KinGUI Reference for Programmers

Table of Contents

KinGUI window layout definition

Application “windows” and their layout are defined by XML blocks. Current DTD is available as Annex A at this document.

An application is defined by one (or maybe some) file or XML block. Several files (or blocks) define several applications. Empty application is allowed (with no *dialog* tags) to provide a simple list with desired windows and their order (that should be defined onto other files).

Here we describe every keyword and concept, and their usage to define a window layout. For a complete example, read the supplied examples in the project source code (like in `kingui/src/examples/guidlg/dd_1.xml`).

We will use file or block as the same thing: a complete XML document.

Units to define position and/or measures are “base units”, and given as tens of font height units (10 is the height of current system font).

Note that this document in oriented to KinGUI environment, but this description files are also valid (and used) for other frontends (like the http one). Tags not related to KinGUI are not explained here in detail.

Remember that XML is case *sensitive*.

You may find a few examples at Annex E.

Note that currently, sources are expected in UTF-8 encoding (other encodings are ignored for now).

TAG usage

Let's review every defined tag and its usage:

<application>

This is main tag, should be the first one found in an XML block, and also the closing one. It defines the application described into this block. Valid attributes are:

- **name**: gives the application internal (unique) name or ID. This is also the name displayed into its title area (tab), but it is unlikely to be this way in the future (probably there will be a <title> tag for that function).
- **windows**: defines a list of public initial windows (defined into this block). KinGUI will automatically open such windows at startup. Often you won't use this option, as you'll prefer to open windows on your own pace. A typical use is to have a main application window, as a root for all the other processes for this application.
- **color**: this defines background application color. Used when you want to highlight this application, or you have some sort of color keying of them. This color is used as application tab background also.

An application block must have at least one <dialog> tag inside.

<title>

This tag defines the window (and in some future, application) title (human readable name). This has nothing to do with window (or application) *name* attribute, but is just the name displayed in its tab.

<initquery>

This tag defines one query to perform to the DB to get initializing data for the dialog. It may appear more than once, so several queries can be issued. Results are used through the *fillfmt* attribute in several tags. It only allows for one attribute:

- **id**: letter that identifies this query (from 'A' to 'Z', case insensitive).

<dialog>

This tag defines a “window”, an application “form” or working area. Every dialog has its own (vertical) tab into its application. Although it is defined as belonging to this application, it may also be recalled from another one (by its name), so it is a good idea to keep its name/ID unique along your whole system. Valid attributes are:

- **name**: gives dialog internal name or ID. This is not to be mistaken with the window name displayed in its tab (which is given by <title> tag). Dialog names should be unique along the whole system.
- **width**: is an indication of intended window width, in base units. In KinGUI, this is ignored.
- **height**: is an indication of intended window width, in base units. In KinGUI, this is ignored.
- **color**: defines an optional background color for this window. This includes working area, and horizontal window tab, and is often used to highlight this window or make an association with a specific color.

A dialog has a compulsory <title> tag, and several window item definition tags.

<taborder>

This tag defines the window tab order, that is, the order to use when jumping from one window item to another, by using the *TAB* (or *shift-TAB*) key. It contains a simple list of unsigned integers, separated with commas, with the window item Ids that we want in the TAB sequence.

<group>

This is a powerful tag, that allows you to group several items (or other groups) in some fashion. Using groups to design dialog layout gives you the ability to make window items position relative to each other, and to move blocks of them accordingly. Groups have two kinds of distribution (how items are layed out inside them): horizontal or vertical. This means that items are sequentially set from left to right, or from top to bottom.

Groups have some attributes that allow them to be visible, like a border (frame) and a name (title), so they may also be used to visually group related items (like some radio buttons). They also may have an ID, so your application may even change their displayed name at any time.

Valid attributes are:

- **name**: group name. If defined, group is displayed and this name is shown on the top left corner (by now). Having a name makes a group to grow one unit at its top (to hold such title).
- **id**: group ID (number). If defined, allows the application to set or get its name, for an instance.

- **distribution**: this is a compulsory attribute, with two posible values: *horizontal* or *vertical*. The first makes items layout to go from left to right, the former does the same from top to bottom.
- **posx**: defines where to set the top-leftmost corner of group, in the x axis (horizontal). This allows you to move this block around the working area. Value is a number, in base units.
- **posy**: defines where to set the top-leftmost corner of group, in the y axis (vertical). This allows you to move this block around the working area. Value is a number, in base units.
- **width**: defines block width, in base units. This is seldom needed, as group grows to the required size to hold its items.
- **height**: defines block height, in base units. This is seldom needed, as group grows to the required size to hold its items.
- **spacing**: defines how many base units to insert between items. This tag is used most of the times.
- **border**: defines the kind of border (frame) we want around the group area. 0 means none, 1 means a simple one, 2 is for one with some shadow. Defining a border makes a group to grow half a unit in each corner.
- **color**: defines a background color for this group, if desired. Note that color covers all the group area, even out of its frame (if borders are required).
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **hint**: this is still unused in a group.

Usually, a group definition only has a *distribution* tag, and often a *posx* and a *posy*.

<statictext>

This simple item just draws a text (that is not modificable by the user, but that can be set by the application). It may also be used as an active area to open an overlay window (to edit the value or combination, for an instance). Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. Default is 10 (one height), so this value is present most of the times.
- **height**: defines item height, in base units. Default is 10 (one height), so this value is seldom needed.
- **align**: defines text alignment into the item area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **valign**: defines text alignment into the item area, in the vertical axis. Default is *top*,

and may also be *bottom* or *center*.

- **id**: this optional value helps application to access to item (to retrieve or set its text value, for an instance, or to know who is generating a notification when clicked)
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **color**: set the background color for the item area. Often used to show an active item (that can be clicked and can retain focus).
- **border**: defines the optional kind of border (frame). It accepts the same numeric values as <group> tag (0=none, 1=simple, 2=shadowed).
- **enabled**: if set to yes, item is enabled (modifiable), else, with no value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

Furthermore, its content can be specified as tag content (between <statictext> and closing </statictext>).

<edittext>

This text item draws a text that is modificable by the user. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. Default is 10 (one height), so this value is present most of the times.
- **height**: defines item height, in base units. Default is 10 (one height), so this value is seldom needed.
- **align**: defines text alignment into the item area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **id**: this value helps application to access to item (to retrieve or set its text value, for an instance, or to know who is generating a notification when modified or CR key is pressed)
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **password**: if specified, and with value *yes*, this makes all the show characters to be 'x', thus hidding the real typed ones.

- **maxlen**: defines the maximum size (in bytes) that will be allowed for this item content (text). This is not the same as *maxchr* value when typing non-ascii characters.
- **maxchr**: defines the maximum number of characters that are allowed into this item content. This is not the same as *maxlen* value when content has some non-ascii values (that need more than one byte for a single char).
- **maxlin**: defines the maximum number of lines that can be written into this item (default is single line, that is, *maxlin*=”1”).
- **enabled**: if set to *yes*, item is enabled (modifiable), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to *yes*, item is visible, else, with *no* value, it is hidden. Default (when this attribute is not present) is yes.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).
- **option**: defines the importance for this item. Default is *optional*, and may also be *required*.

Furthermore, its content can be specified as tag content (between <edittext> and closing </edittext>).

<editnum>

This text item draws a number that is modificable by the user. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. Default is 10 (one height), so this value is present most of the times.
- **height**: defines item height, in base units. Default is 10 (one height), so this value is seldom needed.
- **align**: defines text alignment into the item area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **id**: this value helps application to access to item (to retrieve or set its value, for an instance, or to know who is generating a notification when modified or CR key is pressed)
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.

- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **maxlen**: defines the maximum size (in chars) that will be allowed for this item content (as text).
- **enabled**: if set to *yes*, item is enabled (modificable), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).
- **option**: defines the importance for this item. Default is *optional*, and may also be *required*.
- **type**: defines the kind of number it holds. It can be *integer*, *float, currency*, *iban* or *time* and is *integer* by default. Note that integer may be also fixed point integer, if 'numdec' is specified.
- **numdec**: defines how many decimals you want (limited to 15 maximum). Default is no decimals at all (or default for choosen 'type'). Note that for type *time* this is the number of colons (currently, this must be 1 or 2, and defaults to 1 if not specified).
- **thsep**: defines if you want to display numbers with thousands separators or not. Default is *yes*, and may also be *no*.
- **ptxt**: defines a short text to append to editable value. This text is not editable, and is intended to hold units or similar information (like a percent sign on a percentage value). Currently, it is about 7 bytes long so it can accomodate any utf-8 symbol or short unit name.
- **limit**: defines numeric limits for input; this are two comma separated numbers, as lower and upper limits. If only one number is found, it is assumed as (-N,0) if negative, or (0,N) if positive. Default is no limits at all.

  For integer type, no decimal point is expected (limit is just the whole number including all its decimals, that is, if two decimals are in effect, 1.00 will be written as 100). For floating point type, a couple of floating point values are expected.

Furthermore, its content can be specified as tag content (between <editnum> and closing </editnum>). If not recognized, item is filled with an empty value.

<listbox>

This item defines a complex entity, composed by cells, arranged in rows and columns. Furthermore, it can arrange rows in diferent depth levels, to form a tree list. It allows for single or multiple selection of rows, and even for user editing of single cells. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.

- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case, as a list with one char length is unuseful.
- **height**: defines item height, in base units. Default is 10 (one height), so this value is present most of the times (a list with one row height is seldom very useful).
- **multipleselect**: if specified, makes this list to hold several rows selected. Valid values are *yes* or *no* (which is the default if this attribute is not present).
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **color**: set the background color for the item area. Often used to show an active item (that can be clicked and can retain focus).
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **enabled**: if set to *yes*, item is enabled (selectable and/or modificable), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **sorted**: May be set to *ascending* (A->Z) or *descending* (Z->A). Any other value is the same as the default: not sorted (and in such case, *sortcol* is ignored).
- **sortcol**: Column index to use as a reference for sorting (e.g. the column to sort upon). Ignored if no *sorted* attribute is present.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **depth**: optionally specify to what depth the list is unfolded, when this is a tree list (0 means only root entries, 1 means that and their first level children, etc.)
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

Between <listbox> and closing </listbox>), you can specify list column definitions, and even list row content to fill up all the item content. First, you must define columns with <cdef> tags, then assign every cell value with <li> and <cd> tags (see below for their respective descriptions).

<combobox>

This item defines a single unfoldable selection list. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical).

Usually it is not used with groups, as group already sets every item position.

- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case, as a list with one char length is unuseful.
- **height**: defines item height, in base units. Default is 10 (one height), so this value is present most of the times (a list with one row height is seldom very useful).
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **enabled**: if set to *yes*, item is enabled (can select), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **sorted**: May be set to *ascending* (A->Z) or *descending* (Z->A). Any other value is the same as the default: not sorted.
- **lines**: this is the number of lines that will be visible when list is opened.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

Between <combobox> and closing </combobox>), you can specify list column definitions, and even list row content to fill up all the item content. First, you must define columns with <cdef> tags, then assign every cell value with <li> and <cd> tags (see below for their respective descriptions). It works like in <listbox> item, but here (at least by now), you can only specify one column, and no column definition is needed (it is expected to be “text” type).

<cdef>

This tag defines a single column into a <listbox> (or even a <combox> in some future). It never can appear after a <li> tag (so columns must be defined before data is). Valid attributes are:

- **width**: defines column width in base units.
- **align**: defines horizontal data alignment into this column. Valid values are *left* (default), *right* or *center*.
- **type**: defines the kind of data that this column will hold. Valid values are:

    - *num4* (a 32bit integer). It allows a **subtype** attribute, with a number with the following bits set: b4..7 for the decimal points you desire (this is: value is fixed point integer); b8=1 => use thousands separator; b9=1 => integer is signed. b10+11 represents the kind of number, where 0=standard, 1=percentage, 2=size (5 char, with multiplier -K,M,G,T,P-).
    - *text* (simple UFT-8 text)
    - *atext* (same as before but with attributes, see <cd> tag attributes for valid values).
    - *uDateTime* (32bit DateTime). It allows a **subtype** attribute, with a number from 0 to 5 for the desired display format, encoded as follows: 0=DateTime, 1=Date, 2=Time, 3=Date w/2digit year, 4=Time HH:MM, 5=32bit time HH:MM:SS.DDDD. Note that for subtype 2, 4 and 5, value is a time value instead of a timestamp. Unix style time stamp is seconds since 1/1/1970 0:00:00, and long time is 1/10000s since 0:00:00.0000. There's also subtype 6 and 7, for duration values (H:MM:SS and H:MM respectively)
    - *sDate* (short -16bit- date or time). The **subtype** attribute works like in previous case. Note that for subtypes 2, 4 and 5, value is a short time instead of a short date. Short date is days since 1/1/1900, and short time is double seconds since 0:00:00. There's also subtype 6 and 7, for duration values (H:MM:SS and H:MM respectively).
    - *icon* (16bit icon ID)
    - *id32* (32bit generic user ID)
    - *money* (64bit currency, no decimals)
    - *money2* (64bit currency, 2 decimals)
    - FP32 (32bit floating point). The **subtype** attribute can give the number of decimals (up to 15, set at b4..7)
    - FP64 (64bit floating point). The **subtype** attribute can give the number of decimals (up to 15, set at b4..7)
- **visible**: if present and set to *no*, this column is never displayed. If set to *yes* or this attribute is not present, column is displayable (although it may be hidden). This is useful for columns that are not intended for user but for the system, like row ID, internal data, etc.
- **editable**: if set, every cell in this column can be modified by direct user action (doubleclicking it). Valid values are *yes* or *no*, the former being the default value when this attribute is not specified.
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).

<li>

This tag allows you to specify a complete row in a listbox or combobox item. It includes one or more <cd> tags that specify every cell values. Valid attributes are:

- **selected**: if present and set to *yes*, this makes this row to be selected. If list is single selection (or it is a combobox), the last row with this tag is the one finally selected. If not present or set to *no*, row is unselected.
- **child**: used to specify depth level for this row in a multilevel tree list. Valid values are numbers, 0 meaning same level (this is the same than not setting this attribute), 1 meaning to be one level deeper than previous row (so this row is a child of

preceeding one), and bigger values meaning to go up one or more levels: 2 to escalate one level (so be at the same level as previous line parent), 3 to escalate two levels, etc.

- **id**: used to specify combobox row ID. It has no use in a listbox (where you can use any column as an ID).

This tag's content is a sequence of one or more <cd> tags.

<cd>

Column data tag, specifies data value for one cell in a listbox or combobox. It must conform to the previous definition for such column. It only appears into <li> tags, being first <cd> tag the value for first column, second for second column, and so on. Valid attributes are:

- **color**: sets text (not background) color, provided that this column is of type *atext* (text with attributes). Invalid in all other cases. Color value is in the form "#RRGGBB".
- **tattr**: specifies text attributes for this cell, provided that this column is of type *atext* (text with attributes). Invalid in all other cases. Valid values are *normal* (same as when this attribute is not present), *bold* for bold text, *italic* for italic text, and *bolditalic* for both styles.

<button>

This item defines a pressable button, that shows a name, and (optionally) an icon and color. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case.
- **height**: defines item height, in base units. Default is 10 (one height).
- **align**: defines text alignment into the button area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **color**: set the background color for the item area.
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **img**: if specified, defines an image to display over the button background (like an icon).
- **imgw**: if specified, defines the button image horizontal size, in base units. Image is resized to this value if bigger

- **imgh**: if specified, defines the button image vertical size, in base units. Image is resized to this value if bigger
- **imgm**: if specified, defines how button text interacts with button image. Default is *overlay* (image does not affect text area), and may be set to *exclusive* (image area is excluded from text area).
- **imghp**: if specified, defines image horizontal position into button. Default is *left*. It may also be *right* or *center*.
- **imgvp**: if specified, defines image vertical position into button. Default is *top*. It may also be *bottom* or *center*.
- **enabled**: if set to *yes*, item is enabled (can select), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

<checkbox>

This item defines a selectable checkbox, that shows a name, and (optionally) a color. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case.
- **height**: defines item height, in base units. Default is 10 (one unit height).
- **selected**: if present and set to *yes*, this makes this checkbox to be selected. If not present or set to *no*, checkbox is unselected.
- **align**: defines text alignment into the item area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **color**: set the background color for the item area.
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.

- **enabled**: if set to *yes*, item is enabled (can select), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

<radio>

This item defines a mutually exclusive selectable checkbox in a group, that shows a name, and (optionally) a color. It is often used to quickly choose an option between a few posibilities. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case.
- **height**: defines item height, in base units. Default is 10 (one unit height).
- **selected**: if present and set to *yes*, this makes this checkbox to be selected. If not present or set to *no*, checkbox is unselected.
- **align**: defines text alignment into the item area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **gid**: this value should be present, and just identifies a selection group: all the radio buttons within the same group are mutually exclusive for selection (if you select one, the others get unselected). This value is a number, and has no relationship with any other ID (item, group) in the dialog, although it is recommended to choose a unique value for readability.
- **color**: set the background color for the item area.
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **enabled**: if set to *yes*, item is enabled (can select), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.

- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is present. Default (when this attribute is not present) is disabled.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

<datetime>

This item defines a date, time or combined date and time item. It opens a calendar on activation (for date or combined formats), and also allows for direct key entry or mouse wheel independent adjustment for every parameter (day, month, year, hour, minutes). Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case, and as a guidance, a combined format requires a value of about 80 (8.0 units).
- **height**: defines item height, in base units. Default is 10 (one unit height), and is the only sensitive value (so it should be not present and left as default always).
- **align**: defines text alignment into the item area, in the horizontal axis. Default is *left*, and may also be *right* or *center*.
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **type**: defines which kind of date/time item is this. It may be *date* for an only date managing item, *time* for an only time one, or *datetime* for a combined date and time control.
- **subtype**: defines the storage kind. Current possible values are *udatetime* (32bit), *xdatetime* (32bit) or *sdate* (16bit).
- **color**: set the background color for the item area.
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **enabled**: if set to *yes*, item is enabled (can select), else, with *no* value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **offline**: if set to *disabled*, item is disabled (not modifiable) when a valid connection to the database is not present, and restored to its expected state when connection is

present. Default (when this attribute is not present) is disabled.

- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

<graph>

This item defines a (graphic) chart item. It shows a graphic representation of a multidimensional array of values. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical). Usually it is not used with groups, as group already sets every item position.
- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is compulsory in this case, as default value (10) is too small for any kind of chart.
- **height**: defines item height, in base units. Default is 10 (one unit height), and it must be specified to a bigger value so chart can fit.
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **type**: defines which kind of char is displayed. Currently, only type *vbars* is operative, but there's also other types in development: *hbars*, *lines*, *surfaces*, *addingsurfaces*, *addinghbars*, and *addingvbars*.
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **enabled**: if set to yes, item is enabled (modifiable), else, with no value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

<image>

This item defines an image item. Valid attributes are:

- **posx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal). Usually it is not used with groups, as group already sets every item position.
- **posy**: defines where to set the top-leftmost corner of item, in the y axis (vertical).

Usually it is not used with groups, as group already sets every item position.

- **disx**: defines where to set the top-leftmost corner of item, in the x axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **disy**: defines where to set the top-leftmost corner of item, in the y axis (horizontal), with respect to its expected position. Often used to displace item slightly.
- **width**: defines item width, in base units. This attribute is often specified, as default value (10) is too small for most of the images.
- **height**: defines item height, in base units. Default is 10 (one unit height), and is usually overriden by a bigger value so image can fit.
- **id**: this value helps application to access to item (to retrieve or set its row values, for an instance, or to know who is generating a notification when something is selected, etc)
- **key**: if specified, defines the active key that makes this item to get the focus. Not yet implemented.
- **fillfmt**: instructions to extract initial value from a DB query (see appendix B).
- **enabled**: if set to yes, item is enabled (modifiable), else, with no value, it is visible but can't be modified. Default (when this attribute is not present) is yes.
- **visible**: if set to yes, item is visible, else, with no value, it is hidden. Default (when this attribute is not present) is yes.
- **hint**: defines a user clue (hint) for this item. When user presses left control key and this item is the focused one, a yellow note is shown with this text. Note that a static text item only can have focus when is an active one (currently, when it has a background color).

<include>

This item instructs to include an item from any other <dialog> in place of this tag (this is very useful to keep corporative items being the same across several windows, by just defining them in one place and including in all the other windows, or just to save work when you use the same block of items -group- in several places). Valid attributes are:

- **window**: defines the <dialog> where to look for given item.
- **id**: defines the item to duplicate

<script>

This tag defines a script for dialog management. See Annex C for a complete description of its syntax and available internal methods. It is very common to use a <![CDATA[...]]> enclosing tag to easy writing and readability (so no escapes are needed when writing a simple condition like (a<b), for an instance). There may be some <src> tags inside, including some external files as code, usually in the form <src ref=”loal_file.js”/>, mixed with some explicit code (order is kept, external code is inserted exactly in place of the <src> tag).

Note: for now, a CDATA section overrides any subsequent tag until the </script> terminator. This will be fixed some time in the future, but till then, please don't use such kind of section mixed with <src> tags.

<src>

This tag defines an external reference. For now, it is only allowed into a <script> tag, and refering to a relative local file. There may be several so many files can be included.

- **ref**: defines file to load. External files will often use the *js* extension, as they are basically JavaScript syntax equivalent to embed into current <script> section.

KinGUI JavaScript reference

Every dialog definition may include one script containing code to manage it, and perform all or part of its logic (user interaction, queries, etc.).

You may find some examples in KinGUI resource files examples, at Annex E.

