical Alignment is now supported for Edit controls in both Online and Rich Client tasks.

Choice Controls – Data Source Properties

The data source properties (such as **Item list** and **Display list**) of choice controls were separated into a new group named **Data Source**.

Form – Startup Position Property

The **Windows default bounds** and **Windows default location** values were renamed to **OS default bounds** and **OS default location** respectively.

Tab Control – Hebrew Version

The Hebrew version is now aligned with the English version and supports the following properties for the Tab control:

- Tabs width
- Hot Track
- Multiline
- Image list fie name
- Image list indexes

These properties affect runtime only. They do not affect the Form Editor of the Hebrew version.

Functions – Data View Variables' Indexes

A new function named DataViewVarsIndex() was added to retrieve the data view variables' indexes.

Functions – Names and Types of Global Parameters and Shared Values

The following functions were added to retrieve the names and the type of the global parameters and shared values:

- GetParamNames()
- GetParamAttr()
- SharedValGetNames()
- SharedValGetAttr()

Background Mode – Showing the Verify Message Box

A new special setting named **SpecialAppServerAllowVerifyInBox** was added to allow opening the message box of the Verify operation when the engine is running as an application server in Online deployment mode.

Application Installation Utility – Govern Web Service Installations

A new option was added to the application installation utility to define whether or not to install Systinet.

Enable Usage of 3-GB Address Space

The Magic xpa engine can now access a 3-GB address space.

For this to work you must modify the **Boot.ini** file to enable application memory tuning. To do this, add the **/3GB** parameter to the **ARC** path in the **Boot.ini** file. Refer to:

http://msdn.microsoft.com/en-us/library/windows/hardware/ff556232(v=vs.85).aspx.

Using Magic xpa from .NET Applications

It is now possible for .NET applications, such as ASP.NET or a .NET web service, to interact with Magic xpa.

For more information, please see the **Managed Requesters** Help page.

Broker Execution

When launching the broker, it will now be launched as one process (without a watchdog process as implemented in previous versions). If you need to load the watchdog process, you can start the broker with the **-StartAsWatchdog** parameter.

Note: This is only applicable if the broker is started as an executable (not as a service).

Rich Client Deployment Builders – Mobile Device Support

The Rich Client Deployment Builder now creates Android, iOS and BlackBerry packages.

For iOS, since the compilation is done on a Mac machine, the builder will generate the required files. Therefore, you should simply copy them to the Mac and run the build script. You may also need to copy the **iTunesArtwork** and **iTunesArtwork@2x** files to the Mac.

Note that the deployment files for each operating system will be created in a subfolder with the same name as the operating system. If you want to maintain the previous folder structure for Windows desktop deployment, use the builder metadata file from the previous version.

Rich Client Form Editor – Combo Box Height

The Combo Box control's Height property is now modifiable on Rich Client forms so it can be used to define the controls' height on Android and iOS clients.

On Windows platforms, the value of this property has no effect since the Windows combo box control ignores the Height property and calculates the height according to the font used.

Rich Client Form Editor – Z-order Limitation

The limitation of setting a manual z-order for some controls to be above other controls (such as to define an Edit control with a z-order value higher than a Button control z-order) was removed, so that it can be used to define the controls' z-order on Android and iOS clients.

On Windows platforms, combinations that are not allowed are still not supported, so they will be ignored.

Android™ and iOS™

The Android and iOS clients were enhanced and now support the following features:

- Edit control – behavior change
  - Alpha and Unicode – picture mask

    The picture mask of Alpha and Unicode Edit controls is now calculated when leaving the control (instead of after typing a character). This change provides support for the built-in auto-complete feature of the mobile devices.
  - Date and Time – pickers

    When the focus moves to an editable Date or Time Edit control, the native OS date or time picker will be opened.

    If you want to update the Edit control manually without a picker, you can use the **'picker=0'** value in the **Additional Information** property of the control.
- Resources per platform and device characteristics

  The Android and iOS RIA mobile client now support alternative resources to support specific device configurations.

