Upgrading from Viewer 5.1 to the Viewer 6.0 Suite

This document lists the products in the ICS Suite for Viewer 6.0, provides information about upgrading to the latest version of each product , and details all of the changes in those products. PureEdge recommends that you upgrade your entire suite to the latest version of each product. This minimizes installation problems, and ensures that you have access to the latest features.

Upgrading to the latest version of a product may require a significant amount of work. In these cases, you should take some time to assess whether the upgrade is necessary. A number of "Frequently Asked Questions" are answered on page 6 to assist you in determining whether you need to upgrade.

Recommended Setup for Viewer 6.0

For best results when using Viewer 6.0, PureEdge recommends that you upgrade all of your ICS products to the appropriate versions, as shown:

ICS Product	Version for Viewer 6.0	Previous Version (Viewer 5.1/5.2)
XFDL	6.0	5.1 (or 5.2.0)
ICS API	6.0.0	5.1.0 (or 5.2.0)
ICS Designer	2.3.0	2.2.0 (or 2.2.1)

Note Version 5.2.0 (and version 2.2.1 of the Designer) was part of a limited release, and was not available to all customers. This document notes where the requirements for upgrading from version 5.1 differ from the requirements for upgrading from version 5.2.

ICS Viewer – Upgrading to Version 6.0

When upgrading to Viewer 6.0, you should consider the following Viewer issues:

- **Operating System** – If you are using Windows 95, you will not be able to run Viewer 6.0. You should either upgrade your operating system or continue using an earlier version of the Viewer.
- **User Preferences** – You should manually set all user preferences once you have upgraded the Viewer.
- **IFX Extensions** – If you are using any custom extensions with the Viewer, you will have to copy those extensions from the old version of the Viewer to the correct folder in the new version:

  Viewer 5.x extensions folder: <Installation Folder>\PureEdge\Viewer 5.x\Extensions\

  Viewer 6.0 extensions folder: <Installation Folder>\PureEdge\Viewer 6.0\Extensions\

  If any of the extensions were written using the version 4.4 API or earlier, you will have to recompile them using a newer API. See “Frequently Asked Questions” on page 6 for more information.

You should also examine your forms for the following considerations:

- **UFDL Forms** – If you are using UFDL forms, you must change them to XFDL. Ensure that you also change all .ufd or .ufdl extensions to .xfd. Since Designer 2.3 does not support UFDL either, you will have to use an earlier version of the Designer to convert UFDL forms.
- **.frm Extension** – If you are using the .frm extension for your forms, you should consider changing it to .xfd. The .frm extension still works, but support for it may be dropped in future versions.
- **MIME Type** – If your forms are using the *application/x-xfdl* MIME type, you should consider changing it to *application/vnd.xfdl*. The old MIME type still works, but support for it may be dropped in future versions (this is an XFDL change).
- **Version 4.3 or Earlier Forms** – If you are using forms written in XFDL 4.3 or earlier, you should upgrade them to version 4.4 or later and test them thoroughly. This will ensure that you do not lose functionality that relied on features that are no longer supported.

- **Submissions to Multiple URLs** – If your forms perform submissions to multiple URLs at once, you should test them thoroughly. You may have to adjust the submissions to ensure that they happen in the correct order, and that all submissions continue to work properly.

- ***label* Option** – If your forms use the *label* option, you should view each form to determine whether the size of the items has changed. Items that use the label option but do not have a border on the label will appear taller, which may affect the layout of your form.

- **Height in *extent*** – If your forms have growing fields (or similar items) that compute the height in the *extent*, you should test these forms to make sure sizing works as expected. The items may default to zero height if you have not included an initial value for the compute. In this case, simply add an initial value to the compute (you may wish to use the *extent's* width, since this used to be the default).

- ***toggle* Function** – If you have forms that rely on the *toggle* function, you should test them thoroughly. The implementation of the *toggle* function was rewritten to fix some problems, but some forms may rely on unusual behavior that has been eliminated.

- **Labels and Buttons with Empty Images** – If your forms contain label or buttons with an *image* option that points to a *data* item that is empty or does not exist, then you should review the form thoroughly. The label or button will now display its value, which will change its size and appearance on the form.