Objectives

1. Simplify GUI code, fast development
2. Portability: same code for KinGUI and for Web interface

Guidelines

1. Code should be compatible with JavaScript (ECMA-262, but can be more restrictive or a subset).
2. Predefined functions in KinGUI may be included on an external js file for Web browser access.

This is: for KinGUI, there are a few objects predefined, to allow access to the window items, queries, etc. For Web interface, these are really declared as objects on an external JavaScript file (if not executed by the server instead).

Language definition

Statements

Are the same as in JavaScript. They end at the semicolon, or they are enclosed in curly braces {}. To avoid mistakes, they do NOT end at just the end of line as in real JavaScript (use semicolon instead).

- break [label]

  Breaks execution of current loop, switch or label or if statement. Inside if statement, only the labelled break is allowed.

- comment

  Single line preceeded by double slash

  Multiple line enclosed into slash+asterisk and asterisk+slash

- continue [label]

  Similar to break, but restarting loop instead of exiting it (only applicable to loop statements)

- do {} while (condition)

  Loop executing statements until condition is not met (evaluates to false)

- for ([initial-expression];[condition];[increment-expression]) {}

  Loop with initialization and increment expressions.

- function name([param][,param][...,param]) {}

  Declares a function

- if (condition) {} [else [if (condition)] {}]

Executes statements if condition is met, else executes the other statements (if provided). The if () else if () doesn't work well yet, so it is recommended to use if () {} else { if () {} else {} } form instead.

- label

  Identifies an statement (see break or continue)

- return expression

  Exits and (optionally) specifies value for a function.

- switch (expression) {case label: statements; break; case ...; default: statements;}

  Like C switch, but label can be anything (like a string).

- var name [=value]

  Declares a variable, optionally initializing it. It is also allowed to use the *new* keyword after the equal sign, preceeding BB() or Array() reserved objects to create such a variable. Note that using var keyword, you mean you want to declare a *local variable* (except in global scope, where it is obviously global and this prefix is optional). A very common mistake is to forget using *var* preffix for local variables (and thus, becoming global variables). So as a rule of thumb, use *var* always except when declaring a global variable inside a function (and that should be quite uncommon). Note also that the *var* prefix is only required in the first (linear) appearance of the variable into given function, irrespective of being executed or not: that is, a construct like “if (0) var x=1; else x=2;” is perfectly valid, and x is always local. This is because variables are declared at compile time, and not when executing code.

- while (condition) {}

  Executes statements while condition is met

- with (object) {}

  Sets default object for enclosed statements (not yet implemented, and won't be in short future).

Operators

- Arithmetic:

+	Adds 2 numbers
++	Increment (pre or post)
-	Unary: negation. Binary: substracts 2 numbers
--	Decrement (pre or post)
*	Multiplies 2 numbers
/	Divides 2 numbers (result is floating point)
%	Remainder of dividing 2 numbers

- String:

+	Concatenates 2 strings

+=	Appends one string to another

- Binary block:

+	Adds a value (second and later operands, are often numeric) to a binary block (first operand). Second operand may also be a string, or another binary block with the same granularity, that is appended then.
+=	Appends numeric or string value to a binary block

- Logical:

&&	Logical AND
||	Logical OR
!	Logical NOT

- Bitwise:

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Left shift, inputs zeros on the right
>>	Right shift, propagating sign (top bit)
>>>	Right shift, no sign propagation (inputs zeros on the left)

- Assignment:

=	Assigns second to first operand
+=	Adds 2 numbers and assigns result to first
-=	Substracts 2 numbers and assigns result to first
*=	Multiplies 2 numbers and assigns result to first
/=	Divides 2 numbers and assigns result to first
%=	Calculates modulus of 2 numbers and assigns result to first
&=	ANDs 2 numbers and assigns result to first
|=	ORs 2 numbers and assigns result to first
^=	XORs 2 numbers and assigns result to first
<<=	Left-shifts 2 numbers and assigns result to first

>>=	Right-shifts 2 numbers with sign extension and assigns result to first
>>>=	Right-shifts 2 numbers with no sign extension and assigns result to first

- Comparison:

==	True if operands are equal
!=	True if operands are not equal
>	True if first operand is greater than second
>=	True if first operand is greater or equal than second
<	True if first operand is smaller than second
<=	True if first operand is smaller or equal than second

Types

- boolean

  (true|false)
- number

  Integer (signed/unsigned) and floating point (both are 64bit)
- currency
- string
- binary block

  function
- object

  TBD

Variables

Does NOT allow for redeclarations (first takes precedence) on base variables, but it is allowed to redefine arrays (content but not type for every cell).

Expressions are evaluated with standard operator precedence, and result typing propagates left-to-right in partial results; e.g. (currency+integer → currency) but (integer+currency → integer). This is a very common mistake that leads to unexpected variable values, especially when you use them in functions that expect a specific type.

5. Note that a variable declaration not preceeded by *var* reserved word will be declared global, and that is not recommended if you really don't need it to be global (so use *var* if in doubt).

Reserved words, variables, etc. are all **case sensitive**.

Predefined objects (in KinGUI)

Handlers

Note that most objects exist once you provide a handler for it (e.g. you declare them).

Then, they are called whenever required. For an instance, if you need to set some window item state when window is created, you can declare *win.Init()* and write code inside to do so. That object will be called when window is created, automatically.

Handlers are provided by you, and executed by the system at a given time.

Predefined methods

On the other side, you can call predefined methods to perform actions (like setting a window item state, for an instance: *win.SetText(100,"Hello world");*). These are the ones available for now:

Built-in

These are built-in objects, at the top level.

Array()

Returns an empty array variable.

BB(rSz)

Returns an empty Binary Block variable, with 'rSz' granularity (item size, in bytes).

Note that rSz is limited to 1, 2, 4 or 8 bytes.

win

This object allows to manage all the window events and more. Implements methods and virtual methods (they exist only if a handler is declared on the script):

Variables

wID

Window ID

Handler methods

Init()

Executed when the window is created (window constructor)

Note that also the initialization of global variables may be performed here instead of being declared and assigned on a global section.

Done()

Executed when the window is removed (window destructor)

Init2()

Executed when user login is accepted (that may happen late, and more than once)

Act.iID()

Executed when an item is activated.

This has virtual methods inside, one for every window item. For an instance, if there's a window item with iID=100, you declare win.Act.i100() to receive Act events for such item specifically. See Annex D to learn when events are generated.

Mod.iID()

Executed when an item state is modified.

This has virtual methods in the same manner as Act(). See Annex D to learn when events are generated.

Mod.Any(iID)

Executed when an item state is modified. This is generated in the same cases as Mod.iID(), but may be a convenient place for common code to avoid writting a handler for every window item. In case both Mod.Any() and Mod.iID() exist, the specific one Mod.iID() is called first.

SelMod.iID(row,col,flg)

Executed when an item selection state is modified. Flg.b0=1 when user doubleclicked on it.

This has virtual methods in the same manner as Act(). See Annex D to learn when events are generated.

LocM.iID(mID,Func)

Executed when an item local menu is invoked. MID is menu selection ID, and Func indicates case where this event is generated (2=menu is activated).

This has virtual methods in the same manner as Act(). See Annex D to learn when events are generated.

Tmr.tID()

Executed when a timer expires.

This has virtual methods in the same manner as Act(), for every tID declared (in this case, an example would be win.Tmr.t1()). See Annex D to learn when events are generated.

ChClose(wID)

Executed when a child window is closed.

wMsg(fwID,mID,msg)

Executed when a (binary) message is received from another window. 'msg' is of type BB (binary block), fwID is window ID of sender window, 'mID' is message ID (user specified). See Annex D to learn when events are generated.

TxtMsg(fwID,txt)

Executed when a text message 'txt' is received from another window (fwID).

VarMsg(fwID,var)

Executed when a variable message 'var' is received from another window (fwID).

Methods

Enable(iID,State)

Sets enabled state for given item. Returns previous state (boolean).

Hide(iID,State)

Sets visible state for given item (state!=0 => hide it). Returns previous state (boolean). Hidding an item also disables it, so when you show it again, you also may want to Enable() it.

SetText(iID,txt)

Sets given window item's caption. Translation is applied to 'txt', but it will probably be not in some future.

GetText(iID)

Retrieves text from given item. Note that if content can be read as a number, returned value is numeric (and not text).

SetNum(iID,Num)

Sets given window item's caption for a numeric value (Num must be integer/fixed point integer, floating point, currency or an IBAN account value -text-, and item should be of type EditNum). Note that for fixed point integer, decimal points are specified into EditNum item, and variable value is just a plain integer. If *iban* value is wrong, no value is set.

GetNum(iID)

Retrieves a number from given item, for an EditNum type item. Result can be integer, floating point or currency, or even text, depending on original item type. Note that for fixed point integer, decimal points are specified into EditNum item, and variable value is just a plain integer (1000 will be returned for a 10.00 value on a two decimal digit EditNum item). Text type is returned for *iban* type item, with the complete IBAN account number including calculated check digits.

Msg(mode,txt)

Sets window message text. Translation is applied to 'txt'.

'mode' is "Neutral", "Wait" or "Error" for now.

Confirmation(iID,mode,txt)

Opens a confirmation popup attatched to given item. This blocks current thread till popup is closed. Return value is 0 if no action was taken, 1 if "yes" or other affirmative action is choosen by the user, 2 if "no", "cancel" or similar.

The only 'mode' allowed values are "Attention", "Error" and "Warning" for now.

OpenWin(wName)

Opens given window as a child window from current one, and returns its ID (or 0 if fails). Note that it is still hidden, use JumpWin() to make it visible if desired.

OpenPopUp(wName,iID)

Opens given window as a pop-up window from current one, and returns its ID (or 0 if fails). iID is item ID to which this popup is attatched.

CloseWin(wID)

Closes given window.

JumpWin(wID)

Swaps to given window making it visible and hidding previous one.

GetMyWinID()

Returns window ID of calling instance

SendTxtMsg(wID,txt)

Sends a text message to window 'wID'

SendBinMsg(wID,mID,msg)

Sends a binary message to window 'wID', with 'mID' message ID. 'msg' must be of type BB (binary block).

SendVarMsg(wID,var)

Sends a variable through a message to window 'wID'

Focus(iID)

Sets focus to given item

FocusNext()

Sets focus to next item (in TAB sequence).

FocusPrev()

Sets focus to previous item (in TAB sequence).

SetFocusAlt(mode)

Changes focus selection method. For now, only mode=0 or 1 are defined. 0 means standard (default), 1 means that pressing “Enter” key on an item that has no win.Act.iXXX manager will perform as pressing the “Tab” key (that is, moving focus to next item).

Clear(iID)

Empties given item.

PaintItem(iID)

(Re)paints given item (useful for items that do not repaint themselves after a change, like a ListBox).

SetModiF(state)

Sets or clears window modification state.

LB

This is an intermediate class to access to ListBox window item. Has the following subclasses:

Clear(iID)

Empties content (but not column definitions) for given ListBox

AddRow(iID,Atr,rDef)

Adds a new row to given ListBox. Row is declared as an array variable, with every column indexed as an array element (starting at 0).

Translation is applied to textual columns. Atr is row attributes:

b0: 1=selected, b1: 1=Hidden, b2: 1=overflow on adjacent empty cells,

b5: 1=hide null values

b6..7: 0=normal, 1=partial sum

AddRowF(iID,Atr,rDef,child)

Same as AddRow(), but it provides the raw tree depth attribute 'child'. Values range from 0=same depth as predecessor, 1=child of predecessor, 2 and higher=reduce depth from predecessor (2=1 level up, 3=2 levels up, 4=3 levels up, etc.)

SetRow(iID,Row,Atr,rDef)

Same as AddRow(), but replaces given row by supplied content

InsRow(iID,Row,Atr,rDef)

Same as AddRow(), but inserts given row just before Row.

GetRow(iID,Row)

Returns (as an array) the whole row, or null variable if not successful. This is faster than using GetCell() several times for the same row. It also fills an entry with index “atr” with row attributes.

GetCRow(iID)

Returns the row index where the cursor is (or 65535 if not applicable)

GetIDRow(iID,ColID,ID)

Returns row index that matches given 'ID' value in given 'ColID' column, or 65535 if not found or fails.

GetIDRowN(iID,ColID,ID,fromRow)

Returns row index that matches given 'ID' value in given 'ColID' column, starting at fromRow, or 65535 if not found or fails.

GetCell(iID,rIndex,cIndex)

Retrieves value for given cell. Indexes are zero-based

GetNRow(iID)

Returns number of rows in given ListBox.

DelRow(iID,Row)

Removes given row from ListBox

SelAll(iID,mode,St)

Selects ('St'!=0) or unselects ('St'==0) all the rows in given ListBox.

For 'mode'==0, this is all, for 'mode'=1, this is all the visible ones.

SelRow(iID,ColID,ID,St)

Selects ('St'!=0) or unselects ('St'==0) a row that matches given 'ID' value in given 'ColID' column.

GetSel(iID,ColID,RowID)

Returns true if first row with 'RowID' value into 'ColID' column is selected.

AddRowT(iID,pCol,idCol,Atr,rDef)

Adds a new row to a hierarquical ListBox, inserted as the last child for given parent. pCol is column index where to find parent ID, idCol is column index where to find row (item) ID (unused for now). A parent ID value of 0 is often considered root level.

ViewDepth(iID,Depth)

Sets visible depth for a tree (hierarquical) list.

SumRows(iID)

Sums cells on given ListBox, totalizing into rows with such attribute.

SetCRow(iID,ColID,ID)

Sets cursor to the first row that matches given 'ID' value in given 'ColID' column.

CB

AddRow(iID,rID,Txt)

Adds a new row entry, with given row ID and content.

SelByID(iID,rID)

Selects combo box row by its row ID (the first that matches). Returns True if succeeds, False if not found.

GetSel(iID)

Returns rID for currently selected row.

GetSelR(iID)

Returns row index for currently selected row.

GetNRow(iID)

Returns number of rows in given ComboBox.

GetRowID(iID,row)

Returns given row rID. 'row' is row index, or 65535 meaning currently selected one.

GetRowTxt(iID,row)

Returns given row content (row value has same meaning than in *GetRowID()*).

CalcDim(iID)

Forces ComboBox horizontal size calculation (call it after adding values, so scroll bar is calculated appropriately).

DateTime

Set(iID,mode,value)

Sets initial value. mode is currently "unix" for a timestamp value on a 32b or 64b unsigned integer, "sdate" for a 16bit days-since-1/1/1900 value, or “stime” for a short (16bit) time value (double seconds since 0:00:00). For an only-time item, value is extended to current day.

Get(iID,mode)

Gets current value. mode is "unix" or "sdate" and returned value is thus a 64b unsigned integer (but with 32bit or 16bit significant value).

SetNow(iID)

Sets initial value to right now.

Graph

Clear(iID,mode)

Empties data into given resource graph ('mode' is "data") or the whole graph ('mode' is "all").

InitR(iID,type,nRes)

(Re)creates Resource graphic framework. 'type' is "gantt" or "calendar", and nRes is the number of resources that will be managed.

ScaleR(iID,Start,Dur,Scale)

Fixes dimensions and scale for a Resource graphic. Start, Dur and Scale are optional (set to 0 if not used), and allow to fix the graphic dimensions and scale. Dur is duration in seconds for the whole graphic range, and Scale is the time in seconds that is measured between two time markers (usually, an hour multiple).

GetScaleR(iID)

Returns and array with dimension values. [“start”] gets starting datetime, [“end”] ending datetime, [“scale”] the scale used to draw time axis.

SetRes(iID,ResI,ResID,Color,Name,wStart,wEnd,nwd)

Sets resource with index 'ResI' with following data. Wstart is time when working time starts every working day (in double seconds, i.e. 1800 => 1:00:00), wEnd is when working time ends (use wStart=wEnd=0 if no working time distinction is desired), and nwd is non-working days bitmap (b0=Sunday, bit=1 => non-working week day) so 0 means all are working or no distinction at all.

GetResIdx(iID,ResID)

Returns resource index, or -1 if fails.

AddTask(iID,ResI,tID,Dur,Start,Kind,Mode,Name)

Adds a new task or sets an existing one, depending on 'Mode': 0=>always add, 1=>skip if present, 2=>overwrite if present. Task is identified by 'tID', associated to resource whose index is 'ResI', starts at 'Start' (currently, this is "unix" style datetime numeric value), with 'Dur' seconds duration, name 'Name', and kind 'Kind' (this is color index to represent it). 'Name' may also be

'-1' integer value, meaning to preserve previous name.

GetSelTask(iID)

Returns ID of last task selected by user, if any, or -1 if none.

GetNumTask(iID)

Returns number of tasks held by given graphic item.

GetTask(iID,idx)

Returns task 'idx' (referenced by index). Result is an array variable, with the following entries: "ptsk" (parent task ID), "res" (assigned resource), "dur" (duration in seconds), "start" (starting timestamp), "kind" (kind of task -color-), "txt" (task's name) and "tid" (task id).

SelTask(iID,tID)

Selects first task with such task ID, in given graphic item.

DelTaskID(iID,ResID,tID)

Deletes all tasks that match given task ID. ResID may be a resource ID (then only that resource's tasks are deleted) or 65535 meaning to delete them for any ResID.

DelTaskK(iID,ResID,Kind)

Deletes all tasks for given resource that match given Kind value. Setting ResID to 65535 deletes such tasks for all the resources.

CX

SetSel(iID,State)

Sets selection state for given CheckBox

GetSel(iID)

Gets selection state for given CheckBox

RB

SetSel(iID)

Sets selection state for given Radio, deactivating the rest on its group.

GetSel(gID)

Gets active Radio in given group

Image

Set(iID,img)

Sets image for given Image. 'img' may be text (then a file is loaded if has an extension or path component, or a local stock picture is searched for such name otherwise), a numeric variable (then a local stock picture is used), or a BB variable (then, a temporary file is created with such content, and then loaded into window item).

query

This object performs and manages queries

Variables

none

Handler Methods

Recv(qID)

Called when qID is finished (if not waiting for it with query.Wait(qID))

Methods:

Send(qID,Request,ResultRequest)

Sends a query (to local cache or remote server).

Save()

Sends a save request to remote server. Note that only one request can be managed at once (see InitSave(), DoneSave() and so on, below), so it is implicit.

Note that this may also block this thread while packets are sent (but does not wait for the wuery completion, see Wait() instead).

Wait(qID)

Waits for query qID to be completed. Returns 0 if finished successfully, 1 if finished with no results (empty), 2 if finished with errors, 3 if not yet finished, or -1 if simply fails. It may wait and fail due to timeout, or a query.Cancel() call from other part of the code.

Cancel(qID)

Cancels given query. If still unfinished and was waiting (query.Wait()), lock is released. Query is freed too.

Free(qID)

Frees given query. Does the same job as Cancel(qID) but does not send a cancel request to server.

GetNumItm(qID,objName)

Returns number of items (entries) that contains given query result for given object

GetRec(qID,objName,index)

Returns record at given index, for given object. Result is an array variable, each item identified by its field name, and with a type matching the record definition as close as possible. Note that array index names are always in lowercase.

For referenced values, you can use dots for the whole tree (like r["boss.name"])

InitSave(qID)

Initializes the 'save management' object (implicit: only one can be managed at once).

'qID' is the query ID you'll use once you send it to server (to Wait(qID) for completion, for an instance).

InitSaveL(qID,LockSN)

Initializes the 'save management' object (implicit: only one can be managed at once).

'qID' is the query ID you'll use once you send it to server (to Wait(qID) for completion, for an instance). LockSN is the lock serial number we want to be excluded from the operation (for an instance, if you lock a record as RO -read only-, the only way to write this record is to use here that lock's serial number, provided there's no other lock blocking it).

DoneSave()

Frees the 'save management' object.

AddSaveR(rDef)

Adds a record data to current 'save management' object. Returns internal ID assigned to this item, so you may reference it on subsequent saved items; or returns 0 if fails. 'rDef' is a text defining fields and values to assign to them.

AddSaveT(Txt)

Adds a text block of data to current 'save management' object. Returns internal ID assigned to this item, so you may reference it on subsequent saved items.

AddSaveB(Blk,iID)

Adds a binary block of data to current 'save management' object. Returns internal ID assigned to this item, so you may reference it on subsequent saved items. 'iID' is item ID (real) if block was just modified and we want to do so (instead of creating a new one).

GetSavedID(qID,localID)

Returns real ID assigned to a local ID after saving data successfully

SetLock(qID,Kind,RecordID,ObjName,Fields,TTL)

Locks a record 'RecordID' (or specific 'Fields' if given) for object 'ObjName'. 'Kind' may be “RW” for Read/Write lock (e.g. any access is allowed) or “RO” for Read Only (no modifications accesses are allowed). Every time somebody tries to access such record or fields, a notification arrives for the lock owner. Lock is removed automatically after TTL seconds (in case it is not deleted before).

DelLock(qID,RecordID,ObjName,Fields)

Removes a lock that matches given references.

DelLockByID(ObjName,SN)

Removes a lock on given object with serial number 'SN'.

GetLockRes(qID)

Returns result for a Lock operation (immediate or a later notification). Result is an array with the following entries: [“op”] with operation value (0= somebody read the locked data; 1=somebody tried to write the locked data; 2=lock is about to be deleted as TTL is close to expiration; 3=somebody tried to lock locked data; 4=data is already locked), [“res”] with success result (0=successful, 1=rejected), [“rid”] with affected record ID, [“ttl”] with remaining TimeToLive, [“crea”] with timestamp of lock creation, [“sn”] with lock serial number, [“ouid”] with “offender” user ID, [“object”] with object name, [“oid”] with object's ID, [“nf”] with # of affected fields (0=>whole record), and [“fn[]”] with every affected field name (starting at “fn[0]”).

system

This object gives access to system wide data and operations:

Variables

none

Methods

time()

Returns a 32bit unsigned value with seconds ellapsed since Epoch (19700101-00:00:00)

Dim(var)

Returns number of entries in given array variable. Also returns string length for a text string variable, and block size for a Binary Block variable.