  You can define multiple copies for the same resource and place them in specific folders on the server. The RIA client will automatically retrieve the matching resource according to the device platform, resolution and DPI.

  This can be used, for example, to automatically support:

  - Images for different resolutions.
  - Font files and color files for different platforms.

  For example, if the runtime font file is **Support\fnt_rnt.eng**, then you can put the Android font file at **Support\Android\fnt_rnt.eng** and the iOS font file at **Support\iOS\fnt_rnt.eng**.

  At runtime, the appropriate font file will be used automatically, if it exists. There is no need to define this file implicitly.

  This is also the case for images. For example, you can place an HDPI image at **c:\images\Android\hdpi** and an LDPI image **at c:\images\Android\ldpi**. At runtime, the appropriate image file will be retrieved automatically (if such a file exists).

  For more information about how to use alternative images, refer to the **Alternative images (Android and iOS)** section in the **RIA for Mobile Devices** concept paper.

- The controls' border can be fully customized for controls with a non-system color by using the following values in the **Additional Information** property:
    - 'BorderWidth=xxx' – the border width
    - 'BorderColor=xxx' – the border color from the color table
    - 'BorderFocusWidth=xxx' – the border width when the Edit control is in focus
    - 'BorderFocusColor=xxx' – the border color when the Edit control is in focus
- The corner radius of the Text, Edit, Button, Group and Image controls can be fully customized for controls with a non-system color by using the following values in the **Additional Information** property:.
    - 'CornerRadius=xxx' – defines the controls' corner radius

  Note: For iOS devices, the Subform and Table controls also support this functionality.
- The internal **Press** event is raised when performing a long-press gesture on a control or form on the mobile device.
- The application area size in inches can be retrieved using the **device_physical-width** and **device_physical-height** values in the **ClientOSEnvGet()** function. These values can be used, for example, to determine whether the device is a phone or a tablet.
- The **Row Highlight Style** property is now supported.
- Gradient Support – The form and the Group and Push Button controls support the following gradient styles: Horizontal, Vertical, Diagonal Left and Diagonal Right.

  Note: On Android devices, the Label control also supports the above gradient styles.
- Launching the application from another application

  You can launch your application from another application or from an email using a URI convention such as **myapp://?parameters** on iOS and **http://com.mycompany.myapp?parameters** on Android.

  The query parameters sent to the application can be retrieved using the ClientOSEnvGet function with the value of **'device_udf|getargs'**.

  For more information about how to use this feature**,** refer to the **Launching the application from another application** section in the **RIA for Mobile Devices** concept paper.

iOS Only

- Fixed orientation form, so the form will not be rotated upon screen rotate.

  This is done by defining the value of the **Additional Information** form property to **orientation=portrait** or **orientation=landscape** (as in Android).
- OS commands are now supported when defined as a menu entry (as in Android).
- Popup screens are supported by defining the **popup=Y** value in the **Additional Information** form property (as in Android).

On iOS devices, unlike Android, the window size and location can be defined using the navigation properties and the **Startup Position** property.

- Printing a PDF using the AirPrint protocol.

  Printing a PDF can be done via the Invoke OS command with the command of **Print:filename**, where **filename** is either:

  - A path to a local file (as received from the ServerFileToClient() function)
  - A URL to a file on a web server (such as 'http://1.1.1.1/myfile.pdf')

Android and BlackBerry Only

- Popup screen support – behavior change

  Popup screens are now defined using the **popup=Y** value in the **Additional Information** form property.

  The previous way (using a Floating window type) is no longer supported.

  If you used the Floating window type in previous versions, you need to manually update your application,

Prerequisites Update

The minimum .NET Framework version required to run Magic xpa is V2.0 SP2 (instead of SP1). When using this version, you also need to install the Microsoft Visual C++ 2008 Redistributable.