For a complete list of changes to the Viewer, see "Viewer 6.0 – Complete List of Changes" on page 8.

XFDL – Upgrading to Version 6.0

Due to the number of changes in XFDL 6.0, we recommend that you use the Designer to upgrade your forms. You can do this by loading an old form in the version 6.0 Designer, at which time the Designer will ask if you if you want to upgrade the form.

Once the Designer has upgraded your forms, you should consider the following:

- **Computes and Custom data** – Be sure to test that all computes and custom data are working properly. The Designer may have difficulty updating complex computes, especially if custom data is involved. Furthermore, the Designer moves all custom data to the *custom* namespace, which may not meet your needs.

- **Buttons and Actions** – Check to be sure that none of your buttons or actions were relying on *link* as the default type. If they were, you will have to explicitly set them to *link*.
- **Popups** – Check to see if your popups relied on the first cell being displayed by default. If they were, you may need to change the available cells. For example, you may have designed your forms so that the first cell in each popup read "Select a choice". Since popups now default to a blank entry, this cell would no longer be displayed.
- **File Extension** – If you are using the .frm extension, you should consider changing it to .xfd. While the .frm extension still works, it conflicts with Visual Basic and support for it may be dropped in a future version. If you change to the .xfd extension, remember to adjust any applications that may use the .frm extension.
- **MIME Type** – The Designer's upgrade process will change all form MIME types to *application/vnd.xfdl*. If you have applications that rely on the *application/x-xfdl* MIME type, you should either change the code or change your forms back to the *application/x-xfdl* MIME type.

Remember that the Designer's upgrade routine will not add new features to your form. To take advantage of new features, such as rich text fields, you must add those features to each form yourself.

For a complete list of changes to XFDL, see "XFDL 6.0 – Complete List of Changes" on page 11.

ICS Designer – Upgrading to Version 2.3

When upgrading the Designer, you should consider the following:

- **UFDL Forms** – The Designer no longer supports UFDL. If you are using UFDL forms, you should convert them to XFDL *before* you upgrade to the new version of the Designer.
- **IFX Extensions** – If you are using extensions that were written using the version 4.4 API or earlier, you will have to recompile them using a newer API. See "Frequently asked Questions" on page 6 for more information.

Remember that if you want to take advantage of the new XFDL features, you will have to redesign your forms to use those features.

For a complete list of changes to the Designer, see "Designer 2.3 – Complete List of Changes" on page 16.

ICS API – Upgrading to Version 6.0

When upgrading your applications to use the new API, you should consider the following:

- **UFDL Forms** – If you are updating applications that use UFDL forms, you must convert those forms to XFDL and modify your application accordingly.
- **IFX Extensions** – API 6.0 is binary compatible with API 4.5 and later. If you are using IFX Extensions that were compiled using the version 4.4 API or earlier, those extensions will not work with any products in the 6.0 suite. To use those extensions, you will have to recompile them using a newer API. See "Frequently Asked Questions" on page 6 for more information.
- **Version 4.4 or Earlier Forms** – The API does not version of XFDL prior to 4.5. This means that older forms may not function properly with your application. You should test all forms carefully, and upgrade them if necessary.
- **Data Types** – When updating an application to use the new API, you must convert all *charP* data types to *r_charP* and all *voidP* data types to *r_voidP*.
- **Java Naming Conventions** – To use any of the new functions in Java applications, you will have to use the new package (com.PureEdge) and the new naming conventions in that package.
- **Extended Functions** – You should switch to using the extended functions in all cases. The extended functions will process more quickly and support for the older functions may be dropped in future versions. If you are using extended functions in C, you will also need to update your memory handling to free returned string values.
- **MIME Type** – If you upgrade your forms to version 6.0, all *application/x-xfdl* MIME types will be changed to *application/vnd.xfdl*. If you have any applications that rely on the old MIME type, you should update those applications to use the new MIME type.
- ***toggle* Function** – If you have forms that rely on the *toggle* function, you should test them thoroughly. The implementation of the *toggle* function was rewritten to fix some problems, but some forms may rely on unusual behavior that has been eliminated.

For a complete list of changes to the API, see "API 6.0 – Complete List of Changes" on page 18.