GetType(var)

Returns type for such variable, as a text string. Values are: "bool", "number", "text", "block", "currency" and "array".

Free(var)

Releases (deletes) given variable.

TxtRight(txt,index)

Returns a text variable with the 'txt' content from (zero based) 'index' to the end. If txt is a Binary Block variable, it copies bytes as text, up to a control code (a non-character). You may use a negative index value to refer to characters from the end of the source string (e.g. index=-1 => last char).

TxtLeft(txt,index)

Returns a text variable with the 'txt' content from the beginning to (zero based, included) 'index'. If txt is a Binary Block variable, it copies bytes as text, from the first byte to index (included).

TxtMiddle(txt,index,length)

Returns a text variable with the 'txt' content from (zero based) 'index' up to 'length' chars. If txt is a Binary Block variable, it copies bytes as text, from offset index up to length or a control code (a non-character).

TxtPos(txt,start,needle)

Returns index of 'needle' into 'txt', starting to search at 'start' (zero-based), or empty if fails.

Log(Txt)

Logs given text to KinGUI debug log

PrintDate(date,fmt)

Returns text displaying given numeric 'date'. 'fmt' may be "short" or "full" (“long” is an alias of “full”, that also works).

GetMoney(txt)

Returns an economic value from given text, if possible.

GetMoneyS(var,mode)

Returns standard text representation for given currency value. 'mode' may have b4 set (use thousand separators), b6 set (return as many decimal digits as available instead of default), or b7 set (normalized for DB: no separators and uses dot '.' as decimal point).

GetNum(txt,dp)

Gets a numeric value from given text, if possible. It can return an integer or a floating point value. 'dp' specifies decimal places for fixed point integer if not zero.

GetNumS(num,dp)

Returns text representation for given number, using it as fixed point integer with 'dp' decimal digits. It also shows it with thousands separators. 'num' may also be a text string with a bank account value, that is returned with some separators for better reading (depending on the country code).

GetInt(var)

Gets an integer numeric value from given variable, if possible. It can return a signed or unsigned integer.

GetTime(txt,mode)

Parses 'txt' and tries to convert to given time numeric representation. Allowed modes are 2 (16bit, double seconds), or 5 (32bit, up to 0.1 msec). It also accepts 'txt' to be a numeric variable, expecting a unix-style timestamp and then returning the time part, in given mode.

GetTimeS(var,nmode,mode)

Returns text representation for given numeric time. Allowed 'mode' values are 1 (HH:MM), 2 (HH:MM:SS), 4 (HHMMSS) or 5 (HH:MM:SS.DDDD). 'nmode' indicates how to interpret number (2=16bit, 5=32bit). Also two other modes are allowed, for durations (6=H:MM:SS and 7=H:MM).

GetDate(txt,nmode,mode)

Parses 'txt' and tries to convert to given time numeric representation. Allowed modes are 2 (16bit, short date), or 5 (32bit, unix standard date). It also accepts 'txt' to be a numeric variable, expecting a unix-style timestamp and then returning the date part, in given mode. 'nmode' indicates how to interpret number (2=16bit, 5=32bit).

GetDateS(var,nmode,mode)

Returns text representation for given numeric date. Allowed 'mode' values are 0 (default), 1 (MM/DD/YYYY), 2 (DD/MM/YYYY), 3 (YYYY/MM/DD), 4 (YYYYMMDD), set bit3 to get 2-digit year values. 'nmode' indicates how to interpret number (2=16bit, 5=32bit).

SetBlockG(var,g)

Sets 'var' block granularity to 'g' bytes (size of its items).

GetText(var,ofs)

Tries to convert 'var' block content starting at 'ofs' to a text string.

LoadBB(fn)

Creates a BinaryBlock variable and fills it with the content of file 'fn'.

SaveBB(var,fn)

Creates a file called 'fn' with the content of given 'var', that must be of type BinaryBlock.

Type(var,type)

Returns 'var' as requested 'type', if possible. Valid 'type' values are currently "bool", "integer", "float", "text", and "currency". Note that this is not a conversion, but a forced casting, and is intended for values received from out of a script, that are captured as a type and are really representing another type (for an instance, to force a floating point stored into an integer variable -that of course, doesn't show the expected value- to be turned into a floating point variable, with a binary copy of its content).

Append(var1,pos1,var2)

Returns concatenation of the content of var2 to var1 (at index pos1). This works for some combinations, like appending a text string (var2) into a Binary Block at some offset. In such example, data into var1[pos1] is overwritten and extended if necessary. Items are also converted (so if var1 has an item size of 2 (16bits), every item gets the value of one char from var2 (in that example).

LoadImage(imagename,width,height)

Returns an ID to the loaded image (0=failed). imagename may contain a path (in such case, image is loaded from disk), or be a plain name (then, it is first requested to the internal images stock). It may also be a numeric value, and then a stock image is returned with such ID (not related to the values returned by this function!) and width and height are ignored. width and height may be 0, meaning that image is loaded with no rescaling. Image information is released by calling FreeImage(), or when the script is finished.

GetImageID(imagename,width,height)

Returns an ID to the cached image (0=failed). width and height may be zero to get unscaled image (as cached).

FreeImage(iID)

Frees data for image with given ID.

GetHexT(num,digits)

Returns text with hexadecimal representation for given number, with given number of digits (usually 2, 4, etc.)

DelArrayItem(var,index)

Removes an entry from an array. 'index' may be a numeric or named index, and 'var' must be of type Array.

InsArrayItem(var,index,item)

Inserts an entry into an array. 'index' must be a numeric index (text indexes have no order), and 'var' must be of type Array. 'item' may be any type of variable, and will be inserted with given index. If index is bigger than any existing one, entry will just be appended and index set to highest plus one. If index entry does not exist (but has other entries with greater indexes), entry will just be created with given index (and no other will be changed). If there's already an entry with given index, it will be pushed up (index incremented by one) and so will be any other entries with a bigger index (gaps will be kept, all entries with bigger index will have their indexes incremented by one, no matter gaps are found between them or not).

GetChar(var)

Returns given numeric variable value as a one-char string. Useful to get a string char value, for an instance, as system.GetChar(s[3]) will return the same as system.TxtMiddle(s,3,1). Remember that s[3] evaluates as a number!.

CfgSave()

Saves current user configuration to disk.

CfgGet(name)

Retrieves given variable from user configuration. Variable type depends on stored value(s), so it can be text, numeric, etc. and scalar or an array.

CfgSet(name,val)

Sets given variable in user configuration to given value(s). Variable type will be according to given value: it can be text, numeric, etc. and scalar or an array, but fails if there's mixed data into the array (only supports uniform array, that is: same type entries).

print

This object manages printing. It does only support one print job at any time.

Variables

none

Handler Methods

prepare()

Called to fill given job with desired content (page). Here you may call *DefTxtFont()* to create any font faces you may need, you may *SetMargins()* for the pages, you may also *AddCommon()* with headers, footers or any other common structures, then call *StartPage()* to start every page (then use *Write()* and *DrawImg()* to draw content), finally *EndPage()* and repeat this later step for every page. You can forget about printing itself, as this is decided by user on the printing popup (so it can be printed, previewed or exported).

Methods

Init(jobName,landscape,color)

Open printing window, and prepares a new printing job. Returns wID of printing window (0 if fails). jobName is your desired job name, landscape and color are booleans, when false mean portrait and B&W.

InitE(jobName,landscape,color,fromN,fromA,toN,toA,Subject,Body)

Open printing window, and prepares a new printing job that can be sent by e-mail. Returns wID of printing window (0 if fails). jobName is your desired job name, landscape and color are booleans, when false mean portrait and B&W. Also supplied are e-Mail parameters: fromN (sender name), fromA (sender e-mail address), toN (recipient name), toA (recipient address), Subject and Body (e-mail body text).

DefTxtFont(FontName,FontSize,FontAttr)

Defines a new font for printing job, and returns its index (you will use it to set current font in a statement like *Write("$tf("+FontIndex+")")*). FontSize is char height in 100x points (e.g. 1200 is 12.00pt), and FontAttr may be "Bold", "Italic", "Normal" or "Oblique". FontName may be "" to use default font.

SetMargins(l,r,t,b)

Sets page margins (l=left, r=right, t=top, b=bottom, in standard units i.e. tenths of millimeter) for given job. This should be called before writing anything to it, and has effect only on new pages.

AddCommon(txt)

Adds given commands and text as a common page element. This sequence is executed at every new page. You may call this several times to add several common content.

StartPage()

Starts a new page in the document. Returns page number (starting at one), or smaller in case of error. It also notifies printing window about a new page is starting to be prepared.

Write(txt)

Writes some content into current job. See Annex C – Printing language.

DrawImg(x,y,w,h,iID)

Draws given image into the page (image should be), at given position and with given size (all in standard units, that is, tenths of millimeter). iID is image ID, as returned by system.LoadImage().

EndPage()

Finishes with a page.