Magic xpa 2.2a – New Features, Feature Enhancements and Behavior Changes

As Parent Initial Mode in Rich Client Tasks

The behavior of the **As Parent** initial mode in Rich Client tasks was changed and it will now be recomputed when there is a change in the parent task mode (instead of being set only once when the subform task is opened).

Database Connection Pooling

Magic xpa uses a database pooling mechanism that allows for sharing of connections between different contexts.

This behavior improves the efficiency and the performance of the server.

However, there are scenarios in which you need a separate connection per RIA client, so when the context is closed, so does the connection.

This can now be done by setting the new **SpecialDatabaseConnectionPooling** flag to **N**.

Android™ and iOS™

The Android and iOS clients were enhanced and now support the following features:

- Subform control
- Client side images
- Spinner display – This is done using the SetCrsr() function.
- Image picker from the gallery – This is done by sending the value of 'images' in the ClientFileOpenDlg() function's second argument.
- Call to a native OS code – This is done by evaluating the ClientOSEnvGet() function with the value of **'device_udf|my_string'**.

  Performing this action will send the value **'my_string'** to the function named **userDefinedFunction** in the mobile application code.
- Call from the native OS code to the application – This is done by running the **invokeExternalEvent("my_string")** command in the native OS code.

  Performing this action will raise the internal event named 'External Event' in the Magic xpa application and send the value **'my_string'** to it.

iOS Only

- Simpler customization methodology – The iOS application customization settings (such as defining the package name and the provision file) were changed and are now defined in a single file named **settings.properties** as in Android.

Android Only

- Fixed orientation form, so the form will not be rotated upon screen rotate – This is done by defining the value of the **Additional Information** form property to **orientation=portrait** or **orientation=landscape**.
- Define scrolling on the form or its controls – The Android interface allows only one scroll object inside a hierarchy. This means that, for example, if the form has a scrollable object, such as a Table, Browser or Subform control, then the scroll can be done in either the form or the subobject, but not in both.

  You can set the scrolling for the form or its controls by defining the value of the **Additional Information** property of the form or its controls to **scrollhorizontal**=X or **scrollvertical**=X where X can be Y or N.

Magic xpa 2.2 – New Features, Feature Enhancements and Behavior Changes

Dependency in Mshtml.dll in Rich Client Tasks

The Mshtml.dll file is no longer required to run Rich Client tasks.

If your RIA application needs to use the BrowserScriptExecute() function, then it should include a reference in the CRR to this dll file.

Additional Functions Supported in Rich Client Tasks

The SubformExecMode, MnuRemove and MnuReset functions are now also supported in Rich Client tasks.

Status Bar Panes in Rich Client Tasks

The task mode, wide and zoom indications can be seen in the status bar of a RIA application using a new special setting named **SpecialShowStatusBarPanes**.

Spanish Language Support

Magic xpa now supports a Spanish Language Studio.

The new language was added to the Language list in the installation wizard.

Android™ Support

The Android client was enhanced with the following features:

- Simpler customization methodology
- Repackaging script
- Support device location (GPS)
- Accessing the mobile devices' capabilities (telephone, text messages, emails)
- Support of the Row Highlight Color property for the Table control

iOS™ Support

The iOS client was enhanced with the following features:

- Support of the Row Highlight Color property for the Table control
- Support of the Subform control (Beta feature)

Refer to the **RIA for Mobile Devices PDF** or to the **RIA for Mobile Devices** concept paper available in the Magic xpa Help.

BlackBerry™ Support

The BlackBerry client is now available as part of the installation.

In addition, the BlackBerry client was enhanced with the following features:

- Support of the Line and Group controls
- Support of the Wallpaper form property
- Support encryption of messages between the client and the server (The SpecialClientSecureMessages=N is no longer required.)

Change of behavior:

- Forms with a Floating window type will now open as a popup window. The Title Bar property, which previously defined if the window opened as a popup window, is now supported for full and popup windows and defines whether a title bar will be shown.
- The forms will be automatically scaled according to the device's DPI.

Refer to the **RIA for Mobile Devices PDF** or to the **RIA for Mobile Devices** concept paper available in the Magic xpa Help.