Frequently Asked Questions

Are we entitled to receive upgrades?

Yes. Provided that your organization has a valid maintenance contract with PureEdge Solutions, you are entitled to receive upgrades to the ICS Suite.

What licensing requirements do we need to be aware of?

You need new license keys from PureEdge for every ICS product that you upgrade. New license keys are *not* required for maintenance releases, such as from version 4.4.1 to 4.4.2.

If we upgrade our clients to ICS Viewer 6.0, can they still use existing forms?

In most cases, yes. ICS Viewer is backwards compatible version 4.4 forms and later. If you are using forms from an earlier version, you will have to upgrade those forms.

If we upgrade to ICS Viewer 6.0, will the Viewer retain the preferences we set for the previous version?

No. The Viewer no longer retains preference settings when upgrading.

Our client computers are presently running a custom IFX (InternetForm extensions) created an earlier version of the API. If we upgrade clients to ICS Viewer 6.0, can we continue to use this IFX?

The Viewer will continue to work with extensions that were compiled with the version 4.5 API or later. If you are using extensions compiled with an earlier version of the API, you will have to update those extensions.

To continue using an extension created with an earlier version of the API, you simply need to copy the IFX to the correct folder in the new Viewer installation:

1. Install ICS Viewer 6.0 on the client computer.
2. Move the desired .ifx extension file from the old Viewer folder to the new Viewer folder.

   Old: <Installation Directory>\PureEdge\Viewer x.x\Extensions\

   New: <Installation Folder>\PureEdge\Viewer 6.0\Extensions\

To upgrade your extensions to the version 6.0 API, you must recompile the source code for your IFX with the new API. Be sure to test the IFX after you have recompiled it, and make sure you distribute the new IFX in the correct folder, as outlined in step 2 of the previous procedure.

Viewer 6.0 – Complete List of Changes

Viewer 6.0 includes the following new and changed features:

- **Added Support for XFDL 6.0** – The Viewer respects all changes and additions to XFDL 6.0.
- **Added Support for Netscape 7** – The Viewer now supports Netscape 7.
- **Dropped Support for UFDL** – The Viewer no longer supports the Universal Forms Description Language (UFDL). Furthermore, files with a .ufd or .ufdl extension are no longer recognized and the *application/vnd.ufdl* MIME type is no longer supported.
- **Dropped Windows 95 Support** – Windows 95 is no longer supported and the Viewer is no longer tested in this environment. (This was first introduced in version 5.2.)
- **Deprecated .frm File Extension** – The .frm file extension has been deprecated. This means that all new forms should use the .xfd or .xfdl file extensions, and that support for the .frm file extension may be dropped in future versions.
- **Limited Backwards Compatibility** – ICS Viewer 6.0 supports XFDL version 4.4 and later. If a form of a version earlier than 4.4 is opened, the Viewer will process it as a version 4.4 form, and will apply the version 4.4 rules to the form.

  Furthermore, the Viewer is only binary compatible with version 4.5 and later products. This means that any Viewer extensions (.ifx extensions) written with the version 4.4 or earlier API will not work in the new Viewer.
- **Changed Serialization of URLs in the *url* Option** – If you list multiple URLs in the *url* option, the URLs are now processed in the order listed (formerly the order was undetermined). For example, if a button of type submit listed 2 URLs, the submission would process the first URL and then the second URL.

  However, if an action occurs that replaces or closes the original form (for example, a *replace* or *done* action) no further URLs are processed. For example, if a button of type done listed 2 URLs, the submission would process the first URL then close the original form, preventing the second URL from being processed.
- **Removed Open File Shortcut** – The Viewer no longer supports the ALT-CTRL-F key sequence for opening new files.