utool

This object is a placeholder for user tools, implemented as dynamic binary functions, that are linked at runtime when application starts. Here follows an example on how to implement a utool library (note that we have a private functions section before the exported ones, and that the example also illustrates the use of other useful tools, like local cache databases management, text management, etc.; this library is used also in examples at Annex E):

```
/***************************************************************************
 *   Copyright (C) 2010 by David Lopez Vinacua                             *
 *   david@bajavista.com                                                   *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 ***************************************************************************/

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  systools.c (module -common to many apps- with useful tools)  *
 *  Code and Data for Kin GUI - Customized Tools                 *
 *  This is common to many applications                          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *
 * Created:
 *  080610 David Lopez
 * Modified:
 *
 */

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>

#include "systools.h"

#include "../../dalib/daltypes.h"
#include "../../dalib/daltool.h"
#include "../../dalib/dalpo.h"
#include "../../dalib/dalmem.h"

#include "../../guilib/king_api.h"
#include "../../guilib/king_cache.h"
#include "../../guilib/king_us.h"

// NOTE that this is included into a module (system) but does export only internal tools (utool_XXX
functions)

#ifdef _WIN32_B
# include <windows.h>
# define __export__ __declspec(dllexport)
#elif defined _LINUX
# define __export__
#endif // _WIN32_B

// Declare included tools
__export__ const char system_utool[] =
{"sysLoadFP,sysSaveFP,sysGetNumFP,sysGetFP,sysSetFP,sysLoadFL,sysSaveFL,sysGenFL,sysGetNumFL,sysGetFL,
sysReloadLocalCache,sysSetDBFL"};

/*
 *  ========= Internal data structures =========
 */

typedef struct          // Local file list (files that we want to keep track onto remote DB)
    {
    Word        ID;         // Last known ID for this entry into remote DB (that may change after a
new migration or DB regeneration)
    Word        Fil2;
    Byte        Kind;       // Kind of file
    Byte        Arch;       // Architecture
    Byte        Flg;        // b0: 1=>detected file change (respect to remote DB)
    Byte        Fil1;
    int         Desc;       // Description
    int         Path;       // File path
    int         fn;         // FileName
    u32         chk;        // Last known checksum (local)
    u32         fSz;        // Local file size
    u32         fMod;       // Local file modification timestamp
    u32         bDta;       // DB _dta ID value
    } t_systFDB;

typedef struct
    {
    Word        NumR;       // # of items into r[]
    Word        MaxR;       // Dimension of r[]
    u32         Fil1;
    t_systFDB   r[];
    } t_sysFDBL;

typedef struct          // Local file patterns (to build local file list)
    {
    char        path[128];  // Base path. It may also be a specific file (complete path)
    char        ext[64];    // Extensions list (comma separated, no dots)
    Byte        kind;
    Byte        arch;
    } t_sysFP;

typedef struct          // Local file patterns list. First occurrences prevail (a file that matches
several patterns, is added only with the very first it does)
    {
    Word        NumR;       // # of items into r[]
    Word        MaxR;       // Dimension of r[]
    u32         Fil1;
    t_sysFP     r[];
    }  t_sysFPL;

typedef struct          // File List/Pattern file header
    {
    char              mg[8];
    Word              Ver;
    Word              NumR;
    u32               Fil2;
    } t_FX_fHdr;

static t_sysFDBL *DBL   = NULL; // Local File list
static char      *DBLt  = NULL; // Text storage
static int        DBLm  = 0;    // Size of DBLt
static int        DBLu  = 0;    // Used size into DBLt
static t_sysFPL  *FPL   = NULL; // File patterns list


/*
 *  ========= Generic (internal) tools =========
 */

t_systFDB *GetFDB(Word ID)
// Returns record by ID, or NULL if not found
{
  Word              i;

  if (!DBL || !DBL->NumR)
     return NULL;
  i=0;
  while (i<DBL->NumR)
    {
    if (DBL->r[i].ID==ID)
       return &DBL->r[i];
    i++;
    }
  return NULL;
}

static t_systFDB *AddFDB(Word ID, Byte Kind, Byte Arch, const char *Path, const char *FN, const char
*Desc)
// Adds new entry to DBL (calculates the rest of fields)
{
  #define NumI 64
  t_systFDB        *r;
  void             *p;

  //r=GetFDB(ID);
  r=NULL;
  if (!r)
     {  // New
     if (!DBL)
        { // New list
        DBL=(t_sysFDBL *)malloc(sizeof(t_sysFDBL)+sizeof(t_systFDB)*NumI);
        if (!DBL)
           return NULL;
        DBL->MaxR=NumI;
        DBL->NumR=0;
        }
     if (DBL->NumR>=DBL->MaxR)
        { // Grow list
        p=realloc(DBL,sizeof(t_sysFDBL)+sizeof(t_systFDB)*(DBL->MaxR+NumI));
        if (!p)
           return NULL;
        DBL=(t_sysFDBL *)p;
        DBL->MaxR+=NumI;
        }
     r=&DBL->r[DBL->NumR];
     DBL->NumR++;
     bzero(r,sizeof(t_systFDB));
     }
  // Update *r
  r->ID=ID;
  r->Kind=Kind;
  r->Arch=Arch;
  r->Flg=0;
  r->Desc=txtb_AddText2Buf(&DBLt,&DBLm,&DBLu,Desc,1);
  r->fn=txtb_AddText2Buf(&DBLt,&DBLm,&DBLu,FN,1);
  r->Path=txtb_AddText2Buf(&DBLt,&DBLm,&DBLu,Path,1);
  return r;
  #undef NumI
}

static t_systFDB *FindFDB(const char *FN, Byte Kind, Byte Arch)
// Returns record matching file name, kind and arch, or NULL if not found
{
  Word              i;
  int               fni;

  if (!DBL || !DBL->NumR)
     return NULL;
  fni=txtb_FindText2Buf(DBLt,DBLu,FN,0,NULL);
  if (fni<0)
     return NULL;
  i=0;
  while (i<DBL->NumR)
    {
    if ( (DBL->r[i].Arch==Arch) && (DBL->r[i].fn==fni) )
       {
       if ( (DBL->r[i].Kind==Kind) || ((DBL->r[i].Kind==12) && (Kind==11)) )  // Indep. app prevails
over simple app
          return &DBL->r[i];
       }
    i++;
    }
  return NULL;
}

#define k_FP_Magic "KDBLF101"
#define k_FP_Ver 0x0001

static short LoadFL(void)
// Loads file list DB
{
  FILE               *f;
  t_FX_fHdr           h;
  char                fn[192];
  int                 sz;
  short               res;

  res=-1;
  snprintf(fn,sizeof(fn),"%s/%s/_FilesList.dta",gCfg.Path,gCfg.Dta);
  f=fopen(fn,"rb");
  if (f)
     {
     fseek(f,0,SEEK_END);
     sz=ftell(f);
     fseek(f,0,SEEK_SET);
     if ( (fread(&h,sizeof(h),1,f)==1) && (memcmp(&h.mg[0],k_FP_Magic,sizeof(h.mg))==0) &&
(h.Ver==k_FP_Ver) )
        {
        sz-=sizeof(h);
        if (sz>=sizeof(t_systFDB)*h.NumR)
           {
           if (DBL)
              free(DBL);
           if (DBLt)
              free(DBLt);
           DBLt=NULL;
           DBLu=0;
           DBLm=0;
           DBL=(t_sysFDBL *)malloc(sizeof(t_sysFDBL)+sizeof(t_systFDB)*h.NumR);
           if (DBL)
              {
              DBL->MaxR=h.NumR;
              DBL->NumR=DBL->MaxR;
              res=(fread(DBL+1,sizeof(t_systFDB),h.NumR,f)!=h.NumR);
              sz-=sizeof(t_systFDB)*h.NumR;
              if (!res && (sz>0))
                 {
                 DBLm=(sz+1023) & ~1023;
                 DBLt=(char *)malloc(DBLm);
                 if (DBLt)
                    {
                    res=(fread(DBLt,1,sz,f)!=sz);
                    if (!res)
                       DBLu=sz;
                    }
                 }
              }
           }
        }
     fclose(f);
     }
  return res;
}

static short SaveFL(void)
// Saves file list DB
{
  FILE               *f;
  t_FX_fHdr           h;
  char                fn[192];
  short               res;

  if (!DBL || !DBL->NumR)
     return -1;
  res=-2;
  snprintf(fn,sizeof(fn),"%s/%s/_FilesList.dta",gCfg.Path,gCfg.Dta);
  f=fopen(fn,"wb");
  if (f)
     {
     bzero(&h,sizeof(h));
     memcpy(&h.mg[0],k_FP_Magic,sizeof(h.mg));
     h.Ver=k_FP_Ver;
     h.NumR=DBL->NumR;
     if (fwrite(&h,sizeof(h),1,f)==1)
        {
        res=(fwrite(DBL+1,sizeof(t_systFDB),DBL->NumR,f)!=DBL->NumR);
        if (!res)
           res=(fwrite(DBLt,1,DBLu,f)!=DBLu);
        }
     fclose(f);
     }
  return res;
}
#undef k_FP_Magic
#undef k_FP_Ver


static short Generate1F(const char *fn, const char *Path, const char *FN, Byte Kind, Byte Arch)
// Adds given file to local file list (and tries to guess some other fields against _Files cache)
{
  t_systFDB          *r;
  char               *c1;
  void               *p;
  FILE               *f;
  int                 ri;
  u32                 chk;
  Word                id;
  short               cID,rID,rFN,rK,rA,rC;
  struct stat        st;

  if (!fn)
     return -1;
  // Get id for this file, from cache
  id=0;
  chk=0;
  cID=kgc_GetCacheID("_Files");
  if (cID>0)
     {
     rID=kgc_GetCacheFix(cID,"ID");
     rFN=kgc_GetCacheFix(cID,"_fn");
     rK=kgc_GetCacheFix(cID,"_kind");
     rA=kgc_GetCacheFix(cID,"_arch");
     rC=kgc_GetCacheFix(cID,"_chk");
     if (!( (rID<0) || (rFN<0) || (rK<0) || (rA<0) || (rC<0) ||
            (kgc_GetCacheFtype(cID,rID)<k_kgcFTnum8) || (kgc_GetCacheFtype(cID,rK)<k_kgcFTnum8) ||
(kgc_GetCacheFtype(cID,rA)<k_kgcFTnum8) ||
            (kgc_GetCacheFtype(cID,rC)<k_kgcFTnum8) || (kgc_GetCacheFtype(cID,rFN)!=k_kgcFTtxt) ))
        {
        ri=kgc_SearchCacheF(cID,rK,Kind);
        while (ri>=0)
          {
          p=kgc_GetCacheFv(cID,rA,ri);
          if (p && (*((Byte *)p)==Arch))
             {
             c1=kgc_GetCacheFt(cID,rFN,ri);
             if (strcmp(c1,FN)==0)
                {
                p=kgc_GetCacheFv(cID,rID,ri);
                if (p)
                   id=*((Word *)p);
                p=kgc_GetCacheFv(cID,rC,ri);
                if (p)
                   chk=*((u32 *)p);
                }
             }
          ri=kgc_SearchCacheFn(cID);
          }
        }
     }
  r=FindFDB(FN,Kind,Arch);
  if (!r)
     // Create new entry
     r=AddFDB(id,Kind,Arch,Path,FN,"");
  if (r)
     {
     if (!( (r->Kind==12) && (Kind==11) ))
        r->ID=id; // Update remote id, except for overlapping app-over-indep.app
     // Load file and test checksum and other data
     f=fopen(fn,"rb");
     if (f)
        {
        fseek(f,0,SEEK_END);
        r->fSz=ftell(f);
        fseek(f,0,SEEK_SET);
        if (r->fSz>0)
           {
           p=malloc((r->fSz+3) & ~3);
           fread(p,r->fSz,1,f);
           if (r->fSz & 3)
              bzero(p+r->fSz,(4-(r->fSz & 3)));
           }
        else
           p=NULL;
        if (!fstat(fileno(f),&st))
           r->fMod=st.st_mtime+timezone;  // we want it UTC
        fclose(f);
        if (p)
           {
           r->chk=ChkSum32_fast(p,(r->fSz+3)/4,0);
           if (r->chk!=chk)
              r->Flg|=b0;
           else
              r->Flg&= ~b0;
           }
        }
     }
  return 0;
```

```c
}

static short Generate1FL(t_sysFP *p)
{
  DIR               *dir;
  struct dirent     *de;
  char              *c,*d;
  char               fn[192];
  char               ext[128]; // Extensions list, null terminated each, double null terminated list
  short              sl,el,l;
  char               ok;
  struct stat        st;

  if (!p->path)
     return -1;
  if (!stat(p->path,&st) && (S_ISREG(st.st_mode)) )
     {  // It is a specific file. We will set fn with path and ext with file name components
     c=strrchr(p->path,'/');
     if (c)
        {
        sl=(c-p->path);
        if (sl+1>sizeof(fn))
           return -2;
        if (sl>0)
           memcpy(&fn[0],p->path,sl);
        fn[sl]='\0';
        snprintf(ext,sizeof(ext),"%s",c+1);
        }
     else
        {
        fn[0]='\0';
        snprintf(ext,sizeof(ext),"%s",p->path);
        }
     Generate1F(p->path,fn,ext,p->kind,p->arch);
     return 0;
     }
  dir=opendir(p->path);
  if (dir)
     {
     sl=snprintf(fn,sizeof(fn)-5,"%s",p->path);
     if (fn[sl-1]!='/')
        {
        fn[sl]='/';
        sl++;
        }
     d=&ext[0];
     if (p->ext)
        { // Parse extensions
        c=&p->ext[0];
        el=sizeof(ext)-2;
        while (*c && (el>0))
          {
          if ((*c)==',')
             *d='\0';
          else
             *d=*c;
          c++;
          d++;
          el--;
          }
        }
     *d='\0';
     d++;
     *d='\0';
     while ( (de=readdir(dir)) )
       {
       if (de->d_name[0]=='.')
          continue;
       // Test file for extension match
       if (ext[0])
          { // Test extensions
          ok=0;
          c=&ext[0];
          el=strlen(de->d_name);
          while (*c)
            {
            l=strlen(c);
            if ( (el>l) && (de->d_name[el-l-1]=='.') && (memcmp(&de->d_name[el-l],c,l)==0) )
               {
                ok=1;
                break;
                }
             c+=1+l;
             }
          }
       else
          ok=1;
       if (ok)
          {
          snprintf(&fn[sl],sizeof(fn)-sl,"%s",de->d_name);
          Generate1F(fn,p->path,de->d_name,p->kind,p->arch);
          }
       }
     closedir(dir);
     }
  return 0;
}

static short GenerateFL(t_sysFPL *fp)
// Creates the local file list, based on given File patterns
{
  Word              i;

  if (!fp)
     return -1;
  i=0;
  while (i<fp->NumR)
    {
    if (fp->r[i].kind)
       Generate1FL(&fp->r[i]);
    i++;
    }
  return 0;
}

#define k_FP_Magic "KDBLF100"
#define k_FP_Ver 0x0001

static short LoadFP(void)
// Loads file patterns DB
{
  FILE               *f;
  t_FX_fHdr           h;
  char                fn[192];
  int                 sz;
  short               res,ni;

  res=-1;
  snprintf(fn,sizeof(fn),"%s/%s/_FilesPtrn.dta",gCfg.Path,gCfg.Dta);
  f=fopen(fn,"rb");
  if (f)
     {
     fseek(f,0,SEEK_END);
     sz=ftell(f);
     fseek(f,0,SEEK_SET);
     if ( (fread(&h,sizeof(h),1,f)==1) && (memcmp(&h.mg[0],k_FP_Magic,sizeof(h.mg))==0) &&
(h.Ver==k_FP_Ver) )
        {
        sz-=sizeof(h);
        //if (!(sz%sizeof(t_sysFP)))
        if (sz>=sizeof(t_sysFP)*h.NumR)
           {
           if (FPL)
              free(FPL);
           FPL=(t_sysFPL *)malloc(sizeof(t_sysFPL)+sz);
           ni=sz/sizeof(t_sysFP);
           if (FPL)
              {
              FPL->MaxR=ni;
              FPL->NumR=FPL->MaxR;
              res=(fread(FPL+1,sizeof(t_sysFP),ni,f)!=ni);
              }
           }
        }
     fclose(f);
     }
  return res;
}
```

```c
static short SaveFP(void)
// Saves file patterns DB
{
  FILE               *f;
  t_FX_fHdr           h;
  char                fn[192];
  short               res;

  if (!FPL || !FPL->NumR)
     return -1;
  res=-2;
  snprintf(fn,sizeof(fn),"%s/%s/_FilesPtrn.dta",gCfg.Path,gCfg.Dta);
  f=fopen(fn,"wb");
  if (f)
     {
     bzero(&h,sizeof(h));
     memcpy(&h.mg[0],k_FP_Magic,sizeof(h.mg));
     h.Ver=k_FP_Ver;
     h.NumR=FPL->NumR;
     if (fwrite(&h,sizeof(h),1,f)==1)
        res=(fwrite(FPL+1,sizeof(t_sysFP),FPL->NumR,f)!=FPL->NumR);
     fclose(f);
     }
  return res;
}
#undef k_FP_Magic
#undef k_FP_Ver


static short AddFP(Word idx, Byte kind, Byte arch, const char *path, const char *ext)
// Sets or appends a File Pattern entry
{
  #define NumI 32
  void               *p;

  if (!FPL)
     {
     FPL=(t_sysFPL *)malloc(sizeof(t_sysFPL)+NumI*sizeof(t_sysFP));
     if (!FPL)
        return -32;
     FPL->MaxR=NumI;
     FPL->NumR=0;
     }
  if (idx>FPL->NumR)
     idx=FPL->NumR;
  if ((idx==FPL->NumR) && (FPL->NumR>=FPL->MaxR))
     {  // Grow list
     p=realloc(FPL,sizeof(t_sysFPL)+(NumI+FPL->MaxR)*sizeof(t_sysFP));
     if (!p)
        return -32;
     FPL=(t_sysFPL *)p;
     FPL->MaxR+=NumI;
     }
  FPL->r[idx].kind=kind;
  FPL->r[idx].arch=arch;
  snprintf(FPL->r[idx].path,sizeof(FPL->r[idx].path),"%s",path);
  snprintf(FPL->r[idx].ext,sizeof(FPL->r[idx].ext),"%s",ext);
  if (idx==FPL->NumR)
     FPL->NumR++;
  return SaveFP();
  #undef NumI
}

static void utool_sys_AddAEt(t_kgwThread *t, short v0, Word FunID, const char *en, const char *val)
// Adds array entry with text
{
  int                 ix,l;

  l=strlen(val)+1;
  kgs_ClrVB(&s_v[v0]);
  kgs_ChkVB(&s_v[v0],l);
  s_v[v0].v->Type=k_kgsTypeStr;
  s_v[v0].v->bSz+=(l+7) & ~7;
  memcpy(s_v[v0].v+1,val,l);
  ix=txtb_AddText2Buf(&SGON.Txt,&SGON.tSz,&SGON.uSz,en,1);
  kgs_SetArrayEntry(&t->stk[t->nStk-1].Loc,FunID,b0,ix,s_v[v0].v);
}
```

```
static void utool_sys_AddAEn(t_kgwThread *t, short v0, Word FunID, const char *en, u64 val)
// Adds array entry with text
{
  int                 ix;

  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeNum64u;
  s_v[v0].v->bSz+=sizeof(u64);
  *((u64 *)(s_v[v0].v+1))=val;
  ix=txtb_AddText2Buf(&SGON.Txt,&SGON.tSz,&SGON.uSz,en,1);
  kgs_SetArrayEntry(&t->stk[t->nStk-1].Loc,FunID,b0,ix,s_v[v0].v);
}

/*  ===========================================
 *
 *  ========= Specific (public) tools =========
 *
 *  ===========================================
 */

/*
 *  'sysLoadFP()' tool
 *  Loads File Pattern DB from disk
 */
__export__ const char utool_decl_sysLoadFP[] = {"()"};
__export__ i32 utool_sysLoadFP(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=LoadFP();
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysSaveFP()' tool
 *  Saves File Pattern DB to disk
 */
__export__ const char utool_decl_sysSaveFP[] = {"()"};
__export__ i32 utool_sysSaveFP(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=SaveFP();
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysGetNumFP()' tool
 *  Returns how many File Patterns we have
 */
__export__ const char utool_decl_sysGetNumFP[] = {"()"};
__export__ i32 utool_sysGetNumFP(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
```

```c
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeNum64u;
  if (FPL)
     *((u64 *)(s_v[v0].v+1))=FPL->NumR;
  else
     *((u64 *)(s_v[v0].v+1))=0;
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysGetFP(idx)' tool
 *  Returns given File Pattern entry, as an array. Entries are: "path", "ext", "kind", "arch"
 */
__export__ const char utool_decl_sysGetFP[] = {"(idx)"};
__export__ i32 utool_sysGetFP(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  t_kgsVar           *vv;
  short               v0;
  Word                i;
  char                ok;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  ok=0;
  if ( FPL && FPL->NumR && (nPar==1) )
     {
     vv=kgs_GetVar(ws,t,*ParID,NULL);
     if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0]))
        {
        i=*((u64 *)(s_v[v0].v+1));
        if (i<FPL->NumR)
           {
           utool_sys_AddAEt(t,v0,FunID,"path",FPL->r[i].path);
           utool_sys_AddAEt(t,v0,FunID,"ext",FPL->r[i].ext);
           utool_sys_AddAEn(t,v0,FunID,"kind",FPL->r[i].kind);
           utool_sys_AddAEn(t,v0,FunID,"arch",FPL->r[i].arch);
           ok=1;
           }
        }
     }
  if (!ok)
     {
     kgs_ClrVB(&s_v[v0]);
     kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
     }
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysSetFP(idx,kind,arch,path,ext)' tool
 *  Sets given File Pattern entry. If idx is out of range, entry is appended.
 */
__export__ const char utool_decl_sysSetFP[] = {"(idx,kind,arch,path,ext)"};
__export__ i32 utool_sysSetFP(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 5
  t_kgsVar           *vv;
  short               v0;
  Word                i;
  char                ok;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  s_v[v0].v->bSz+=sizeof(u64);
  ok=0;
  if (nPar==5)
     {
     vv=kgs_GetVar(ws,t,*ParID,NULL);
     if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0+1]))
        {
        i=*((u64 *)(s_v[v0+1].v+1));  // idx
        vv=kgs_GetVar(ws,t,*(ParID+1),NULL);
        if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0+2])) // kind
           {
           vv=kgs_GetVar(ws,t,*(ParID+2),NULL);
           if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0+3])) // arch
              {
              vv=kgs_GetVar(ws,t,*(ParID+3),NULL);
              if (vv && !kgs_ConvertVar2Str(vv,&s_v[v0+4])) // path
                 {
                 vv=kgs_GetVar(ws,t,*(ParID+4),NULL);
                 if (vv && !kgs_ConvertVar2Str(vv,&s_v[v0+5])) // ext
                    {
                    AddFP(i,*((u64 *)(s_v[v0+2].v+1)),*((u64 *)(s_v[v0+3].v+1)),(char *)
(s_v[v0+4].v+1),(char *)(s_v[v0+5].v+1));
                    ok=1;
                    }
                 }
              }
           }
        }
     }
  *((char *)(s_v[v0].v+1))=ok;
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}


/*
 *  'sysLoadFL()' tool
 *  Loads File List DB from disk
 */
__export__ const char utool_decl_sysLoadFL[] = {"()"};
__export__ i32 utool_sysLoadFL(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=LoadFL();
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysSaveFL()' tool
 *  Saves File List DB to disk
 */
__export__ const char utool_decl_sysSaveFL[] = {"()"};
__export__ i32 utool_sysSaveFL(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=SaveFL();
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysGenFL()' tool
 *  Updates File List from existing one plus the result of File Patterns
 */
__export__ const char utool_decl_sysGenFL[] = {"()"};
__export__ i32 utool_sysGenFL(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=GenerateFL(FPL);
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysGetNumFL()' tool
 *  Returns how many File items we have
 */
__export__ const char utool_decl_sysGetNumFL[] = {"()"};
__export__ i32 utool_sysGetNumFL(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  short               v0;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeNum64u;
  if (DBL)
     *((u64 *)(s_v[v0].v+1))=DBL->NumR;
  else
     *((u64 *)(s_v[v0].v+1))=0;
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysGetFL(idx)' tool
 *  Returns given File list entry, as an array.
 *  Entries are: "id", "kind", "arch", "flg", "desc", "path", "fn", "chk", "fsz", "fmod", "cchk",
"csz", "cmod", "dta"
 */
__export__ const char utool_decl_sysGetFL[] = {"(idx)"};
__export__ i32 utool_sysGetFL(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 1
  t_kgsVar           *vv;
  void               *p;
  short               v0;
  Word                i;
  char                ok;
  short               cID,rID,rCK,rSZ,rMD;
  int                 ri;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  ok=0;
  if ( DBL && DBL->NumR && (nPar==1) )
     {
     vv=kgs_GetVar(ws,t,*ParID,NULL);
     if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0]))
        {
        i=*((u64 *)(s_v[v0].v+1));
        if (i<DBL->NumR)
           {
           // Get cache entry for given ID, to append cached values first
           cID=kgc_GetCacheID("_Files");
           if (cID>0)
              {
              rID=kgc_GetCacheFix(cID,"ID");
              rCK=kgc_GetCacheFix(cID,"_chk");
              rSZ=kgc_GetCacheFix(cID,"_sz");
              rMD=kgc_GetCacheFix(cID,"_fts");
              if (!( (rID<0) || (rCK<0) || (rSZ<0) || (rMD<0) ||
                      (kgc_GetCacheFtype(cID,rID)<k_kgcFTnum8) ||
(kgc_GetCacheFtype(cID,rCK)<k_kgcFTnum8) || (kgc_GetCacheFtype(cID,rSZ)<k_kgcFTnum8) ||
                      (kgc_GetCacheFtype(cID,rMD)<k_kgcFTnum8) ))
                 {
                 ri=kgc_SearchCacheF(cID,rID,DBL->r[i].ID);
                 if (ri>=0)
                    {
                    p=kgc_GetCacheFv(cID,rCK,ri);
                    if (p)
                       utool_sys_AddAEn(t,v0,FunID,"cchk",*((u32 *)p));
                    p=kgc_GetCacheFv(cID,rSZ,ri);
                    if (p)
                       utool_sys_AddAEn(t,v0,FunID,"csz",*((u32 *)p));
                    p=kgc_GetCacheFv(cID,rMD,ri);
                    if (p)
                       utool_sys_AddAEn(t,v0,FunID,"cmod",*((u32 *)p));
                    }
                 }
              }
           // Now include local values
           utool_sys_AddAEn(t,v0,FunID,"id",DBL->r[i].ID);
           utool_sys_AddAEn(t,v0,FunID,"kind",DBL->r[i].Kind);
           utool_sys_AddAEn(t,v0,FunID,"arch",DBL->r[i].Arch);
           utool_sys_AddAEn(t,v0,FunID,"flg",DBL->r[i].Flg);
           utool_sys_AddAEt(t,v0,FunID,"desc",txtb_GetText2Buf(DBLt,DBLu,DBL->r[i].Desc));
           utool_sys_AddAEt(t,v0,FunID,"path",txtb_GetText2Buf(DBLt,DBLu,DBL->r[i].Path));
           utool_sys_AddAEt(t,v0,FunID,"fn",txtb_GetText2Buf(DBLt,DBLu,DBL->r[i].fn));
           utool_sys_AddAEn(t,v0,FunID,"chk",DBL->r[i].chk);
           utool_sys_AddAEn(t,v0,FunID,"fsz",DBL->r[i].fSz);
           utool_sys_AddAEn(t,v0,FunID,"fmod",DBL->r[i].fMod);
           utool_sys_AddAEn(t,v0,FunID,"dta",DBL->r[i].bDta);
           ok=1;
           }
        }
     }
  if (!ok)
     {
     kgs_ClrVB(&s_v[v0]);
     kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
     }
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}

/*
 *  'sysReloadLocalCache(name)' tool
 *  Schedules for a local cache reload
 */
__export__ const char utool_decl_sysReloadLocalCache[] = {"(name)"};
__export__ i32 utool_sysReloadLocalCache(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word
*ParID)
/* Function */
{
  #define             NumV 2
  t_kgWin            *win;
  t_kgsVar           *vv;
  char               *c;
  short               v0,sl;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=0;
  win=kgw_GetWinByName("Services");
  if ( win && (nPar==1) )
     {
     vv=kgs_GetVar(ws,t,*ParID,NULL);
     if (vv && !kgs_ConvertVar2Str(vv,&s_v[v0+1]))
        {
        c=(char *)(s_v[v0+1].v+1);
        sl=strlen(c);
        *((Byte *)(s_v[v0].v+1))=kgw_SendWinMsg(ws->wID,win->wID,k_kgmmUsr1+1,sl+1,c);
        }
     }
  s_v[v0].v->bSz+=sizeof(u64);
  kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
  t->stk[t->nStk-1].Flg|=b0;
  kgs_FreeVB(v0,NumV);
  #undef NumV
  return 0;
}


/*
 *  'sysSetDBFL(dbID,desc,dtaID)' tool
 *  Sets desc and dtaID for given File List entry, identified by the dbID. desc="" or dtaID=0 avoids
modifying them
 */
__export__ const char utool_decl_sysSetDBFL[] = {"(dbID,desc,dtaID)"};
__export__ i32 utool_sysSetDBFL(t_kgsWinS *ws, t_kgwThread *t, Word FunID, Byte nPar, Word *ParID)
/* Function */
{
  #define             NumV 2
  char               *c;
  t_systFDB          *r;
  t_kgsVar           *vv;
  short               v0;
  u32                 dbID,dtaID;

  v0=kgs_AllocVB(NumV);
  if (v0<0)
     return -32;
  kgs_ClrVB(&s_v[v0]);
  s_v[v0].v->Type=k_kgsTypeBool;
  *((Byte *)(s_v[v0].v+1))=0;
  if (nPar==3)
     {
     vv=kgs_GetVar(ws,t,*ParID,NULL);             // dbID
     if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0+1]) && *((u64 *)(s_v[v0+1].v+1)))
        {
        dbID=*((u64 *)(s_v[v0+1].v+1));
        r=GetFDB(dbID);
        if (r)
           {
           vv=kgs_GetVar(ws,t,*(ParID+2),NULL);      // dtaID
           if (vv && !kgs_ConvertVar2u64(vv,&s_v[v0+1]))
              {
              dtaID=*((u64 *)(s_v[v0+1].v+1));
              vv=kgs_GetVar(ws,t,*(ParID+1),NULL);   // desc
              if (vv && !kgs_ConvertVar2Str(vv,&s_v[v0+1]))
                 {
                 c=(char *)(s_v[v0+1].v+1);
                 if (dtaID)
                    r->bDta=dtaID;
                 if (*c)
                    r->Desc=txtb_AddText2Buf(&DBLt,&DBLm,&DBLu,c,1);
                 *((Byte *)(s_v[v0].v+1))=1;
```

```
                    }
                }
            }
        }
    }
    s_v[v0].v->bSz+=sizeof(u64);
    kgs_AddVar(&t->stk[t->nStk-1].Loc,FunID,&s_v[v0].v);
    t->stk[t->nStk-1].Flg|=b0;
    kgs_FreeVB(v0,NumV);
    #undef NumV
    return 0;
}
```

From the script, you would call *utool.AdjustRgraph(1045)* for an instance.

Calling to other scripts' functions

Every script's function may be called from within another one, by just preceeding it by the script's name, i.e. <script_name>.<function_name>(<params>)

Just be careful that when called from another script, that function will NOT see it's script's global variables, but the caller's one. So functions intended to be called from the outside should not use global variables at all, or variables must be declared (and be global) in every script that uses such functions. Also note that if they declare some, it will be created in caller's global space.

Limitations

After 2010 rewrite, there should be little restrictions about language concepts, for an instance, you can declare and assign a variable inside a condition, call any function at any moment, etc. as the expression parser and evaluator are now very standard in their behaviour. The only limitation is the number of local variables they can create for an expression evaluation, so there's a limit in the complexity of an expression (currently, it's about more than 30 temporal values, so should be ok for most human-readable expressions).

Script is very handy, and quite fast as it is precompiled at startup, but for really intensive routines, it is still recommended to use C compiled code (on a utool function, for an instance, or the traditional C modules).

Debugging

Scripting is quite new for the application, so it is possible that it still gets some errors. Also it is very easy to make some error in the script itself, so debugging is quite helpful.

Note that *there's a debugger module*, that can run along your application, and that allows you to read the compiled code, see the variables and their values, but also to set breakpoints (both global or per thread) and watchpoints (only for global variables, execution stops on that thread when such variable is changed), and to execute your script step by step, helping you to locate any error that may be around. There's a wiki about this debugger in the project's wiki (see http://kindb.sourceforge.net). The rest of this document talks about the text logs that are generated at runtime (that are also useful for forensics but a little hard for the novice).

Script manager writes lots of information when in debug mode, so just use it in pre-production stages.

Typical debug levels (values for the *Debug Level* entry in kingui.cfg file) are between 1 and 15 (1 logs minor warnings and most errors, 15 logs almost everything for every script's instruction). Script manager output goes to *debug.log* (compiler overall results), to *debug009.log* (most compiler and interpreter output including every instruction executed and operation results) and to debug010.log (shows a bulk disassembly of all the compiled scripts as a reference when analyzing errors).