BlackBerry PlayBook™ Support

Magic xpa RIA client can run on the BlackBerry PlayBook tablet using the PlayBook's **BlackBerry Runtime for Android apps**' capabilities.

The script for repackaging an Android 2.3.3 RIA application to BAR file format, which is the compatible file format required for an application to run on the BlackBerry Tablet OS, is available as part of the installation.

Network Installation

The .NET runtime security policy (by default) disables code from running if it exists on a network drive. To run Magic xpa from a network drive you need to adjust your security policy.

This can be done via the Microsoft .NET Configuration tool or by running the following command:

```
c:\Windows\Microsoft.NET\Framework\v2.0.50727\caspol.exe -machine
-addgroup All_Code -strong -file <path>MgxpaRuntime.exe -noname -
noversion FullTrust -name Magicxpa_Assemblies_Access -description
"Code group granting trust to Magic xpa assemblies"
```

This command needs to be executed only once in each of the computers.

You can also add a **–silent** key for a silent installation.

Mobile Application Samples

A sample project named **Mobile Demo** was added to demonstrate how you can write mobile applications using the mobile device capabilities of Magic xpa's RIA technology.

Magic’s Company Rebranding: A New Look for a New Age of Magic

Magic is launching its rebranding as a proactive strategic move to reflect the exciting changes that we are undergoing as a company and the major developments in our product offering.

Magic’s rebranding highlights our company values, the set of beliefs that guide and inspire us at every level throughout our organization.

For our brand architecture—the framework that defines the relationship between the corporate brand and the product portfolio—we have chosen to use the 'master brand' model, focusing more on our company brand (Magic) rather than on the branding assigned to our different products and services. This emphasizes that Magic’s different products are all instances of the same unified technology stack and the same fresh approach to enterprise software.

Accordingly, we have renamed our products as follows:

- **uniPaaS** is now named **Magic xpa Application Platform**
- **iBOLT** is now named **Magic xpi Integration Platform**

Magic's rebranding aims to illustrate that all our corporate activities and communications are infused with our core values, differentiating us from competitors, strengthening our relationships with customers, and increasing the popularity of our products and services.

Renaming of Product Executables

As part of the rebranding process, some of the product files were renamed, including:

- **uniStudio.exe**, which is now called **MgxpaStudio.exe**
- **uniRTE.exe**, which is now called **MgxpaRuntime.exe**
- **uniRQBroker.exe**, which is now called **MgBroker.exe**
- **uniRQMonitor.exe**, which is now called **MgBrokerMonitor.exe**

uniPaaS 2.1 – New Features, Feature Enhancements and Behavior Changes

Android and iOS Support

The uniPaaS RIA client is now capable of running on Android™ and iOS™ devices.

As for the Windows Mobile and BlackBerry, the Magic RIA client for Android and iOS is a native operating system (OS) application implementing the Magic RIA client protocol. Using the Magic RIA client for the different mobile devices, developers can deploy enterprise connected, highly interactive RIA applications on the different mobile devices.

Note that the RIA client for Android is a Beta version.

Refer to the **RIA for Mobile Devices PDF** or to the **RIA for Mobile Devices** concept paper available in the Magic xpa Help.

Call to a Destination Subform or Frame in Online Tasks

The Call operation in Online programs was enhanced with a new property called 'Destination'. This property provides the ability to dynamically call a program or a task and run it in a subform or a frame (similar to Rich Client).

Retain Focus in Online Tasks

The Call operation in Online programs was enhanced with a new property called 'Retain Focus'. This property defines whether the focus will remain on the current control or be moved to the first control of the called program or task after executing the Call operation.

This property is enabled when performing the call from a logic unit other than the Task, Record, or Control logic units to a destination subform or frame using the Destination property.

Parallel Execution in Online Tasks

Parallel execution is now supported for Online tasks.

The Application Modal window type was removed and the Modal window type will behave as the old Application Modal.