- **New Character Set Support** – The Viewer now supports multiple character sets for fonts, as specified by the *fontinfo* option. Currently, this includes the Symbol character set.
- **Changed Activation of the Screen Reader** – The screen reader is no longer activated through a selection in the Viewer Preferences. Instead, the Viewer always makes screen reader information available. If JAWS is active, it will automatically recognize and read this information.
- **Changed Email Default** – The Viewer now defaults to using the active MAPI client when sending email. If you do not have a MAPI client available, you can change this setting in the Viewer Preferences.
- **Added Zoom Feature** – The Viewer now supports the ability to zoom forms from 25% to 200% of normal size. This feature is controlled by buttons on the Viewer's toolbar.
- **Changed Sizing of Items with *label* Options** – For version 6.0 forms, items for which the label option is set, but that have no border on the label, are now two pixels taller. This additional height accounts for the space the border would use if it were there.
- **Changed Shortcut Keys**

  The following shortcut keys have been changed:

CTRL + I	No longer opens the about dialog. It now toggles italics in rich text fields

- **New Shortcut Keys**

  The following shortcut keys have been added:

CTRL + B	Toggles bold in rich text fields.
CTRL + F	Opens the Font dialog, which allows you to adjust font properties in rich text fields.
CTRL + G	Opens the Paragraph dialog, which allows you to adjust indent and justification in rich text fields.
CTRL + U	Toggles underline in rich text fields.
CTRL + R	Right justify text in rich text fields.

CTRL + SHIFT + PLUS	Increases the zoom (magnification) of the form.
CTRL + MINUS	Decreases the zoom (magnification) of the form.
CTRL + SHIFT + ?	Opens the about dialog.

- **User Preferences No Longer Copied During Upgrade** – The Viewer no longer copies existing user preferences when upgrading from a previous version. (This was first introduced in version 5.2.)
- **Changed 'X' Style Check Boxes** – X Style check boxes now adjust the thickness of the X according to the size of the check box. This results in a thicker and more noticeable X as the size of the check box increases. (This was first introduced in version 5.2.)
- **Fixed Wildcards for Certificate Filtering in *signdetails* Option** – Fixed a problem that prevented wildcards from functioning properly when used in the *signdetails* option to control certificate filtering. (This was first introduced in version 5.2.)
- **Fixed Problem Opening Forms with IE6 Under Windows XP** – Fixed a problem with IE6 under Windows XP that caused forms to open in standalone mode rather than in the browser. (This was first introduced in version 5.2.)
- **Updated *toggle* Function** – The implementation of the toggle function has been updated to resolve outstanding problems. This may affect some forms that rely on unusual behavior.
- **Changed Default Height for *extent*** – Previously, if you did not provide a height for an extent (for example, if you used a compute to calculate the height but did not provide a starting value) the height would default to equal the width. The height will now default to zero if no initial value is provided.
- **Added New ufv_settings** – The Viewer now supports the following new ufv_settings:
  - *aboutboxtext* – Allows you add information to the Viewer's about box. This is useful if you want to add contact information, such as a name, address, and phone number.
  - *helpcursor* – Allows you to control the appearance of the pointer when the Viewer is in help mode. The pointer can appear either as an arrow or an arrow with a question mark.
- **Changed Behavior of Buttons and Labels with Empty Images** – If a button or label's image option points to a data item that is empty or does not exist, then the button or label will display its value option instead. Formerly, the button or label displayed nothing.

XFDL 6.0 – Complete List of Changes

The following list summarizes that changes and additions to XFDL. For further information about these changes, refer to the *XFDL Specification*.

- **Added Namespace Support** – XFDL now supports namespace. All XFDL syntax is in the XFDL namespace, which is defined by the URI: *http://www.PureEdge.com/XFDL/6.0*. All other information, such as custom items and options, must be in another namespace. You can define any namespace you like or use the custom namespace provided by PureEdge, which is defined by the URI: *http://www.PureEdge.com/Custom*.

- **Removed *version* Option** – XFDL no longer supports the *version* option. Instead, the version of the XFDL is defined in the namespace declaration for the form. However, you can still reference a form's version using the *global.global.version* reference.

- **XML Data Model** – XFDL now supports an XML Data Model. This model allows form designers to create separate blocks of arbitrary XML within an XFDL form that share data with form elements, such as fields. This is useful for integrating with applications that require data in a particular XML format, such as schema-compliant data. The XML Data Model is based on the XForms standard, as published by the W3C, but is not limited to XForms. Refer to the *XFDL Specification* for more information.