Examples of debug files:

- debug.log

```
20100210-151016.289592 (14257) U kgs_CompileScript(): compilation for window 'Mfam' took
5.43ms, 4096 bytes of code starting at 133552, 5 local functions declared
```

- debug009.log

```
20100210-174812.633484 (15211) U kgs_ExecThread(1): start executing thread (window cmda), entry
point PC:175024 (cmda.cmda.Init), 0 global vars
20100210-174812.633496 (15211) U kgs_ExecThread(1): executing statement, PC: 175024, Op=46 (0
-> EwID), 8 bytes
20100210-174812.633514 (15211) U kgs_AddVarV(): added 1879:'EwID' at 16 (deleted:0, reused:0),
16 bytes, area now has 1 variables (1 used), 1024 bytes (32 used), varblock: 0x2591300
20100210-174812.633527 (15211) U kgs_ExecThread(1): executing statement, PC: 175032, Op=46 (0
-> CwID), 8 bytes
20100210-174812.633537 (15211) U kgs_AddVarV(): added 1881:'CwID' at 32 (deleted:0, reused:0),
16 bytes, area now has 2 variables (2 used), 1024 bytes (48 used), varblock: 0x2591300
20100210-174812.633549 (15211) U kgs_ExecThread(1): executing statement, PC: 175040, Op=46 (0
-> ceeIx), 8 bytes
20100210-174812.633559 (15211) U kgs_AddVarV(): added 1882:'ceeIx' at 48 (deleted:0, reused:0),
16 bytes, area now has 3 variables (3 used), 1024 bytes (64 used), varblock: 0x2591300
20100210-174812.633571 (15211) U kgs_ExecThread(1): executing statement, PC: 175048, Op=46 (0
-> pwID), 8 bytes
20100210-174812.633581 (15211) U kgs_AddVarV(): added 1883:'pwID' at 64 (deleted:0, reused:0),
16 bytes, area now has 4 variables (4 used), 1024 bytes (80 used), varblock: 0x2591300
20100210-174812.633592 (15211) U kgs_ExecThread(1): executing statement, PC: 175056, Op=17
(return), 8 bytes
20100210-174812.633605 (15211) U kgs_LogVars(): Content of varblock 0x2591300
20100210-174812.633621 (15211) U kgs_LogVars():  000 [o:00016 s:0016] 1879:EwID=(Num64)0
20100210-174812.633632 (15211) U kgs_LogVars():  001 [o:00032 s:0016] 1881:CwID=(Num64)0
20100210-174812.633643 (15211) U kgs_LogVars():  002 [o:00048 s:0016] 1882:ceeIx=(Num64)0
20100210-174812.633656 (15211) U kgs_LogVars():  003 [o:00064 s:0016] 1883:pwID=(Num64)0
20100210-174812.633678 (15211) U kgs_LogVars(): Finished. 4 vars shown, ended at 80 (header
says 4 vars -4 used- and 80 bytes)
20100210-174812.633689 (15211) U kgs_ExecThread(1): Finished. Executed for 194us
```

- debug010.log

  Here you can search by the PC address to read compiled code and get an idea of what happened. PC value is the number after the log's line header (after the 'U'). In this example you can read the 4 instructions executed in previous dabue009.log example, at the end of this disassembly:

```
20100210-174811.159733 (15211) U kgs_LogScriptCode(): script 'cmda'
20100210-174811.159742 (15211) U kgs_LogScriptCode(): disassembly starts
======================>
20100210-174811.159751 (15211) U              :
20100210-174811.159761 (15211) U  159288{  8}: NOp [function GetX()]
20100210-174811.159771 (15211) U  159296{  8}: NOp [1:5 (if), 2:7 (PC: 159352)]
20100210-174811.159782 (15211) U  159304{ 16}: RAss v -> _tv_eval_
20100210-174811.159792 (15211) U  159320{  8}: SkipF 32 [PC=159352]
20100210-174811.159802 (15211) U  159328{ 16}: "X" -> _tv_eval_
20100210-174811.159811 (15211) U  159344{  8}: return ()
20100210-174811.159821 (15211) U  159352{ 16}: "" -> _tv_eval_
20100210-174811.159830 (15211) U  159368{  8}: return ()
20100210-174811.159840 (15211) U              :

(...)

20100210-174811.170569 (15211) U  175024{  8}: 0 -> EwID
20100210-174811.170579 (15211) U  175032{  8}: 0 -> CwID
20100210-174811.170589 (15211) U  175040{  8}: 0 -> ceeIx
20100210-174811.170598 (15211) U  175048{  8}: 0 -> pwID
20100210-174811.170608 (15211) U  175056{  8}: return
20100210-174811.170617 (15211) U kgs_LogScriptCode(): <====================== disassembly
ends
```

Annex A - dialog.dtd

This the Document Type Definition for the application window descriptions. Please refer to latest dialog.dtd present in examples/ directory at the source distribution:

```
<?xml version="1.0" encoding="utf-8" ?>
<!ELEMENT application (title,dialog+)>
<!ATTLIST application
  name CDATA #REQUIRED
  windows CDATA #IMPLIED
  color CDATA "0"
>
<!ELEMENT dialog ( title,taborder?,( initquery | group | statictext | edittext | listbox | combobox |
button | checkbox | radio | datetime | graph | image | script)* )>
<!ATTLIST dialog
  name CDATA #REQUIRED
  width CDATA #REQUIRED
  height CDATA #REQUIRED
  color CDATA "0"
  version CDATA "0"
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT taborder (#PCDATA)>
<!ELEMENT initquery (#PCDATA)>
<!ATTLIST initquery
   id CDATA "A"
>
<!ELEMENT group (group | statictext | edittext | editnum | listbox | combobox | button | checkbox |
radio | datetime | graph | image)* >
<!ATTLIST group
  name CDATA #IMPLIED
  id CDATA "0"
  distribution (V|H) "V"
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #IMPLIED
  height CDATA #IMPLIED
  spacing CDATA #IMPLIED
  border CDATA #IMPLIED
  color CDATA "0"
  hint CDATA ""
>
<!ELEMENT statictext (#PCDATA)>
<!ATTLIST statictext
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #IMPLIED
  height CDATA "10"
  align (left|right|center) "left"
  valign (top|bottom|center) "top"
  id CDATA "0"
  fillfmt CDATA ""
  color CDATA "0"
  border CDATA #IMPLIED
  hint CDATA ""
```

```
>
<!ELEMENT edittext (#PCDATA)>
<!ATTLIST edittext
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "left"
  password (yes|no) "no"
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  maxlen CDATA "0"
  maxchr CDATA "0"
  maxlin CDATA "1"
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
  option (optional|required) "optional"
>
<!ELEMENT listbox (cdef*,li*)>
<!ATTLIST listbox
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA #IMPLIED
  multipleselect (yes|no) "no"
  id CDATA "0"
  key CDATA ""
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
  depth CDATA "0"
>
<!ELEMENT combobox (li*)>
<!ATTLIST combobox
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "left"
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  lines CDATA "2"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
  sorted (ascending|descending|no) "no"
>
<!ELEMENT button (#PCDATA)>
<!ATTLIST button
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "left"
  valign (top|bottom|center) "top"
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  img CDATA ""
  imgw CDATA "10"
  imgh CDATA "10"
  imgm (overlay|exclusive) "overlay"
  imghp (left|right|center) "left"
  imgvp (top|bottom|center) "top"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
>
<!ELEMENT checkbox (#PCDATA)>
<!ATTLIST checkbox
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  selected (yes|no|undef) "yes"
  align (left|right|center) "left"
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
>
<!ELEMENT radio (#PCDATA)>
<!ATTLIST radio
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  selected (yes|no|undef) "yes"
  align (left|right|center) "left"
  id CDATA "0"
  gid CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
>
<!ELEMENT datetime (#PCDATA)>
<!ATTLIST datetime
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "right"
  type (date|time|datetime) "date"
  subtype (udatetime|xdatetime|sdate) "udatetime"
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
>
<!ELEMENT graph (cdef*,li*)>
<!ATTLIST graph
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA #IMPLIED
  id CDATA "0"
  type (vbars|hbars|lines|surfaces|addingsurfaces|addingvbars|addinghbars|gantt|calendar) #REQUIRED
  key CDATA ""
  hint CDATA ""
>
<!ELEMENT image (#PCDATA)>
<!ATTLIST image
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA #IMPLIED
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  hint CDATA ""
>
<!ELEMENT editnum (#PCDATA)>
<!ATTLIST editnum
```

```
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  ofsx CDATA #IMPLIED
  ofsy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "left"
  id CDATA "0"
  key CDATA ""
  fillfmt CDATA ""
  maxlen CDATA "0"
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
  offline (enabled|disabled) "disabled"
  option (optional|required) "optional"
  thsep (yes|no) "yes"
  numdec CDATA "0"
  type (integer|float|currency) "integer"
  ptxt CDATA ""
  limit CDATA "0 | 0"
>
<!ELEMENT script (#PCDATA)>
<!ELEMENT cdef (#PCDATA)>
<!ATTLIST cdef
  width CDATA "4"
  align (left|right|center) "left"
  type (num4|text|atext|uDateTime|sDate|icon|money|money2|FP32|FP64|id32) "text"
  subtype CDATA ""
  editable (yes|no) "no"
  visible (yes|no) "yes"
  fillfmt CDATA ""
>
<!ELEMENT li (cd+)>
<!ATTLIST li
  selected (yes|no) "no"
  child (same|down|up1|up2|up3|up4|up5|up6|up7|up8|up9) "same"
  id CDATA "0"
>
<!ELEMENT cd (#PCDATA)>
<!ATTLIST cd
  color CDATA "0"
  tattr (normal|bold|italic|bolditalic) "normal"
>
<!ELEMENT include (#PCDATA)>
<!ATTLIST include
  window CDATA ""
  id CDATA "0"
>
```

Annex A bis – other DTDs

User profile DTD

```
<?xml version="1.0" ?>
<!ELEMENT profile (description?,(application|var)* )>
<!ELEMENT description (#PCDATA)>
<!ELEMENT application ((dialog|var)*)>
<!ATTLIST application
  name CDATA #REQUIRED
  atr (Hidden|ReadOnly|RW) "Hidden"
>
<!ELEMENT dialog ((item|var)*)>
<!ATTLIST dialog
  name CDATA #REQUIRED
  atr (Hidden|ReadOnly|RW) "RW"
>
<!ELEMENT item EMPTY>
<!ATTLIST item
  id CDATA #REQUIRED
  atr (Hidden|ReadOnly|RW) "RW"
>
<!ELEMENT var EMPTY>
<!ATTLIST var
  name CDATA #REQUIRED
  value CDATA "0"
>
```

User Interface state DTD

This is used to save current GUI state (windows and their values) so the complete user session can be saved for later use on a brand new environment. This is just a draft yet, and should not be confused with *dialog.dtd*.

```
<?xml version="1.0" ?>
<!ELEMENT application (dialog+)>
<!ATTLIST application
  name CDATA #REQUIRED
>
<!ELEMENT  dialog  (  (statictext|edittext|listbox|combobox|button|checkbox|radio|datetime|graph|image)*)
>
<!ATTLIST dialog
  name CDATA #REQUIRED
>
<!ELEMENT statictext (#PCDATA)>
<!ATTLIST statictext
  id CDATA #REQUIRED
>
<!ELEMENT edittext (#PCDATA)>
<!ATTLIST edittext
  id CDATA #REQUIRED
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  seli CDATA #IMPLIED
  sele CDATA #IMPLIED
  enabled (yes|no) "yes"
>
<!ELEMENT listbox (cdef*,li*)>
<!ATTLIST listbox
  id CDATA #REQUIRED
  enabled (yes|no) "yes"
  depth CDATA "0"
>
<!ELEMENT combobox (li*)>
<!ATTLIST combobox posx CDATA
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  id CDATA #REQUIRED
  key CDATA ""
  fillquery CDATA ""
  fillfmt CDATA ""
  lines CDATA "2"
  hint CDATA ""
  enabled (yes|no) "yes"
>
<!ELEMENT button (#PCDATA)>
<!ATTLIST button
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "left"
  id CDATA #REQUIRED
  key CDATA ""
  fillquery CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  img CDATA ""
  imgw CDATA "10"
  imgh CDATA "10"
  hint CDATA ""
  enabled (yes|no) "yes"
>
<!ELEMENT checkbox (#PCDATA)>
<!ATTLIST checkbox
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  selected (yes|no|undef) "yes"
  align (left|right|center) "left"
  id CDATA #REQUIRED
  key CDATA ""
  fillquery CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
>
<!ELEMENT radio (#PCDATA)>
<!ATTLIST radio
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  selected (yes|no|undef) "yes"
  align (left|right|center) "left"
  id CDATA #REQUIRED
  gid CDATA "0"
  key CDATA ""
  fillquery CDATA ""
  fillfmt CDATA ""
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
>
<!ELEMENT datetime (#PCDATA)>
<!ATTLIST datetime
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA "10"
  align (left|right|center) "right"
  type (date|time|datetime) "date"
  id CDATA #REQUIRED
  key CDATA ""
  color CDATA "0"
  hint CDATA ""
  enabled (yes|no) "yes"
>
<!ELEMENT graph (#PCDATA)>
<!ATTLIST graph
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA #IMPLIED
  id CDATA #REQUIRED
  type (vbars|hbars|lines|surfaces|addingsurfaces|addingvbars|addinghbars|gantt|calendar) #REQUIRED
  key CDATA ""
  hint CDATA ""
>
<!ELEMENT image (#PCDATA)>
<!ATTLIST image
  posx CDATA #IMPLIED
  posy CDATA #IMPLIED
  disx CDATA #IMPLIED
  disy CDATA #IMPLIED
  width CDATA #REQUIRED
  height CDATA #IMPLIED
  id CDATA #REQUIRED
  key CDATA ""
  hint CDATA ""
>
<!ELEMENT cdef (#PCDATA)>
<!ATTLIST cdef
  width CDATA #IMPLIED
  align (left|right|center) "left"
  visible (yes|no) "yes"
>
```

```
<!ELEMENT li (cd+)>
<!ATTLIST li
  selected (yes|no) "no"
  child (same|down|up1|up2|up3|up4|up5|up6|up7|up8|up9) "same"
  id CDATA #REQUIRED
>
<!ELEMENT cd (#PCDATA)>
<!ATTLIST cd
  color CDATA "0"
 tattr (normal|bold|italic|bolditalic) "normal"
>
```

Annex B – DB Query attributes

`initquery` and `fillfmt` allow for automatic filling of window items from DB queries upon window creation.

- `initquery` has the same format as any other Query Request (see *kinql.pdf*), but an attribute notes an internal identification letter (A through Z), like `<initquery id=”A”>_user.login=1</initquery>`.
- `fillfmt` is a window item attribute with a superset of a Result Request Specification (see *kinql.pdf*)

`fillfmt` format adds, to normal Result Request Specification, operators to allow for value concatenation (you can combine more than one field value to fill every window item cell) and also attributes to specify individual window item cells (columns on a ListBox, for an instance). All that is preceeded by the query internal identification letter and a colon, like in `<statictext id=”100” align=”center” width=”60” fillfmt=”A:_user.name+' '+.cog1” />`. Note the use of single quotes inside the double quotes section.

Operators are the plus sign, and as values, it allows raw field names (that will be requested to the DB), and text literals, enclosed between double quotes (may be &quot; when found on an XML source) or single quotes (&apos;).

Most window items allow for only one result request specification, but there are exceptions:

- StaticText only allows for one specification (and one result).
- EditText is the same case.
- ListBox allows one specification for every column (visible or not). Every result is placed onto a new row.
- ComboBox allows for two specifications: first is for row ID, second for row content. Both must be present (but if you don't need the ID, you can keep that specification empty). Every result is placed onto a new row.
- DateTime allows for one specification, and one result.
- Graph does not allow this attribute, but should be placed on the <cdef> tags (when implemented), one for each dimension.
- Image allows for one specification, and one result.

Use commas to separate multiple specifications.

Annex C – Print language

Printing in KinGUI is performed by creating a print job, and sending a series of commands and text to it. These commands (and some variables) are explained here, and all are a two-letter code following a dollar sign (*$xx*) and, optionally, followed by some param(s) between parentheses. Params allow for simple arithmetic (sums, substractions...) with constants and internal variables. Note that writing a dollar sign requires escaping it (as double dollar *$$*).

When talking about standard units (measures), they are always 10x mm (that is, tenths of millimeter, e.g. 1000 means 100mm or 0.1meter)

Commands

Alignment

$al – Align left

It sets text alignment to the left from this command on. This is the default alignment.

$ar – Align right

It sets text alignment to the right from this command on. Alignment is reset to default (left) after newline.

$ac – Align center

It sets text alignment to be centered from this command on. Alignment is reset to default (left) after newline.

Text placement

$hg() – Set horizontal position

Sets current horizontal text position, param is position in standard units from left page margin (e.g. *$hg(200)* will start writing next text 20mm from the left margin).

$vg() – Set vertical position

Sets current vertical text position, param is position in standard units from top page margin (e.g. *$vg(200)* will start writing next text 20mm from the topmost margin; *$vg($ph-1000)* will start text at 100mm above bottom margin).

$tr() – Set text rotation

Sets current text rotation, param is absolute rotation from horizontal, in degrees. Absolute means that it does not accumulate: a second text rotation starts at the same horizontal (0 degrees) and not the previous text rotation value. Note that for *$tr(90)*, *$hg(200)* then means 20mm from the bottom margin of the page.

$tl() – Set text bounding limit

Sets current text end position, param is length in standard units from current position. This effectively creates a bounding box, and gets reset after newline.

Typical use is to align text within an area.

$cr – Carriage Return

Moves to next line starting point (and also resets many attributes like alignment, text bounding box, etc).

Text attributes

$tb – Bold text

Sets attribute for following text, until next *$t_*, carriage return or end of line (\n). This is deprecated in favor of using a bold font (see *$tf()*).

$ti – Italic text

Sets attribute for following text, until next *$t_*, carriage return or end of line (\n). This is deprecated in favor of using a bold font (see *$tf()*).

$tn – Normal text

Clears attributes for following text.

$tf() – Set/change current text font

Changes text font. Param is font index (a value starting at 1, as returned by DefTxtFont(), for an instance).

$tc() – Set/change current text color

Changes text color. Param is gray level (if there's only one value) or RGB value (three, comma separated, values). Values are 0=black, 255=maximum (white, or red, green or blue).

Drawing

$dl() – Draw a line, absolute

Draws a line. When four params are present, they are (ox, oy, ex, ey): absolute origin and end in standard units. Fifth param (optional) is line width, in standard units.

$dr() – Draw a line, relative

Draws a line. When four params are present, they are (ox, oy, disx, disy): absolute origin and relative end (displacement from origin, it can be negative or positive) in standard units. Fifth param (optional) is line width, in standard units.

$dq() – Draw a rectangle outline, absolute

Outlines a rectangle. When four params are present, they are (ox, oy, ex, ey): absolute top-left corner, and absolute bottom-right corner in standard units. Fifth param (optional) is line width, in standard units.

$db() – Draw a rectangle outline, relative

Outlines a rectangle. When four params are present, they are (ox, oy, disx, disy): absolute top-left corner, and relative bottom-right corner in standard units. Fifth param (optional) is line width, in standard units.

$da() – Draw a solid rectangle, absolute

Paints a rectangle. When four params are present, they are (ox, oy, ex, ey): absolute top-left corner, and absolute bottom-right corner in standard units.

$dc() – Draw a solid rectangle, relative

Paints a rectangle. When four params are present, they are (ox, oy, disx, disy): absolute top-left corner, and relative bottom-right corner in standard units.

$lw() – Set line width

Sets line width for subsequent drawing operations. Param is thickness in standard units.

Variables

Text position and attributes

$tx – Current X position

Holds current horizontal position from left margin, in standard units.

$ty – Current Y position

Holds current vertical position from top margin, in standard units.

$th – Current text height

Holds current text height (full, no baseline), in standard units.

Page information

$pw – Page width

Holds absolute page width, in standard units (e.g. for an A4 page, that would be 2100, that is 210.0mm).

$ph – Page heigth

Holds absolute page height, in standard units (e.g. for an A4 page, that would be 2970, that is 297.0mm).

$uw – User width

Holds user page width (usable width between margins), in standard units.

$uh – User height

Holds user page height (usable width between margins), in standard units.

Document information

$pn – Page number

Holds current page number

$tp – Total number of pages

Holds the number of pages of the whole document

Annex D – KinGUI events description

Most of the user interaction is performed through events issued by the framework to your managers. In this section we'll enumerate all the possible events you may receive, and how to use the provided information to get the job done. Note that these are the real binary events inside KinGUI application (and to your binary libraries), and only some of them are mapped to script events (where noted). This is a short list:

Event	Description
k_kgmeNOP	Should never be received at all.
k_kgmeInit	Issued just after the creation of a window -or window overlay- (and before it is displayed for the first time). It is intended to let the user fill any window items that are dynamically set, and also to prepare application dependent buffers, variables, etc. If exists, win.Init() is script handler called instead.
k_kgmeDone	Issued when a window is about to be destroyed. You receive the event *before* anything is destroyed, so you have access to all the window structures (window items, buffers, etc). It is intended to allow you to save things, but specially to release all the buffers you may hold. If exists, win.Done() is script handler called instead.
k_kgmeAct	Issued on activation of a window item. This is a very generic event, and usually a specific examination of the *itm* structure is required to take an action. If exists, win.Act.iXXX() is script virtual handler called instead (XXX is item ID).
k_kgmwPkt	Issued when a network packet is received with CR1 field holding your window *wID*. It carries a pointer to the received packet. If exists, query.Recv(qID) is script handler called **if** query gets finished with this packet (otherwise, query.Wait(qID) will not block after this packet is received if it fulfills pending query). Note that qID is actually held into CR2 field.
k_kgmeMod	Issued on modification of state of a window item, this is as generic as k_kgmeAct, it often is issued in combination with it but should not be confused with each other. If exists, win.Mod.iXXX() is script handler called instead, where XXX is item ID.
k_kgmeMsg	Issued by user application to user application, this is an 'Interprocess Communication' method based in messages. User just provides a pointer to a data buffer, its size, and a message ID (also user defined). If exists, win.wMsg(fwID,mID,msg) is script handler called instead.
k_kgmeSelMod	Issued to indicate a change in the selection state of a window item, it also supplies a 'Row' value (and sometimes even a 'Col') for lists to give a clue on what is changing. If exists, win.SelMod.iXXX(row,col,flg) is script handler called instead, where XXX is item ID.
k_kgmeLocMenu	Issued upon different strategies, indicated by the 'Func' field: 1 indicates that menu is being composed, and a user manager may remove or add options. 2 just notifies about a user selection, providing the *mID* of the choosed option. If exists, win.LocM.iXXX() is script virtual handler called instead, where XXX is item ID.
k_kgmeSEv	This is a subscription event: an application registers to receive notifications when specific events happen. To register, you use a registration function.
k_kgmeTmr	This is a timer event: you receive it after a specified time lapse. A timer may be one-time or just repeat periodically, depending on the attributes you set on kgw_TimerSet() call. If exists, win.Tmr.tXXX() is script virtual handler called instead, where XXX is timer ID number.