The Window menu and window-related events are still not supported.

View Refresh in Rich Client Tasks

The View Refresh behavior in Rich Client tasks was enhanced. After performing a view refresh with 'Relocate Mode=0', the position of the current record now remains as-is (and is not changed to be the first record).

Row Placement Property in a Table Control

A new property was added to support the fixed number of table records.

Setting this property value to Yes means that the row's height will be resized according to the table's height, thus keeping the same number of records as designed in the Studio.

Default Location of Forms

A new option named Windows Default Location was added to the form's Startup Position property. When using this option, the form is positioned at the Windows default location and has the dimensions specified in the form's size.

The Default option in which the form is positioned at the Windows default location and has the bounds determined by the Windows default was renamed to Windows Default Bounds.

.NET Data Binding

It is now easier than ever to use uniPaaS data variables with .NET controls.

Two new properties were added to the .NET control model so that you can define the .NET control property to which you want to bind the data and the event that will be the trigger for the data update.

After these properties are defined, you can use the .NET control in the same way as you use any uniPaaS built-in control. You simply need to attach your data variable to the .NET control's Data property.

The previous functionality of assigning a .NET variable to the .NET control is still supported and can be done by using the .NET Object property in the .NET control.

For more information on how to use .NET data binding, see the .NET Tutorial sample installed with uniPaaS.

.NET – DataViewToDNDataTable() Function

A new function was added to create a .NET DataTable object out of the task's data view.

The .NET DataTable objects can then be used as the data source of other .NET objects.

.NET Controls – Change of Behavior

When clicking on a .NET control defined with Allow Parking = No, the focus will not leave the current control or record.

Requester and Broker Logs – Default File Names

The default value for the requester and broker logs were changed as follows:

mrb_event.log => BrokerActivity.log, mrb.log => Broker.log, req.log => Requester.log

Mobile Devices – Additional Information

A new property was added to allow maximum flexibility with mobile device properties.

This property will be used to send specific pre-defined information to the mobile devices.

Mobile Devices – Press Event

A new event was added to handle the long press on a control in mobile devices. At this stage, this event is supported for BlackBerry only.

.NET 3rd Party Samples

A new sample project was added to demonstrate how you can use 3rd party .NET objects and controls in your application to improve the functionality and user interface.

The 3rd party assemblies are not included with this package and should be downloaded from the 3rd party vendor site as described in the sample programs.

Note: If your 3rd party package version is different than the one used in the samples, you will have to replace the assemblies defined in the CRR with the ones from your version.

uniPaaS 2.0a – New Features, Feature Enhancements and Behavior Changes

MDI Frame Color

A new color was added to the default color file with the value corresponding to an MDI Frame background.

This color is used as the MDI Form color in newly created projects (and can be changed in the proper ties of the Main Program form).

End-User Functionality Component

The End-User Functionality component is now added by default to newly created projects.

A new option was added in the migration wizard, which lets you select whether to also add this component to the migrated project.

Note that if you want to use both this component and the Report Generator, then the Report Generator component should be defined above the End-User Functionality component.

Subform Behavior

A new property named Refresh When Hidden was added to the Subform control in Online tasks.

This property provides the ability to load and refresh the task running in the Subform control the same way it was done in uniPaaS 1.9.

A value of Yes means that the subform tasks will be called for the first time after the Record Prefix of the host and again after each refresh of the subform. (This is similar to the behavior in uniPaaS 1.9 when the subform was attached to a container control and the container control was hidden.)

A value of No means that the subform tasks will be called only if the subform is visible. (This is similar to the behavior in uniPaaS V1.9 when the subform had a Visible expression.)

For nested subforms defined with a value of No, if the parent subform has a value of Yes at runtime, then the nested subform wi ll also be executed with a value of Yes.

The migration from uniPaaS 1.x will set the value of this property as follows:

- No – When the subform is attached to a container control.
- Yes – When the subform has a Visible expression.
- Exp – When the subform is attached to a container control and also has a Visible expression. The expression is the expression of the Visible property.