- **Changed Compute Syntax** – XFDL no longer uses the <compute> tag to define computes. Instead, computes are defined as attributes for the option you are setting. For example, a *value* option set with a compute might look like this:

  ```
  <value compute="page1.nameField.value">Jane E. Smith</value>
  ```

  Additionally, because computes are now enclosed in quotation marks (“), elements of the compute that used to be enclosed in quotations marks, such as strings, are now enclosed by apostrophes ('). For example, the in the following compute the value 10 is enclosed in apostrophes:

  ```
  <value compute="page1.amountField.value + '10'"></value>
  ```

- **Change to Syntax for If/Then/Else Computes** – If you are using an if/then/else compute that contains a namespace in the "then" portion, you must enclose the "then" portion in parentheses, as follows: *x ? (namespace:y) : z*.

If you do not enclose the "then" portion in parentheses, the parser will mistake the colon following the namespace prefix as the end of the "then" portion, and the compute will not function properly.

- **Maintaining Formatting of Computes** – To maintain the formatting of computes across multiple lines, you must place a newline entity reference wherever you want to break a line. The newline entity reference is *
*. For example:

```
<custom:my_opt xfdl:compute = "toggle(activated, &#xA;
   'off', 'on') == '1' ? &#xA;
   set('convertedDate.value', &#xA;
   sample_package.convertDate(theDate.value, &#xA;
   theLocale.value)) : ''"></custom:my_opt>
```

- **Changed Allowed Characters for Sid and Element Names** – XFDL 6.0 supports the following Unicode characters in sid and element names (hex notation in parentheses): 0-9 (30 to 39), A-Z (41 to 5A), a-z (61 to 7A), _ (5F), (B7), (C0 to D6), (D8 to F6), (F8 to FF).

  Sid and element names cannot start with: 0-9 (30 to 39) or (B7).

  Note that the $ character is no longer allowed, and that the _ (underscore) character is no longer mandatory in custom item or option names.

- **New Escaped Characters** – You must escape the ampersand (&), apostrophe ('), and less than sign (<) to use these characters in computes or as character data. To escape these characters, use the following encoding:

Character	Escape Sequence
Ampersand (&)	&
Apostrophe (')	'
Less Than (<)	<

- **Addition of ‘or’ and ‘and’ as Reserved Words** – XFDL now includes 'and' and 'or' as reserved words. They represent logical AND (&&) and logical OR (||), respectively.

- **Removal of *content* Attribute** – The *content* attribute is no longer used. Arrays are written in the same way, but without the *content* attribute. For example:

```
<bgcolor>
   <ae>100</ae>
   <ae>100</ae>
   <ae>100</ae>
</bgcolor>
```

  Computes are now written as attributes, rather than array elements. For example:

```
<value compute="Field1.value"></value>
```

- **Added Character Set Support** – XFDL now supports multiple character sets for fonts, as specified by the *fontinfo* option. Currently, this includes the Symbol character set.

- **Changed MIME Type For Forms** – All forms and applications should use the *application/vnd.xfdl* MIME type instead of the *application/x-xfdl* MIME type. While application/x-xfdl will still work with PureEdge software, support for it may be dropped in future versions. Furthermore, the *application/uwi_form* MIME type is no longer supported except when validating signatures.

- **Added Explicit Global Page and Global Items** – XFDL now declares explicit global page and global items. The global page is marked with the "globalpage" tag, and global items are marked with the "global" tag. Both global pages and global items must have a sid of "global". For example:

```
<globalpage sid="global>
   <global sid="global">
      ...global form settings...
   </global>
</globalpage>
<page sid="Page1">
   <global sid="global">
      ...global page settings...
   </global>
   ...page contents...
</page>
```

- **New XFDL Function Calls** – XFDL now supports the following function calls:

- *countDatagroupItems* - Allows you to count the total number of items in a particular datagroup.
- *countGroupedItems* - Allows you to count the total number of items in a particular group.
- *getAttr* - Allows you to get the value of an attribute on any element in a form.
- *getPosition* - Allows you to get the position index of an element within its parent.
- *getPref* - Allows you to get the value of any setting in Viewer Preferences form.
- *getGroupedItem* - Allows you to get the SID (scope identifier) of an item in a particular group.
- *setAttr* - Allows you to set the value of an attribute on any element in a form.
- *xmlmodelUpdate* – Allows you to update the XML Data Model in memory. This is useful if you have made computational changes to the data model.