Detail – Binary version

This is a more detailed list, with params meaning in every case (for specific references to script's handlers, skip this table and see following one):

Event	Description
k_kgmeNOP	
k_kgmeInit	Issued just after the creation of a window -or window overlay- (and before it is displayed for the first time). It is intended to let the user fill any window items that are dynamically set, and also to prepare application dependent buffers, variables, etc. This can be seen as a local window constructor. User often obtains window pointer, and uses *UsrD* pointer to assign some memory buffer (often a structure) with internal variables to keep state information for the duration of the window. In case you forget to free it, it will be done automatically on window destructor -provided you don't have more allocated pointers inside-. Please don't forget to set it back to NULL if you free it on your own destructor (see k_kgmeDone).
k_kgmeDone	Issued when a window is about to be destroyed. You receive the event *before* anything is destroyed, so you have access to all the window structures (window items, buffers, etc). It is intended to allow you to save things, but specially to release all the buffers you may hold. You may also want to send messages to other windows, telling them about your imminent destruction, sending them some data, state, etc. Please don't forget to set freed *UsrD* window pointer to NULL, or the window manager will try to free it again and break the heap.
k_kgmeAct	Issued on activation of a window item. This is a very generic event, and usually a specific examination of the *itm* structure is required to take an action. These are some hints depending on the item type: - t_kwWinI1 (Static Text): Issues this event whenever it is clicked (if it is not disabled and it is visible) - t_kwWinI2 (Edit Text): Issues this event whenever Enter key is pressed (main or keypad) - t_kwWinI5 (Button): Issues this event whenever it is pressed (clicked or space pressed), if it is not disabled and it is visible - t_kwWinI7 (Graphic): Issues this event in case there's a click on the item area (scroll bar is excluded: no event is generated then). User may test graphic cursor (*cIdx* field) to know what variable or task was clicked. For resource-based graphics, it also sets current task (selected by user on the graphic) pointed by *cIdx* (check *Flg.b0* to ensure it was due to this action though). - t_kwWinI9 (Check Box): Issues this event whenever it is clicked (if it is not disabled and it is visible). User should check for b0 in St field to know about its current status. This event is issued *after* status has changed; before that, a k_kgmeMod event was issued, just *before* changing status. - t_kwWinI11 (Radio Button): Issues this event when its state goes to 'selected', but you may check also St field for b0 set.
k_kgmwPkt	Issued when a network packet is received with CR1 field holding your window *wID*. It carries a pointer to the received packet (be careful not to modify it as it is not copied but just referenced on the receive queue).
k_kgmeMod	Issued on modification of state of a window item, this is as generic as k_kgmeAct, it often is issued in combination with it but should not be confused with each other. Some hints about its usage: - t_kwWinI2 (Edit Text): Issues this event whenever content is modified by the user interaction (not if you modify it setting its value).

Event	Description
	- t_kwWinI3 (List Box): Issues this event whenever its selection state changes by the user interaction: toggle selection of a row in multiple select list (in such case, a k_kgmeSelMod event is generated just before this one), change selection on single selection list (in this case, k_kgmeSelMod event will be issued just after this one). Also when cursor is just moved (with keys). It is also issued when the application selects a row using kgw_SetWIi_LBSel(), but this shouldn't happen, and this behavior will probably be removed.
	- t_kwWinI4 (Combo Box): Issues this event whenever its selection changes by the user interaction. Actually, this event is issued just after changing to the new selection value; a k_kgmeSelMod event is issued just before changing the previous selection value. This event is also issued when user uses the key fast filter (types part of a word to filter visible values), because the filter string (available as 'me' field) has changed. In this case, no k_kgmeSelMod is issued as selection didn't change.
	- t_kwWinI6 (DateTime): Issues this event after its value is changed (by user interaction, no event is issued if you set it from application). You may check current date/time value ('DT' field).
	- t_kwWinI9 (Check Box): Issues this event just *before* its state is toggled. Then, a k_kgmeAct event is sent *after* state has changed to the new value.
	- t_kwWinI11 (Radio Button): Issues this event just *before* changing its state to 'selected'; then it is selected and k_kgmeAct event is issued; finally, the previously selected Radio Button is notified with this event (once its state is set to 'unselected').
k_kgmeMsg	Issued by user application to user application, this is an 'Interprocess Communication' method based in messages. User just provides a pointer to a data buffer, its size, and a message ID (also user defined). Care must be taken to ensure that processes don't fill message buffers out of their size. KinGUI framework also uses messages, internally, for some high level operations, but never to a user window manager.
k_kgmeSelMod	Issued to indicate a change in the selection state of a window item, it also supplies a 'Row' value (and sometimes even a 'Col') for lists to give a clue on what is changing.
	- t_kwWinI3 (List Box): Issues this event when user double-clicks a cell and the list is not editable (so user may take care of the action), or simply clicks on a cell (for multiple select, a k_kgmeMod is also issued, just following this; for a single selection list, this event happens *after* setting the selection state, a k_kgmeMod is issued *before* that); in these cases, 'Col' field is usually valid (65535 indicates invalid: out of range or not applicable). It is also issued when user just moves the selection with some keys, in these cases, 'Col' field is not valid, and 'Row' is the one that is loosing the cursor (as this event is issued *before* changing position); a k_kgmeMod event is issued *after* position has changed.
	- t_kwWinI4 (Combo Box): Issues this event when user clicks on a value on the list (and only then, keys don't issue this event), changing current selected value. This event happens *after* the new selection value is in effect; *before* that, a k_kgmeMod event is issued so user can react to a loose of selection on a specific value.
	- t_kwWinI7 (Graphic): For a resource based graphic, it issues this event when user changes task selection (both with a click or just moving with cursor arrows). Row gets the task index that gets selected.

Event	Description
k_kgmeLocMenu	Issued upon different strategies, indicated by the 'Func' field: a value of 1 indicates that menu is being composed, and a user manager receiving this event may remove system options, or add its own ones, before it is displayed. A value of 2 just notifies about a user selection, providing the *mID* of the choosed option. User may just return 0 to allow KinGUI framework to perform the default actions for system options. Returning a different value will prevent any action to be performed.
k_kgmeSEv	This is a subscription event: an application registers to receive notifications when specific events happen. To register, you use a registration function; currently, only network monitoring is implemented, by calling king_nwEvent() to register whatever you want to get. It works based on a mask (of the events you want to receive), specific for every module and registration function. For an instance, for the network module, mask bits are as follows: b0=>connection list changes, b1=>connection status changes, b2=>Tx bytes changes, b3=>Rx bytes changes. This event will be issued for other modules in the future as well.
k_kgmeTmr	This is a timer event: you receive it after a specified time lapse. A timer may be one-time or just repeat periodically, depending on the attributes you set on kgw_TimerSet() call. You receive the tID value you set upon timer creation, for your reference.

Detail – Script version (handlers)

This is a more detailed list, with params meaning in every case for script's handlers version of events:

Event	Description
win.Init()	Issued just after the creation of a window -or window overlay- (and before it is displayed for the first time). You can init things not set at the global sections of the script (like calling functions or other tools, or setting discrete values).
win.Done()	Issued when a window is about to be destroyed. You receive the event **before** anything is destroyed, so you have access to all the window data. One useful thing to do here is to close any child window so they get closed too when this one is (otherwise, they would get orphaned but still alive). You can also notify parent about any data it may need before disappearing.
win.Act.iXXX()	Issued on activation of a window item. These are some hints depending on the item type: - Static Text: Issues this event whenever it is clicked (if it is not disabled and it is visible) - Edit Text: Issues this event whenever Enter key is pressed (main or keypad) - Button: Issues this event whenever it is pressed (clicked or space pressed), if it is not disabled and it is visible - Graphic: Issues this event in case there's a click on the item area (scroll bar is excluded: no event is generated then). User may test graphic cursor (*cIdx* field) to know what variable or task was clicked. For resource-based graphics, it also sets current task (selected by user on the graphic) pointed by *cIdx* (check *Flg.b0* to ensure it was due to this action though). - Check Box: Issues this event whenever it is clicked (if it is not disabled and it is visible). User should check to know about its current status. This event is issued *after* status has changed; before that, a k_kgmeMod event was issued, just *before* changing status.

Event	Description
	- Radio Button: Issues this event when its state goes to 'selected', but you may prefer to check its status instead.
query.Recv(qID)	Issued when last network packet is received with CR1 field holding your window *wID*, indicating that given query (identified by its qID) has been already fulfilled (finished).
win.Mod.iXXX()	Issued on modification of state of a window item; this is as generic as k_kgmeAct, it often is issued in combination with it but should not be confused with each other. Some hints about its usage: - Edit Text: Issues this event whenever content is modified by the user interaction (not if you modify it setting its value). - List Box: Issues this event whenever its selection state changes by the user interaction: • toggle selection of a row in multiple select list (in such case, a k_kgmeSelMod event is generated just before this one) • change selection on single selection list (in this case, k_kgmeSelMod event will be issued just after this one) • when cursor is just moved (with keys). • when the application selects a row using kgw_SetWIi_LBSel(), but this shouldn't happen (this behavior will probably be removed in some future if not already). - Combo Box: Issues this event whenever its selection changes by the user interaction. Actually, this event is issued just after changing to the new selection value; a k_kgmeSelMod event is issued just before changing the previous selection value. This event is also issued when user uses the key fast filter (types part of a word to filter visible values), because the filter string has changed. In this case, no k_kgmeSelMod is issued as selection didn't change. - DateTime: Issues this event after its value is changed (by user interaction, no event is issued if you set it from application). You may want to check current date/time value. - Check Box: Issues this event just **before** its state is toggled. Then, a k_kgmeAct event is sent **after** state has changed to the new value. - Radio Button: Issues this event just **before** changing its state to 'selected'; then it is selected and k_kgmeAct event is issued; finally, the previously selected Radio Button is notified with this event (once its state is set to 'unselected').
win.wMsg (fwID,mID,msg)	Issued by user application to user application, this is an 'Interprocess Communication' method based in messages. User just provides a pointer to a data buffer, its size, and a message ID (also user defined). Care must be taken to ensure that processes don't fill message buffers out of their size. KinGUI framework also uses messages, internally, for some high level operations, but never to a user window manager. Script receives this data encapsulated into a BB variable.
win.SelMod.iXXX (row.col.flg)	Issued to indicate a change in the selection state of a window item, it also supplies a 'Row' value (and sometimes even a 'Col') for lists to give a clue on what is changing, and a Flg with some additional info (currently, b0=1 when double-clicked).

Event	Description
	- List Box: Issues this event when: • user double-clicks a cell and the list is not editable (so user may take care of the action). Flg.b0=1. • user clicks on a cell (for multiple select, a k_kgmeMod is also issued, just following this; for a single selection list, this event happens **after** setting the selection state, a k_kgmeMod is issued **before** that); in these cases, 'Col' field is usually valid (65535 indicates invalid: out of range or not applicable). • user just moves the selection with some keys; in these cases, 'Col' field is not valid, and 'Row' is the one that is loosing the cursor (as this event is issued **before** changing position); a k_kgmeMod event is issued **after** position has changed. - Combo Box: Issues this event when user clicks on a value on the list (and only then: keys don't issue this event), changing current selected value. This event happens **after** the new selection value is in effect; **before** that, a k_kgmeMod event is issued so user can react to a loose of selection on a specific value.
win.LocM.iXXX (mID,Func)	Issued upon different strategies, indicated by the 'Func' field: a value of 1 indicates that menu is being composed, and a user manager receiving this event may remove system options, or add its own ones, before it is displayed. This is not implemented yet. A value of 2 just notifies about a user selection, providing the *mID* of the choosed option. User may just return 0 to allow KinGUI framework to perform the default actions for system options. Returning a different value will prevent any action to be performed.
win.Tmr.tXXX()	This is a timer event: you receive it after a specified time lapse. A timer may be one-time or just repeat periodically, depending on the attributes you set on kgw_TimerSet() call. You receive the tID value you set upon timer creation, for your reference.

Annex E – KinGUI resource files examples

Script library

First example is an empty application that only contains useful common code (i.e. a script library). For an instance, you may call *tools.GetX(1)* to get “X” (this example function is used to fill table columns for boolean flags with human-readable “X” -true- or empty -false- instead of the numeric equivalent “1” or “0”).

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE application SYSTEM "dialog.dtd">
<application name="tools" color="#d0d0e8">
  <!-- This has no real application, but only fake windows with reusable code -->
  <title>Tools</title>
  <dialog width="100" height="100" name="stool" version="0.2.0">
    <script>
      <![CDATA[

      function GetX(v)
      // Return 'empty' or 'X' depending on 'v' being false or true
      {
        if (v>0)
           return "X";
        return "";
      }

      function GetSaveReqRef(ref)
      // Returns text with ref for a save request
      {
        if (ref)
           return "<"+ref+">";
        else
           return "0";
      }

      // === Set tools ===
      function GetNumInSet(set,Val)
      // Returns index of Val into set, or -1 if not present
      // set is expected to be an array
      {
        var i=0;
        var ni=system.Dim(set);
        while (i<ni)
          {
          if (set[i]==Val)
             return i;
          i++;
          }
        return -1;
      }

      // === Printing tools ===
      function PrintTableHeaders(bx,cy,ics,hh,nc,cw,th)
      // Prints table headers
      // bx=leftmost offset; cy=current Y; ics=intercolumn separation; hh=header height
      // nc=# of columns; cw=array[nc] with column widths; th=text with headers names
      //  (separated by |)
      {
        var i=0;
        var cx=bx;
        var cp=0;
        var hl=system.Dim(th);
        while (i<nc)
          {
          var e=system.TxtPos(th,cp,"|");
          if (e)
             {
             var s=system.TxtMiddle(th,cp,e-cp);
             cp=e+1;
             }
          else
             {
             var s=system.TxtRight(th,cp);
             cp=hl+1; // Ensure we break
```

```
            }
          print.Write("$db("+(cx+2)+","+(cy-hh+4)+","+(cw[i]-4)+","+hh+",1)$hg("+cx+")$vg("+cy+")
           $tl("+cw[i]+")$ac"+s);
          if (cp>=hl)
             break;
          cx+=cw[i]+ics;
          i++;
          }
      }

      function PrintTableRow(bx,cy,ics,nc,cw,s,atr)
      // Prints table headers
      // bx=leftmost offset; cy=current Y; ics=intercolumn separation
      // nc=# of columns; cw=array[nc] with column widths; s=array[nc] with column content
      // atr: b5=1 => hide null values
      {
        var i=0;
        var cx=bx;
        var cp=0;
        while (i<nc)
          {
          switch (system.GetType(s[i]))
            {
            case "number":
              if (s[i] || !(atr & 32))
                 print.Write("$hg("+cx+")$vg("+cy+")$tl("+cw[i]+")$ar"+system.GetNumS(s[i],0));
              break;
            case "currency":
              if (s[i] || !(atr & 32))
                 print.Write("$hg("+cx+")$vg("+cy+")$tl("+cw[i]+")$ar"+system.GetMoneyS(s[i],16));
              break;
            default:
              print.Write("$hg("+cx+")$vg("+cy+")$tl("+cw[i]+")$al"+s[i]);
            }
          cx+=cw[i]+ics;
          i++;
          }
      }