In addition, up until this version, when clicking on a Subform control, the Control Prefix and Control Suffix of the first control were always executed. This is now fixed, so only the Control Prefix of the clicked control will be executed.

EDP File Structure

The EDP file was changed a bit to prevent opening a uniPaaS 2.0 project in previous versions, in order to maintain the integrity of the sources.

uniPaaS 2.0 – New Features, Feature Enhancements and Behavior Changes

Prerequisites

uniPaaS is a native .NET application, so in order to run uniPaaS on your machine, you must have .NET framework installed on your machine with one of the following configurations:

- Both .NET Framework V2.0 SP1 and Microsoft Visual C++ 2008 Redistributable.

  You can install them by running the NetFx20SP1_x86.exe and VCRedist_x86.exe files from the Scripts\RIA folder.
- .NET Framework V3.5 (or above)

Network Installation

The .NET runtime security policy (by default) disables code from running if it exists on a network drive. To run uniPaaS from a network drive you need to adjust your security policy.

This can be done via the Microsoft .NET Configuration tool or by running the following command:

c:\Windows\Microsoft.NET\Framework\v2.0.50727\caspol.exe -machine -addgroup All_Code -strong -file <path>uniRTE.exe -noname -noversion FullTrust -name uniPaaS_Assemblies_Access –description "Code group granting trust to uniPaaS assemblies"

This command needs to be executed only once in each of the computers.

You can also add a –silent key for a silent installation.

Online Changes

In uniPaaS 2.0, the Online GUI is based on the .NET framework instead of Win32.

The .NET framework does not support all the appearances and control behaviors that were supported in previous versions, so there is a change in the uniPaaS Runtime appearance as well.

For a full list of changes, refer to the What's Different in uniPaaS 2.0 help topic.

Some of the major enhancements are described below.

.NET Integration in Online and Batch Tasks

.NET Integration was introduced to RIA applications in uniPaaS V1.8.

With this release you can enhance your Client/Server application offering by easily embedding and integrating any .NET control or assembly in your Online and Batch tasks.

For more information on how to add and manipulate .NET modules, see the **.NET Tutorial** sample installed with uniPaaS.

Online MDI

The Online Runtime MDI Frame is now defined as any other form in the Main Program (similar to the Rich Client MDI Frame).

The benefit of this change is the centralizing of all the form properties from the different locations they were once defined in, into a single location.

The MDI can be disabled by defining Open Task Window = No in the Main Program properties.

Online Frameset

Splitter forms are now defined using a frameset form for an Online task (as in Rich Client tasks) instead of using a Splitter form.

User State Persistency in Online Tasks

The Online task's form state persistency is now defined as a simple Yes/No property.

This change makes it simpler to use the form state persistency since you now do not need to define and maintain a list of unique identifier names for your forms.

Rich Client – Hebrew Support

The Logical() and Visual() functions are supported.

Rich Client – Forms

All UOM types (dialog units, centimeters, inches) are supported.

Rich Client – Drag and Drop

1. Drag & drop is supported for all of the controls.
2. User-defined format is supported.
3. DragSetCrsr() function is supported.

Rich Client – Post Refresh by Parent Event

A new internal event was added to execute logic every time the subform is refreshed.

This event is raised before the Record Prefix in the subform task when the subform is refreshed by the parent. The event is raised when one of the following occurs:

1. The subform is defined with Auto Refresh=Y and a variable sent to the subform as a parameter is changed.
2. The Subform Refresh event is raised.
3. The subform is executed for the first time.

The event is not raised when a 'View Refresh' is raised in the subform task.

Sample Projects

Additional sample programs were added to the Online Samples and Rich Internet Samples projects. The new sample programs are for some of the uniPaaS functions and for connectivity to Google Calendar, Google Blogger, Facebook, LinkedIn and Twitter.

uniPaaS 1.x Release Notes

For information about uniPaaS 1.x releases, see the PastReleaseNotes.pdf file.