- **Change to Default Type for Actions and Buttons** – Action and button items now default to a type of *select* rather than a type of *link*.
- **New Encoding Attribute and Encoding Method** – Options and array elements now support an *encoding* attribute that defines the encoding method used for the tag's contents. For example:

```
<mimedata encoding="base64"></mimedata>
```

  Valid encoding methods are: *xml*, *base64*, *base64-gzip*. Note that base64-gzip is a new encoding method that compresses the contents and then converts them to base64. The default encoding is *xml*, and all MIME data is now stored in *base64-gzip* format.
- **Added Support for Rich Text Fields** – XFDL now supports rich text fields. These are enabled through the following new options: *rtf* and *texttype*.
- **Changed Default Display Value for Popup Items** – If a popup does not have a label or a value option, it will be blank. Formerly, the popup displayed the value of the first cell.
- **Added Verifier Flag in *signformat* Option** – XFDL now supports a verifier flag in the *signformat* option. This flag controls the level of rigor applied when verifying certificates. (This was first added in XFDL 5.2.)

- **Added Parent Operator** – XFDL now supports a double dot (..) parent operator for references. The parent operator must appear at the beginning of a reference, and can appear multiple times to indicate additional ancestors. For example, the following are now valid references:

  ```
  ..FIELD1.value
  ....value
  ```

  The first reference gets the parent of the current node, then locates the Field1 node, then the value node. The second gets the parent of the current node, then the parent of that node, then locates the value node.

- **New *signnamespces* and *transmitnamespaces* Options** – XFDL supports two new filtering options: *signnamespaces* and *transmitnamespaces*. These options allow you to filter which namespaces are included in signatures and transmissions.

Designer 2.3 – Complete List of Changes

Designer 2.3 includes the following new features:

- **Added Support for XFDL 6.0** – The Designer respects all changes and additions to XFDL 6.0.
- **Dropped Support for UFDL** – The Designer no longer supports the Universal Forms Description Language (UFDL). Furthermore, files with a .ufd or .ufdl extension are no longer recognized.
- **Dropped Windows 95 Support** – Windows 95 is no longer supported and the Designer is no longer tested in this environment.
- **Deprecated .frm File Extension** – The .frm file extension has been deprecated. This means that all new forms should use the .xfd or .xfdl file extensions, and that support for the .frm file extension may be dropped in future versions.
- **Limited Backwards Compatibility** – The Designer can open forms in any version of XFDL. However, the following limitations apply:
  - *Forms prior to version 4.4* – The Designer can read these forms, but will interpret them as version 4.4 forms and apply version 4.4 rules to them.
  - *Forms prior to version 5.1* – The Designer can read these forms, but cannot display them until they are upgraded to version 5.1 or later. When a form prior to version 5.1 is opened, the user is immediately prompted to upgrade the form.

  Furthermore, the Designer is only binary compatible with version 4.5 and later products. This means that any extensions (.ifx extensions) written with the version 4.4 or earlier API will not work in the new Designer.
- **Removed Custom Item Creation** – The Designer no longer supports the creation of custom items through the normal user interface. To create custom items, you must now use Code View.
- **Designer Retains Window Size** – The Designer now remembers whether the form window should be maximized when opening a form. This setting is based on the window size the user last worked in. For example, if the user maximizes the form window and then closes the Designer, the Designer will open the next form in a maximized window.

- **Designer Sets Fields to Wordwrap by Default** – When you create Fields, the Designer now automatically sets them to wordwrap.
- **New Shortcut Keys** – The Designer now supports the following shortcut keys:

  CTRL + F1 Brings the selected item to the front.

  CTRL + F2 Sends the selected item to the back.

- **Bug Fixes** – A number of bugs were fixed (they were first fixed in the version 2.2.1 release). For more information, refer to the Designer 6.0 documentation.