      ]]>
    </script>
  </dialog>
</application>
```

Default application

This is the resource file for the default public application (that is: login and configuration windows, and a few popups). This can serve as a general example, with several ways to express layout (with no script, in this case):

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE application SYSTEM "dialog.dtd">
<!-- This is main application. It includes common windows 'Login' and 'Config',
     but also many common tools -->
<!-- You may add some window of your own to the initial windows list below -->
<application name="start" windows="Login,Config,Services">
  <title>Start</title>

  <!-- This part shouldn't be modified. You might append your own windows at the end,
       but better use other files for this same application name and they will be aggregated -->
  <!-- You may also use a separate file with only the application tag (no dialogs inside)
       to enforce window order -->

  <dialog width="260" height="50" name="Login" version="0.1.1">
    <title>Login</title>
    <group posx="5" posy="5" spacing="5" width="600" height="200">
      <group id="1000" ofsx="5" ofsy="5" width="185" height="65" name="Login" border="2">
        <statictext ofsx="5" ofsy="10" width="90" align="right">User name:</statictext>
        <edittext id="101" ofsx="100" ofsy="10" width="80" align="left" maxlen="16"
                  hint="Type here your user name" offline="enabled"/>
        <statictext ofsx="5" ofsy="25" width="90" align="right">Password:</statictext>
        <edittext id="102" ofsx="100" ofsy="25" width="80" align="left" password="yes" maxlen="16"
                  hint="Type here your password" offline="enabled"/>
        <button id="1" ofsx="100" ofsy="40" width="80" height="20" align="center" valign="center"
                key="K" color="#c8f0c0" img="ok" imgw="15" imgh="15"
```

```xml
              hint="Send your user and password for validation" offline="enabled">Log in</button>
        </group>
        <group id="1001" ofsx="195" ofsy="5" width="270" height="65" name="Server Connections"
               border="2">
          <listbox id="110" ofsx="5" ofsy="10" width="260" height="50"
                   hint="Available servers and connection status">
            <cdef width="0" align="left" type="num4" visible="no">cID</cdef>
            <cdef width="70" align="right" type="text" visible="yes">IP Address</cdef>
            <cdef width="20" align="center" type="text" visible="yes">St</cdef>
            <cdef width="85" align="left" type="uDateTime" subtype="0" visible="yes">Since</cdef>
            <cdef width="40" align="right" type="num4" subtype="2048" visible="yes">Rx</cdef>
            <cdef width="40" align="right" type="num4" subtype="2048" visible="yes">Tx</cdef>
          </listbox>
        </group>
        <group id="1004" ofsx="470" ofsy="5" width="" height="65" name="Session" border="2">
          <button id="140" ofsx="5" ofsy="10" width="100" height="25" align="center"
                  valign="center" key="S" img="ok" imgw="15" imgh="15"
                  hint="Save your current session layout"
                  offline="enabled">Save session layout</button>
          <button id="999" ofsx="5" ofsy="40" width="100" height="20" align="center"
                  valign="center" color="#f0c8c0" img="cancel" imgw="15" imgh="15"
                  hint="Close the whole application" offline="enabled">Exit</button>
        </group>
        <group id="1002" ofsx="5" ofsy="75" width="185" height="40" name="Logged user" border="2">
          <statictext ofsx="5" ofsy="10" width="50" align="right">Name:</statictext>
          <statictext id="120" ofsx="60" ofsy="10" width="120" align="left"/>
          <statictext ofsx="5" ofsy="25" width="50" align="right">At:</statictext>
          <statictext id="121" ofsx="60" ofsy="25" width="120" align="left"/>
        </group>
        <group id="1006" ofsx="5" ofsy="120" width="185" height="25" name="Local time" border="2">
          <statictext id="160" ofsx="5" ofsy="10" width="175" align="center"/>
        </group>
        <group id="1100" ofsx="195" ofsy="75" width="270" height="45" name="User identification"
               border="2">
          <button id="1102" ofsx="175" ofsy="15" width="90" height="25" align="center" valign="center"
                  key="S" img="ok" imgw="15" imgh="15"
                  hint="Take a picture of you for identification" offline="enabled">
            Identification using Camera</button>
        </group>
      </group>
    </dialog>

    <dialog width="260" height="70" name="UserPicCapt" version="0.1.0">
      <title>Picture Capture</title>
      <group distribution="V" posx="5" posy="5" spacing="5">
        <group id="1000" ofsx="5" ofsy="5" width="340" height="295" name="Images" border="1">
          <image id="1001" ofsx="5" ofsy="15" width="160" height="120" hint="Captured image"/>
          <statictext ofsx="5" ofsy="135" width="160" align="center">Captured</statictext>
          <image id="1002" ofsx="170" ofsy="15" width="160" height="120" hint="Background image"/>
          <statictext ofsx="170" ofsy="135" width="160" align="center">Background</statictext>

          <image id="1003" ofsx="5" ofsy="155" width="160" height="120" hint="Foreground image"/>
          <statictext ofsx="5" ofsy="275" width="160" align="center">Foreground</statictext>
          <image id="1004" ofsx="170" ofsy="155" width="160" height="120" hint="Edges detect image"/>
          <statictext ofsx="170" ofsy="275" width="160" align="center">Edges</statictext>

        </group>
        <button id="1" ofsx="350" ofsy="10" width="60" height="25" align="center" valign="center"
                key="K" color="#c8f0c0"
                hint="Start capture and identification" offline="enabled">Capture!</button>
        <button id="2" ofsx="350" ofsy="45" width="60" height="25" align="center" valign="center"
                key="C" color="#c8c0f0"
                hint="Cancel process (do not capture nor identify by image)"
                offline="enabled">Cancel</button>
      </group>
    </dialog>

    <dialog width="260" height="70" name="Config" version="0.1.1">
      <title>Configuration</title>
      <group distribution="H" posx="5" posy="5" width="600" height="400" spacing="5">
        <group id="1000" ofsx="5" ofsy="5" width="310" height="100" name="Localization Settings"
               border="1">
          <statictext ofsx="5" ofsy="10" width="80" align="right">Country:</statictext>
          <combobox id="106" ofsx="90" ofsy="10" width="215" lines="20" hint="Select your country"
                    offline="enabled"/>
          <statictext ofsx="5" ofsy="25" width="80" align="right">Language:</statictext>
          <combobox id="100" ofsx="90" ofsy="25" width="215" lines="20" hint="Select your language"
                    offline="enabled"/>
          <statictext ofsx="5" ofsy="40" width="80" align="right">Date format:</statictext>
          <combobox id="101" ofsx="90" ofsy="40" width="80" lines="3"
                    hint="Select your preferred date format" offline="enabled">
            <li id="1"> <cd>MM/DD/YYYY</cd> </li>
            <li id="2"> <cd>DD/MM/YYYY</cd> </li>
            <li id="3"> <cd>YYYY/MM/DD</cd> </li>
          </combobox>
          <statictext ofsx="5" ofsy="55" width="80" align="right">Currency:</statictext>
          <combobox id="102" ofsx="90" ofsy="55" width="80" lines="20"
                    hint="Select your working currency" offline="enabled"/>
          <statictext ofsx="175" ofsy="40" width="65" align="right">Thousand sep:</statictext>
          <combobox id="103" ofsx="245" ofsy="40" width="60" lines="3"
                    hint="Select your preferred thousands separator" offline="enabled">
            <li id="44" selected="yes"> <cd>, (comma)</cd> </li>
            <li id="46" selected="no"> <cd>. (dot)</cd> </li>
          </combobox>
          <statictext ofsx="175" ofsy="55" width="65" align="right">Decimal sep:</statictext>
          <combobox id="104" ofsx="245" ofsy="55" width="60" lines="3"
                    hint="Select your preferred decimal separator" offline="enabled">
            <li id="44" selected="no"> <cd>, (comma)</cd> </li>
            <li id="46" selected="yes"> <cd>. (dot)</cd> </li>
          </combobox>
          <statictext ofsx="175" ofsy="70" width="65" align="right">Bank account:</statictext>
          <combobox id="105" ofsx="245" ofsy="70" width="60" lines="3"
                    hint="Select your preferred bank account default format for your national banks"
                    offline="enabled">
            <li id="0" selected="no"> <cd>Local</cd> </li>
            <li id="1" selected="yes"> <cd>IBAN</cd> </li>
          </combobox>
          <statictext ofsx="5" ofsy="85" width="80" align="right">Time Zone:</statictext>
          <combobox id="170" ofsx="90" ofsy="85" width="80" align="left" lines="8"
                    hint="Select your time zone with respect to Universal Time (UTC)"
                    offline="enabled"/>
          <checkbox id="171" ofsx="175" ofsy="85" width="130" selected="no"
                    hint="Select it if you're currently in summer time daylight savings"
                    offline="enabled">
            Summer time</checkbox>
        </group>
        <group id="1100" ofsx="5" ofsy="110" width="310" height="25" name="Version info" border="1">
          <statictext ofsx="5" ofsy="10" width="80" align="right">GUI Version:</statictext>
          <statictext id="160" ofsx="90" ofsy="10" width="215" align="left"/>
        </group>
        <group id="1600" ofsx="5" ofsy="140" width="310" height="25" name="Face identification"
               border="1">
          <statictext ofsx="5" ofsy="10" width="80" align="right">Shots:</statictext>
          <editnum id="1601" ofsx="90" ofsy="10" width="80" align="left"
                   hint="Number of frames required to autoadjust camera exposition"
                   offline="enabled"/>
          <button id="1602" ofsx="240" ofsy="10" width="60" height="10" align="center" valign="center"
                  hint="Take a snapshot of the background, to help identifying faces"
                  offline="enabled">Shot Bg</button>
        </group>
        <group id="1200" ofsx="320" ofsy="5" width="275" height="100" name="Installed Printers"
               border="1">
          <listbox id="130" ofsx="5" ofsy="10" width="265" height="55"
                   hint="Select your default printer" offline="enabled">
            <cdef width="0" align="left" type="num4" visible="no">ID</cdef>
            <cdef width="19" align="left" type="atext" visible="yes">Branch</cdef>
            <cdef width="130" align="left" type="text" visible="yes">Printer name</cdef>
            <cdef width="100" align="left" type="text" visible="yes">Driver</cdef>
          </listbox>
          <button id="132" ofsx="5" ofsy="70" width="65" height="25" align="center" valign="center"
                  hint="View/Set default printer options" offline="enabled">Printer defaults</button>
          <button id="131" ofsx="77" ofsy="70" width="65" height="25" align="center" valign="center"
                  hint="Try to print a test page to selected printer" offline="enabled">
            Print test page</button>
          <button id="133" ofsx="145" ofsy="70" width="60" height="25" align="center" valign="center"
                  hint="Add a new printer" offline="enabled">Add a new printer</button>
          <button id="134" ofsx="210" ofsy="70" width="60" height="25" align="center" valign="center"
                  color="#d0a0a0"
                  hint="Remove selected printer" offline="enabled">Remove printer</button>
        </group>
        <group id="1300" ofsx="320" ofsy="110" width="275" height="25" name="Session preferences"
               border="1">
          <checkbox id="110" ofsx="5" ofsy="10" width="130" selected="yes"
                    hint="Save session state when closing in local disk"
                    offline="enabled">Save locally</checkbox>
          <checkbox id="111" ofsx="140" ofsy="10" width="130" selected="yes"
                    hint="Save session state when closing in server"
                    offline="disabled">Save to server</checkbox>
        </group>
        <group id="1400" ofsx="320" ofsy="140" width="275" height="25"
               name="Security and authentication" border="1">
          <statictext ofsx="5" ofsy="10" width="60" align="right">Certificate:</statictext>
          <edittext ofsx="70" ofsy="10" width="200" align="left" id="140" maxlen="24"
                    hint="Type here your site certificate" offline="enabled"/>
        </group>
        <group id="1500" ofsx="320" ofsy="170" width="275" height="40" name="Look and feel"
               border="1">
          <statictext ofsx="5" ofsy="10" width="130" align="right">Line cursor style:</statictext>
          <combobox id="150" ofsx="140" ofsy="10" width="130" align="left" lines="4"
                    hint="Select your preferred line (for listboxes, etc.) cursor shape"
                    offline="enabled">
            <li id="0" selected="yes"> <cd>Dashed</cd> </li>
            <li id="1" selected="no"> <cd>3D void</cd> </li>
            <li id="2" selected="no"> <cd>3D bar</cd> </li>
            <li id="3" selected="no"> <cd>Solid bar</cd> </li>
          </combobox>
          <statictext ofsx="5" ofsy="25" width="130" align="right">
            ListBox column extra width:</statictext>
          <editnum id="151" ofsx="140" ofsy="25" width="50" align="left"
                   hint="Pixels to add to every column to improve readability" maxlen="5"
                   offline="enabled"/>
        </group>
      </group>
      <button id="1" posx="5" posy="-25" width="60" height="20" align="center" valign="center"
              key="S" color="#c8f0c0" hint="Save settings to make them permanent"
              offline="enabled">Save</button>
      <button id="11" posx="70" posy="-25" width="60" height="20" align="center" valign="center"
              key="A" color="#c8c0f0" hint="Activate current settings\n(only for this session)"
              offline="enabled">Apply</button>
    </dialog>

    <dialog width="260" height="50" name="Services" version="0.1.1">
      <title>Services</title>
      <!-- Here we group different common services: Instant Messaging, Local cache manager, etc -->
      <!-- for now, this is still being defined -->
      <group distribution="V" posx="5" posy="5" width="600" height="455">
        <group id="1000" ofsx="5" ofsy="0" width="330" height="65" border="2" name="Cache status">
          <listbox id="150" ofsx="5" ofsy="10" width="320" height="50"
                   hint="Status and events related to local data cache" offline="enabled">
            <cdef width="85" align="right" type="uDateTime" visible="yes">At</cdef>
            <cdef width="40" align="center" type="text" visible="yes">Obj</cdef>
            <cdef width="190" align="left" type="text" visible="yes">Message/Event</cdef>
          </listbox>
        </group>
        <group id="1100" ofsx="5" ofsy="70" width="330" height="95" border="2"
               name="Resource window list">
          <listbox id="1101" ofsx="5" ofsy="10" width="320" height="80"
                   hint="Current application and windows list present in loaded resources"
                   offline="enabled">
            <cdef width="15" align="left" type="text" visible="yes"></cdef>
            <cdef width="80" align="left" type="atext" visible="yes">Name</cdef>
            <cdef width="40" align="left" type="text" visible="yes">Version</cdef>
            <cdef width="30" align="center" type="text" visible="yes">Kind</cdef>
            <cdef width="55" align="center" type="text" visible="yes">Script</cdef>
            <cdef width="0" align="left" type="num4" visible="no">aTag</cdef>
            <cdef width="60" align="left" type="text" visible="yes">File</cdef>
          </listbox>
        </group>
        <group id="1200" ofsx="5" ofsy="170" width="330" height="95" border="2"
               name="Local file list update state">
          <listbox id="1201" ofsx="5" ofsy="10" width="320" height="80"
                   hint="Current local file list and updated status" offline="enabled">
            <cdef width="0" align="left" type="num4" visible="no">ID</cdef>
            <cdef width="0" align="left" type="num4" visible="no">ReqSt</cdef>
            <cdef width="10" align="center" type="num4" visible="yes">FT</cdef>
            <cdef width="80" align="left" type="text" visible="yes">File Name</cdef>
            <cdef width="35" align="right" type="num4" subtype="2048" visible="yes">Size</cdef>
            <cdef width="80" align="right" type="uDateTime" subtype="0" visible="yes">
              Last Modif.</cdef>
            <cdef width="45" align="center" type="text" visible="yes">Status</cdef>
            <cdef width="120" align="left" type="text" visible="yes">Destination</cdef>
          </listbox>
        </group>
        <group id="1300" ofsx="5" ofsy="270" width="330" height="95" border="2"
               name="Libraries details">
          <listbox id="1301" ofsx="5" ofsy="10" width="320" height="80"
                   hint="Current list of included libraries" offline="enabled">
            <cdef width="0" align="left" type="num4" visible="no">ID</cdef>
            <cdef width="70" align="left" type="text" visible="yes">Library name</cdef>
            <cdef width="40" align="right" type="text" visible="yes">Version</cdef>
            <cdef width="130" align="right" type="text" visible="yes">Build</cdef>
          </listbox>
        </group>
      </group>
    </dialog>

    <!-- Common tools -->

    <dialog width="260" height="70" name="PrintOpt" version="0.1.0">
      <title>Printer Options</title>
      <group distribution="V" posx="5" posy="5" spacing="5">
        <group distribution="H">
          <group id="1000" distribution="H" name="Color Format" border="1">
            <radio id="101" gid="1001" width="65" align="left" selected="yes"
                   hint="Black and White or Gray Scale" offline="enabled">B/W</radio>
            <radio id="102" gid="1001" width="65" align="left" enabled="no"
                   hint="Full color" offline="enabled">Color</radio>
          </group>
          <group id="1100" distribution="H" name="Orientation" border="1">
            <radio id="111" gid="1101" width="95" align="left" selected="yes"
                   hint="Vertical orientation" offline="enabled">Portrait (vertical)</radio>
            <radio id="112" gid="1101" width="95" align="left" enabled="no"
                   hint="Horizontal orientation" offline="enabled">Landscape (horizontal)</radio>
          </group>
        </group>
        <group distribution="H">
          <group id="1200" distribution="H" name="Page size" border="1">
            <combobox id="121" width="135" lines="3" hint="Select your favourite page size"
                      offline="enabled"/>
          </group>
          <group id="1300" distribution="H" name="Printer fonts" border="1">
            <listbox id="131" width="195" height="50"
                     hint="This is just an informative list of available fonts" offline="enabled">
              <cdef width="0" align="left" type="num4" visible="no">fID</cdef>
              <cdef width="120" align="left" type="text" visible="yes">Name</cdef>
            </listbox>
          </group>
        </group>
        <group distribution="H">
          <button id="1" width="60" height="20" align="center" valign="center" key="K" color="#c8f0c0"
                  hint="Validate changes" offline="enabled">OK (set)</button>
          <button id="2" width="60" height="20" align="center" valign="center" key="C" color="#c8c0f0"
                  hint="Cancel any change" offline="enabled">Cancel</button>
        </group>
      </group>
    </dialog>

    <dialog width="260" height="70" name="PrintList" version="0.1.0">
      <title>Available Printers</title>
      <group distribution="V" posx="5" posy="5" spacing="5">
        <group id="1000" distribution="H" name="Printer list" border="1">
          <listbox id="101" width="250" height="60" hint="These are the locally detected printers"
                   offline="enabled">
            <cdef width="0" align="left" type="num4" visible="no">pID</cdef>
            <cdef width="19" align="left" type="atext" visible="yes">Branch</cdef>
            <cdef width="130" align="left" type="text" visible="yes">Printer name</cdef>
            <cdef width="100" align="left" type="text" visible="yes">Driver</cdef>
          </listbox>
        </group>
        <group id="1100" distribution="H" name="Available Printer Definitions" border="1">
          <combobox id="111" width="250" lines="5" hint="PPD to use for selected printer"
                    offline="enabled"/>
        </group>
        <group distribution="H">
          <button id="1" width="60" height="20" align="center" valign="center" key="K" color="#c8f0c0"
                  hint="Get selected printer and add it to our printer list" offline="enabled">
            Add printer</button>
          <button id="2" width="60" height="20" align="center" valign="center" key="C" color="#c8c0f0"
                  hint="Close this selection overlay" offline="enabled">Close</button>
        </group>
      </group>
    </dialog>

    <dialog width="260" height="70" name="Print" version="0.1.0">
      <title>Print Job</title>
      <group distribution="V" posx="5" posy="5" spacing="5">
```

```xml
      <group id="1300" distribution="H" name="Print job" border="1">
        <statictext id="130" width="250" align="left"/>
      </group>
      <group distribution="H">
        <group distribution="V">
          <group id="1400" distribution="H" name="Printer list" border="1">
            <listbox id="140" width="250" height="60" hint="Choose printer to print to"
                     offline="enabled">
              <cdef width="0" align="left" type="num4" visible="no">pID</cdef>
              <cdef width="19" align="left" type="atext" visible="yes">Branch</cdef>
              <cdef width="130" align="left" type="text" visible="yes">Printer name</cdef>
              <cdef width="100" align="left" type="text" visible="yes">Driver</cdef>
            </listbox>
          </group>
          <group id="1100" distribution="H" name="Orientation" border="1">
            <radio id="111" gid="1101" width="123" align="left" selected="yes"
                   hint="Vertical orientation" offline="enabled">Portrait (vertical)</radio>
            <radio id="112" gid="1101" width="123" align="left" enabled="no"
                   hint="Horizontal orientation" offline="enabled">Landscape (horizontal)</radio>
          </group>
          <group id="1500" distribution="H" name="Pages" border="1">
            <radio id="151" gid="1501" width="60" align="left" selected="yes"
                   hint="Print the whole document" offline="enabled">All</radio>
            <radio id="152" gid="1501" width="60" align="right" enabled="no"
                   hint="Print only selected page range or list" offline="enabled">Selected:</radio>
            <edittext id="153" width="120" align="left" maxlen="32"
                      hint="Type here your desired page range (1-3) or list (2,3,5)"
                      offline="enabled"/>
          </group>
        </group>
        <group distribution="V">
          <group distribution="H">
            <group id="1000" distribution="H" name="Color Format" border="1">
              <radio id="101" gid="1001" width="65" align="left" selected="yes"
                     hint="Black and White or Gray Scale" offline="enabled">B/W</radio>
              <radio id="102" gid="1001" width="65" align="left" enabled="no"
                     hint="Full color" offline="enabled">Color</radio>
            </group>
          </group>
          <group distribution="H">
            <group id="1200" distribution="H" name="Page size" border="1">
              <combobox id="121" width="135" lines="3" hint="Select your favourite page size"
                        offline="enabled"/>
            </group>
          </group>
          <group distribution="H">
            <group id="1600" distribution="H" name="Duplex" border="1">
              <radio id="161" gid="1601" width="65" align="left" selected="yes"
                     hint="Print only on one side" offline="enabled">No (single)</radio>
              <radio id="162" gid="1601" width="65" align="left" enabled="no"
                     hint="Print on both sides of paper" offline="enabled">Yes (2 sides)</radio>
            </group>
          </group>
        </group>
      </group>
    </group>
    <button id="100" posx="-65" posy="5" width="60" height="20" align="center" valign="center"
            enabled="no" key="V" color="#c8c0f0" hint="Show a print preview"
            offline="enabled">Preview</button>
    <button id="1" posx="-65" posy="30" width="60" height="20" align="center" valign="center"
            enabled="no" key="P" color="#c8f0c0" hint="Send job to selected printer (print)"
            offline="enabled">Print</button>
    <button id="110" posx="-65" posy="55" width="60" height="20" align="center" valign="center"
            enabled="no" key="P" color="#c8f0c0" offline="enabled"
            hint="Select a file name and send job to it (print to file)">to File</button>
    <button id="2" posx="-65" posy="80" width="60" height="20" align="center" valign="center"
            key="C" color="#c8c0f0" hint="Don't print at all" offline="enabled">Close</button>
  </dialog>

  <dialog width="260" height="70" name="PrintPreviewV" version="0.1.0">
    <title>Print Preview</title>
    <group distribution="H" posx="5" posy="5" spacing="5">
      <group id="1000" distribution="H" name="Page preview" border="1">
        <image id="100" width="198" height="281"/>
      </group>
      <group id="1100" distribution="V">
        <button id="201" width="80" height="20" align="center" valign="center" enabled="no"
                hint="See previous page" offline="enabled">Previous Page</button>
        <button id="202" width="80" height="20" align="center" valign="center" enabled="no"
```

```
                hint="See next page" offline="enabled">Next Page</button>
        <button id="2" width="80" height="20" align="center" valign="center"
                hint="Close this preview" offline="enabled">Close</button>
        <group distribution="H">
          <statictext width="30" align="right">Page:</statictext>
          <statictext id="110" width="45" align="left"/>
        </group>
      </group>
    </group>
  </dialog>

  <dialog width="260" height="70" name="PrintPreviewH" version="0.1.0">
    <title>Print Preview</title>
    <group distribution="V" posx="5" posy="5" spacing="5">
      <group id="1000" distribution="H" name="Page preview" border="1">
        <image id="100" width="281" height="198"/>
      </group>
      <group id="1100" distribution="H">
        <button id="201" width="80" height="20" align="center" valign="center" enabled="no"
                hint="See previous page" offline="enabled">Previous Page</button>
        <button id="202" width="80" height="20" align="center" valign="center" enabled="no"
                hint="See next page" offline="enabled">Next Page</button>
        <button id="2" width="80" height="20" align="center" valign="center"
                hint="Close this preview" offline="enabled">Close</button>
      </group>
    </group>
  </dialog>

  <dialog width="260" height="180" name="FileNav" version="0.1.0">
    <title>File browser</title>
    <group distribution="V" posx="5" posy="5" spacing="5">
      <group id="1000" distribution="H" spacing="5" name="File list" border="1">
        <group distribution="V" spacing="5">
          <group distribution="H" spacing="5">
            <statictext width="55" align="right">Path:</statictext>
            <edittext id="100" width="395" align="left" maxlen="256"
                      hint="Current path (may include file name)" offline="enabled"/>
          </group>
          <group distribution="H" spacing="5">
            <button id="102" width="80" height="15" align="center" valign="center" enabled="yes"
                    hint="Go to parent dir" offline="enabled">Up</button>
          </group>
          <group distribution="H" spacing="5">
            <listbox id="101" width="400" height="115" hint="Files" offline="enabled">
              <cdef width="0" type="num4" visible="no">Mode</cdef>
              <cdef width="10" type="icon" visible="yes"/>
              <cdef width="180" align="left" type="text" visible="yes">Name</cdef>
              <cdef width="70" align="right" type="num4" subtype="2048" visible="yes">Size</cdef>
              <cdef width="90" align="right" type="uDateTime" subtype="0"
                    visible="yes">Modified</cdef>
            </listbox>
            <group distribution="V" spacing="5">
              <statictext width="100" align="left">Preview:</statictext>
              <image id="110" width="100" height="100"/>
            </group>
          </group>
          <group distribution="H" spacing="5">
            <button id="1" width="80" height="20" align="center" valign="center" enabled="yes"
                    img="ok" imgw="15" imgh="15"
                    hint="Get current file (and exit)" offline="enabled">Get</button>
            <button id="2" width="80" height="20" align="center" valign="center" enabled="yes"
                    img="cancel" imgw="15" imgh="15"
                    hint="Exit with no selection" offline="enabled">Cancel</button>
          </group>
        </group>
      </group>
    </group>
  </dialog>

  <dialog width="260" height="180" name="DateRange" version="0.1.0">
    <title>Dates range</title>
    <group distribution="H" posx="5" posy="5" spacing="5">
      <group distribution="V" spacing="5">
        <group id="1000" distribution="H" spacing="5" name="Current range" border="1">
          <statictext width="40" align="right">From:</statictext>
          <statictext id="101" width="80" align="left"/>
          <statictext width="35" align="right">To:</statictext>
          <statictext id="102" width="80" align="left"/>
        </group>
```

```xml
        <group id="1100" distribution="V" spacing="5" border="1">
          <group distribution="H" spacing="5">
            <button id="110" width="80" height="20" align="center" valign="center" enabled="yes"
                    hint="1 day range" offline="enabled">Day</button>
            <button id="111" width="80" height="20" align="center" valign="center" enabled="yes"
                    hint="1 week range" offline="enabled">Week</button>
            <button id="112" width="80" height="20" align="center" valign="center" enabled="yes"
                    hint="1 month range" offline="enabled">Month</button>
          </group>
          <group distribution="H" spacing="5">
            <button id="113" width="80" height="20" align="center" valign="center" enabled="yes"
                    hint="3 month range" offline="enabled">Quarter</button>
            <button id="114" width="80" height="20" align="center" valign="center" enabled="yes"
                    hint="6 month range" offline="enabled">Semester</button>
            <button id="115" width="80" height="20" align="center" valign="center" enabled="yes"
                    hint="1 year range" offline="enabled">Year</button>
          </group>
        </group>
        <group id="1200" distribution="H" spacing="5" border="1">
          <button id="121" width="80" height="20" align="center" valign="center" enabled="yes"
                  hint="Yesterday, last week, last month..." offline="enabled">Previous</button>
          <button id="120" width="80" height="20" align="center" valign="center" enabled="yes"
                  hint="Today, this week, this month..." offline="enabled">Current</button>
          <button id="122" width="80" height="20" align="center" valign="center" enabled="yes"
                  hint="Tomorrow, next week, next month..." offline="enabled">Next</button>
        </group>
      </group>
      <group distribution="V" spacing="5">
        <button id="1" width="80" height="20" align="center" valign="center" enabled="yes"
                img="ok" imgw="15" imgh="15"
                hint="Set current date range (and exit)" offline="enabled">Ok</button>
        <button id="2" width="80" height="20" align="center" valign="center" enabled="yes"
                img="cancel" imgw="15" imgh="15"
                hint="Exit with no change at all" offline="enabled">Cancel</button>
      </group>
    </group>
  </dialog>

  <dialog width="260" height="70" name="ExitConfirmation" color="#ffd0c8" version="0.1.0">
    <title>Confirmation to exit</title>
    <statictext ofsx="10" ofsy="20" width="300" height="10" align="center">
      Are you sure you want to exit (close) the whole application?</statictext>
    <statictext id="100" ofsx="10" ofsy="40" width="300" height="10" align="center" />
    <statictext id="101" ofsx="10" ofsy="55" width="300" height="10" align="center" />
    <button id="1" ofsx="35" ofsy="75" width="120" height="20" align="center" valign="center"
            key="K" color="#c8f0c0"
            img="ok" imgw="15" imgh="15" hint="Yes, close the whole application" offline="enabled">
      Yes, close it all</button>
    <button id="2" ofsx="165" ofsy="75" width="120" height="20" align="center" valign="center"
            color="#f0c8c0" img="cancel" imgw="15" imgh="15"
            hint="No, just return back to application" offline="enabled">
      No, get back to application</button>
  </dialog>

  <dialog width="260" height="70" name="Confirmation" color="#ffd0c8" version="0.1.0">
    <title>Confirmation required</title>
    <statictext id="100" ofsx="10" ofsy="30" width="300" align="center" />
    <statictext id="101" ofsx="10" ofsy="45" width="300" height="40" align="center"
                valign="center" />
    <button id="1" ofsx="33" ofsy="95" width="120" height="25" align="center" valign="center"
            key="K" color="#c8f0c0" img="ok" imgw="15" imgh="15"
            hint="Acknowledge, click this button if agree with the question" offline="enabled">
      Yes, I agree</button>
    <button id="2" ofsx="158" ofsy="95" width="120" height="25" align="center" valign="center"
            color="#f0c8c0" img="cancel" imgw="15" imgh="15"
            hint="No, just return back to the application" offline="enabled">
      No, get back to application</button>
  </dialog>

  <!-- -->
  <!-- You may add your own windows (for main application) here but see note at the beginning -->
  <!-- -->

</application>
```