API 6.0 – Complete List of Changes

ICS API 6.0 includes the following new features:

- **Added Support for XFDL 6.0** – The API respects all changes and additions to XFDL 6.0.
- **Dropped Support for UFDL** – The API no longer supports the Universal Forms Description Language (UFDL). Furthermore, files with a .ufd or .ufdl extension are no longer recognized.
- **Added Support for Solaris 2.8** – The API will run on Solaris 2.7 and 2.8.
- **Dropped Windows 95 Support** – Windows 95 is no longer supported and the API is no longer tested in this environment.
- **Deprecated .frm File Extension** – The .frm file extension has been deprecated. This means that all new forms should use the .xfd or .xfdl file extensions, and that support for the .frm file extension may be dropped in future versions.
- **Limited Backwards Compatibility** – The API supports XFDL version 4.4 and later. If a form of a version earlier than 4.4 is opened, the API will process it as a version 4.4 form, and will apply the version 4.4 rules to the form.

  Furthermore, the API is only binary compatible with version 4.5 and later products. This means that any extensions (.ifx extensions) written with the version 4.4 or earlier API will not work in the new API.
- **Added and Updated Data Types (C API)** – The C API includes a number of new data types for naming consistency and multibyte support. In general, the new data types will not impact existing applications, as the API is binary compatible with previous versions. However, if you need to recompile your code under the version 6.0 API, you must make the following changes:

  - change charP to r_charP
  - change voidP to r_voidP

  You should refer to the API documentation for more information about these changes.
- **Changed Java API Naming Conventions** – The package name for all objects has changed to: *com.PureEdge*. Class names in the new package begin with an uppercase character. For

example: *com.PureEdge.StringHolder*. Method names in the new package now begin with a lowercase character. For example: *getLiteralByRef*.

The old *com.uwi* package is still available, and still uses the old naming conventions. However, the old package does not contain the new API functions. Furthermore, the old package has been deprecated, and support for it may be removed in a future release.

- **Added Flags to *readForm* Function** – The following new flags have been provided for readForm:
  - UFL_AUTOCOMPUTE_OFF – This turns off the compute system, which speeds up load times.
  - UFL_SERVER_SPEED_FLAGS – This turns off the following features: computes, automatic formatting, duplicate sid detection, the event model, and relative page and item tags (for example, itemprevious, itemnext, and so on). This is intended to decrease server processing times.
- **Deprecated and Extended Functions** – The following functions have been deprecated and replaced with extended functions: *getLiteral*, *getLiteralByRef*, *setLiteral*, *setLiteralByRef*, *getInfo*, and *dereference*. While the deprecated functions continue to work in this version, you should begin to use the Extended functions as they process more quickly. Furthermore, support for the deprecated functions may be dropped in future versions. (Note that when using the new Ex functions under C, the caller must free the returned string values.)
- **Added New Functions** – The API includes a number of new functions to support the addition of namespace and to provide other functionality. You should refer to the API documentation for more information about these functions.
- **Renamed *functionCallEvaluate*** – The *functionCallEvaluate* method has be renamed to *evaluate* in the com.PureEdge package. The old method name still works with the com.uwi package.
- **Change to *encloseFile* Encoding Method** – The *encloseFile* function now uses base64-gzip encoding when creating the *mimedata* option that contains the file. Formerly, it used base64 encoding.
- **Updated *toggle* Function** – The implementation of the toggle function has been updated to resolve outstanding problems. This may affect some forms that rely on unusual behavior.

- **Added New Configuration File** – The API now relies on a configuration file to set certain behaviors. For more information, refer the API 6.0 *Installation and Setup Guide*.
- **Bug Fixes** – A number of bugs were fixed (they were first fixed in the version 5.2 release). For more information, refer to the API 6.0 release notes.

How to Contact PureEdge Solutions

Electronically

Visit our web site at: **www.PureEdge.com/**.

For support, send email to: **support@PureEdge.com**.

For purchasing or sales information, send an email to **info@PureEdge.com**.

You can find documentation for other PureEdge products on our documentation web site at: **http://docs.PureEdge.com/**

By Mail

4396 West Saanich Road
Victoria, BC
V8Z 3E9 Canada

By Phone

Telephone: 1-250-708-8000
Fax: 1-250-708-8010
Toll Free: 1-888-517-2675