System management sub-application

This is an example of a file with a couple of windows definition, including complete script code. Note that it also uses specific utool functions for some tasks to improve efficiency (such functions are also in this document, in the utool example):

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE application SYSTEM "dialog.dtd">
<application name="System" windows="sys_files" color="#a0d8d8">
  <title>System</title>
  <dialog width="260" height="350" name="sys_files" color="#a0d8d8" version="0.1.0">
    <title>Files Management</title>
    <group distribution="V" posx="5" posy="5" width="600" height="450">
      <group id="1000" posx="10" posy="5" width="580" height="115" border="1" name="Patterns list">
        <listbox id="1009" ofsx="5" ofsy="10" width="505" height="100"
                 hint="Select a pattern to work on it">
          <cdef width="15" align="right" type="id32" visible="yes">idx</cdef>
          <cdef width="0" align="left" type="id32" visible="no">kind</cdef>
          <cdef width="45" align="left" type="text" visible="yes">Kind</cdef>
          <cdef width="0" align="left" type="id32" visible="no">arch</cdef>
          <cdef width="45" align="left" type="text" visible="yes">Arch</cdef>
          <cdef width="50" align="left" type="text" visible="yes">Extensions</cdef>
          <cdef width="200" align="left" type="text" visible="yes">Base path</cdef>
        </listbox>
        <button id="1098" ofsx="515" ofsy="55" width="60" height="25"
                hint="Create a new pattern based on current"
                align="center" valign="center" imgw="15" imgh="15" img="new">Clone</button>
        <button id="1099" ofsx="515" ofsy="85" width="60" height="25" hint="Create a new pattern"
                align="center" valign="center" imgw="15" imgh="15" img="new">New</button>
      </group>
      <group id="1100" posx="10" posy="125" width="580" height="45" border="1" name="Current pattern">
        <statictext ofsx="410" ofsy="15" width="50" height="10" align="right">Index:</statictext>
        <statictext id="1101" ofsx="465" ofsy="15" width="50" height="10" align="left"/>
        <statictext ofsx="5" ofsy="15" width="50" height="10" align="right">Kind:</statictext>
        <combobox id="1102" ofsx="60" ofsy="15" width="80" height="10" hint="Kind of file"
                  lines="10"/>
        <statictext ofsx="145" ofsy="15" width="50" height="10" align="right">Arch:</statictext>
        <combobox id="1103" ofsx="200" ofsy="15" width="80" height="10" hint="Architecture"
                  lines="10"/>
        <statictext ofsx="5" ofsy="30" width="50" height="10" align="right">Extens:</statictext>
        <edittext id="1104" ofsx="60" ofsy="30" width="80" height="10" align="left"
                  hint="Extensions, comma separated with no preceeding dot (example: 'pgn,gif,jpg')"
                  maxlen="64"/>
        <statictext ofsx="145" ofsy="30" width="50" height="10" align="right">Path:</statictext>
        <edittext id="1105" ofsx="200" ofsy="30" width="135" height="10" align="left"
                  hint="Base path" maxlen="128"/>
        <button id="1199" ofsx="515" ofsy="15" width="60" height="25" hint="Save current pattern"
                align="center" valign="center" imgw="15" imgh="15" img="save">Save</button>
      </group>
      <group id="1200" posx="10" posy="175" width="580" height="275" border="1"
             name="Current local source files list">
        <listbox id="1209" ofsx="5" ofsy="10" width="505" height="260" multipleselect="yes"
                 hint="Select a file to work on it">
          <cdef width="15" align="right" type="id32" visible="yes">idx</cdef>
          <cdef width="25" align="right" type="id32" visible="yes">ID</cdef>
          <cdef width="0" align="left" type="id32" visible="no">kind</cdef>
          <cdef width="45" align="left" type="text" visible="yes">Kind</cdef>
          <cdef width="0" align="left" type="id32" visible="no">arch</cdef>
          <cdef width="40" align="left" type="text" visible="yes">Arch</cdef>
          <cdef width="10" align="center" type="id32" visible="yes">UpL</cdef>
          <cdef width="10" align="center" type="id32" visible="yes">Flg</cdef>
          <cdef width="50" align="left" type="text" visible="yes">FileName</cdef>
          <cdef width="100" align="left" type="text" visible="yes">Description</cdef>
          <cdef width="75" align="right" type="uDateTime" visible="yes">Local Modif.</cdef>
          <cdef width="75" align="right" type="uDateTime" visible="yes">DB Modif.</cdef>
          <cdef width="40" align="right" type="num4" subtype="2048" visible="yes">Local Sz</cdef>
          <cdef width="40" align="right" type="num4" subtype="2048" visible="yes">DB Sz</cdef>
          <cdef width="80" align="left" type="text" visible="yes">Path</cdef>
          <cdef width="0" align="left" type="id32" visible="no">chk</cdef>
          <cdef width="20" align="left" type="id32" visible="yes">dtaID</cdef>
        </listbox>
        <button id="1299" ofsx="515" ofsy="10" width="60" height="25"
                hint="Update source files list using current patterns" align="center" valign="center"
                imgw="15" imgh="15" img="buscar">Update</button>
        <button id="1297" ofsx="515" ofsy="40" width="60" height="25"
                hint="Get missing data from DB" align="center" valign="center"
                imgw="15" imgh="15" img="save">Get missing</button>
        <button id="1296" ofsx="515" ofsy="70" width="60" height="25"
```

```
            hint="Edit selected entry" align="center" valign="center"
            imgw="15" imgh="15" img="edit">Edit</button>
    <button id="1295" ofsx="515" ofsy="100" width="60" height="25"
            hint="Save local file list (does not upload anything)" align="center" valign="center"
            imgw="15" imgh="15" img="save">Save list</button>
    <button id="1298" ofsx="515" ofsy="245" width="60" height="25"
            hint="Upload files that need an update on the remote DB (synchronize)" align="center"
            valign="center" imgw="15" imgh="15" img="save">Upload</button>
  </group>
</group>
<script>
  <![CDATA[
  // Globals
  var   cpIx  = 0;          // Current pattern index
  var   cnti  = 0;          // Current # of pattern items
  var   cnfi  = 0;          // Current # of file items
  var   FEwID = 0;          // File Edition PopUp

  function GetKindName(kind)
  {
    switch (kind)
      {
      case 0: return "?";
      case 1: return "Hlp";
      case 2: return "DTD";
      case 3: return "Dialog";
      case 4: return "Lang. File";
      case 5: return "img/ap";
      case 6: return "img/sys";
      case 7: return "img/usr";
      case 8: return "img/wil";
      case 9: return "module";
      case 10: return "library";
      case 11: return "app";
      case 12: return "indep .app";
      }
  return "";
  }

  function GetArchName(kind)
  {
    switch (kind)
      {
      case 0: return "any";
      case 1: return "Linux 32b";
      case 2: return "Linux 64b";
      case 3: return "Win 32b";
      }
  return "";
  }

  function RefreshList()
  // Repaints pattern list
  {
    win.Clear(1009);
    cnti=utool.sysGetNumFP();
    var i=0;
    while (i<cnti)
      {
      var r=utool.sysGetFP(i);
      var s[0]=i;
      s[1]=r["kind"];
      s[2]=GetKindName(s[1]);
      s[3]=r["arch"];
      s[4]=GetArchName(s[3]);
      s[5]=r["ext"];
      s[6]=r["path"];
      win.LB.AddRow(1009,0,s);
      i++;
      }
    win.PaintItem(1009);
  }

  function RefreshFList()
  // Repaints files list
  {
    win.Clear(1209);
    cnfi=utool.sysGetNumFL();
    var i=0;
    while (i<cnfi)
      {
      var r=utool.sysGetFL(i);
      var s[0]=i;
      s[1]=r["id"];
      s[2]=r["kind"];
      s[3]=GetKindName(s[2]);
      s[4]=r["arch"];
      s[5]=GetArchName(s[4]);
      s[7]=r["flg"];
      s[8]=r["fn"];
      s[9]=r["desc"];
      s[10]=r["fmod"];
      s[11]=r["cmod"];
      s[6]=(s[10]>s[11]);
      s[12]=r["fsz"];
      s[13]=r["csz"];
      s[14]=r["path"];
      s[15]=r["chk"];
      s[16]=r["dta"];
      var flg=(s[6]!=0);
      win.LB.AddRow(1209,flg,s);
      i++;
      }
    win.PaintItem(1209);
  }

  function FillKindCB(iID)
  // Fills given ComboBox with 'kind' values
  {
    i=0;
    while (i<32)
      {
      var s=GetKindName(i);
      if (s=="")
         break;
      win.CB.AddRow(iID,i,s);
      i++;
      }
  }

  function FillArchCB(iID)
  // Fills given ComboBox with 'arch' values
  {
    i=0;
    while (i<8)
      {
      var s=GetArchName(i);
      if (s=="")
         break;
      win.CB.AddRow(iID,i,s);
      i++;
      }
  }

  function win.Init()
  {
    utool.sysLoadFP();
    RefreshList();
    utool.sysLoadFL();
    RefreshFList();
    FillKindCB(1102);
    FillArchCB(1103);
  }

  function win.SelMod.i1009(row,col,flg)
  // Selection of a pattern
  {
    if (row==65535)
       return;
    var r=win.LB.GetRow(1009,row);
    cpIx=r[0];
    win.SetText(1101,cpIx);
    win.CB.SelByID(1102,r[1]);
    win.CB.SelByID(1103,r[3]);
    win.SetText(1104,r[5]);
    win.SetText(1105,r[6]);
    win.SetModiF(0);
  }
```

```
      function win.Act.i1098()
      // Clone
      {
        cpIx=cnti;
        win.SetText(1101,cpIx);
        win.SetModiF(0);
      }

      function win.Act.i1099()
      // New
      {
        cpIx=cnti;
        win.SetText(1101,cpIx);
        win.CB.SelByID(1102,0);
        win.CB.SelByID(1103,0);
        win.Clear(1104);
        win.Clear(1105);
        win.SetModiF(0);
      }

      function win.Act.i1199()
      // Save
      {
        utool.sysSetFP(cpIx,win.CB.GetSel(1102),win.CB.GetSel(1103),win.GetText(1105),win.GetText(1104
));
        win.SetModiF(0);
        win.SetText(1101,cpIx);
        RefreshList();
      }

      function win.Act.i1104()
      {
        win.SetModiF(1);
      }

      function win.Act.i1105()
      {
        win.SetModiF(1);
      }

      function win.SelMod.i1102(row,col,flg)
      {
        win.SetModiF(1);
      }

      function win.SelMod.i1103(row,col,flg)
      {
        win.SetModiF(1);
      }

      function win.Act.i1297()
      // Get missing data from DB
      {
        win.Msg("Wait","Synchronizing missing data from DB...");
        if (!query.Send(5,"_Files.ID>0","_Files.ID,._Dta.ID,.Desc"))
           {
           var res=query.Wait(5);
           if (res==0)
              {
              var nItm=query.GetNumItm(5,"_Files");
              var i=0;
              while (i<nItm)
                {
                var r=query.GetRec(5,"_Files",i);
                utool.sysSetDBFL(r["id"],r["desc"],r["_dta.id"]);
                i++;
                }
              }
           query.Free(5);
           }
        utool.sysGenFL();
        RefreshFList();
        win.Msg("Neutral","Ok.");
      }

      function win.Act.i1299()
      // Update file list
      {
```

```
  win.Msg("Wait","Checking local files...");
  utool.sysGenFL();
  RefreshFList();
  win.Msg("Neutral","Ok.");
}

function win.Act.i1298()
// Synchronize (upload)
{
  var n=win.LB.GetNRow(1209);
  var i=0;
  while (i<n)
    {
    if (win.LB.GetCell(1209,i,6))
       {  // Needs updating
       var r=win.LB.GetRow(1209,i);
       if (r[8] && (r["atr"] & 1))
          {
          win.LB.SetCRow(1209,0,r[0]);
          var fn=r[14]+"/"+r[8];
          var q="_Files.ID="+r[1]+",._kind="+r[2]+",._arch="+r[4];
          if (r[6]) // File content has changed: upload it too
             var b=system.LoadBB(fn);
          else      // Just update _Files record (not the heavy file content)
             b[0]=0;
          if (system.Dim(b)>0)
             {
             query.InitSave(30);
             var fn=query.AddSaveT(r[8]);
             q+=",._fn=<"+fn+">";
             var desc=query.AddSaveT(r[8]);
             if (desc)
                q+=",.Desc=<"+desc+">";
             q+=",._sz="+r[12]+",._chk="+r[15]+",._fTS=u"+r[10];
             if (system.GetType(b)=="block")
                { // Send file content too
                var bb=query.AddSaveB(b,r[16]);
                if (bb)
                   q+=",._Dta=<"+bb+">";
                }
             var bID=query.AddSaveR(q);
             win.Msg("Wait","Saving file "+r[8]+" to server...");
             var res=query.Save();
             if (!res)
                {
                query.Wait(30);
                var fID=query.GetSavedID(30,bID);
                if (fID)
                   r[1]=fID;
                }
             query.DoneSave();
             if (res)
                {
                win.Msg("Error","Failed saving profile to server");
                return;
                }
             }
          }
       // Finally, clear 'pending' status, selection, and update other cells
       r[6]=0;
       win.LB.SetRow(1209,i,0,r);
       win.PaintItem(1209);
       }
    i++;
    }
  // Finally, reload cache
  utool.sysReloadLocalCache("_Files");
  win.SetModiF(0);
  utool.sysSaveFL();
  win.Msg("Neutral","Reloading cache... please use Update button to resync");
}

function win.Act.i1296()
// Edit
{
  var cr=win.LB.GetCRow(1209);
  if (cr==65535)
     return;
  var r=win.LB.GetRow(1209,cr);
```

```
      EFwID=win.OpenPopUp("sys_edfile",1209);
      if (EFwID)
         {
         v=new Array();
         v[0]=cr;
         v[1]=r;
         win.SendVarMsg(EFwID,v);
         }
    }

    function win.Act.i1295()
    // Save list
    {
      utool.sysSaveFL();
      win.SetModiF(0);
    }

    function win.VarMsg(fwID,v)
    // Receive message with a variable
    {
      switch (fwID)
        {
        case EFwID:
          // Edit File PopUp: [0]=row index, [1]=array with row content
          if (v[0]<65535)
             {
             win.LB.SetRow(1209,v[0],1,v[1]);   // Update content, and select it if not already
             var r=v[1];
             utool.sysSetDBFL(r[1],r[9],0);
             win.PaintItem(1209);
             win.SetModiF(1);
             }
          break;
        }
    }

    ]]>
  </script>
</dialog>

<dialog width="540" height="350" name="sys_edfile" version="0.1.1">
  <title>Edit file</title>
  <statictext ofsx="5" ofsy="10" width="40" height="10" align="left">Name:</statictext>
  <statictext id="1050" ofsx="50" ofsy="10" width="100" height="10" align="left"/>

  <statictext ofsx="5" ofsy="25" width="60" height="10" align="left">Kind:</statictext>
  <statictext ofsx="70" ofsy="25" width="50" height="10" align="left">Arch:</statictext>
  <statictext ofsx="125" ofsy="25" width="200" height="10" align="left">Description:</statictext>
  <combobox id="1001" ofsx="5" ofsy="40" width="60" height="10" hint="Kind of file" lines="10" />
  <combobox id="1002" ofsx="70" ofsy="40" width="50" height="10" hint="Target architecture"
          lines="5" />
  <edittext id="1003" ofsx="125" ofsy="40" width="200" height="10"
          hint="File description (informative)" maxchr="200" maxlin="1"/>

  <button id="1099" ofsx="100" ofsy="60" width="60" height="25"
        hint="Save changes" color="#B0F0C0" align="center" valign="center"
        imgw="15" imgh="15" img="save">Ok</button>
  <button id="1098" ofsx="165" ofsy="60" width="60" height="25"
        hint="Close with no changes" align="center" valign="center"
        imgw="15" imgh="15" img="back">Cancel</button>
  <script>
    <![CDATA[
    // Globals
    var pwID  = 0;  // Parent window
    var rIx   = 0;  // Row index
    var rr    = new Array();  // Row content

    function win.Init()
    {
      sys_files.FillKindCB(1001);
      sys_files.FillArchCB(1002);
    }

    function win.Act.i1098()
    {
      win.CloseWin(win.GetMyWinID());
    }

    function win.Act.i1099()
```

```
      {
        rr[2]=win.CB.GetSel(1001);
        rr[3]=sys_files.GetKindName(rr[2]);
        rr[4]=win.CB.GetSel(1002);
        rr[5]=sys_files.GetArchName(rr[4]);
        rr[9]=win.GetText(1003);
        var b=new Array();
        b[0]=rIx;
        b[1]=rr;
        win.SendVarMsg(pwID,b);
        win.CloseWin(win.GetMyWinID());
      }

      function win.VarMsg(fwID,v)
      // Receive message with a variable
      {
        pwID=fwID;
        // [0]=rowIdx, [1]=row content (array)
        rIx=v[0];
        rr=v[1];
        win.SetText(1050,rr[8]);
        win.CB.SelByID(1001,rr[2]);
        win.CB.SelByID(1002,rr[4]);
        win.SetText(1003,rr[9]);
      }

      ]]>
    </script>
  </dialog>
</application>
```