RCXML Technical Specification

ver. 0.5

Rev.1.02

RCXML Elements List

Elements	Purpose
assign	Assign a variable a value
bookmark	Set bookmark
catch	Catch an event
check_mem	Check the system memory
check_space	Check the available space
clear	Clear one or more form item variables
delete_file	Selete a specific file
device	Declare an appliance
else	Used in <if> elements
elseif	Used in <if> elements
error	Catch an error event
exit	Exit a session
fetch	Preload a RCXML file
goto	Go to another session in the same or different document
if	Simple conditional logic
log	Generate a debug message
open_file	Open a specific file
power_off	Power off
power_on	Power on
rcxml	Top-level element in each RCXML document
record	Start recording immediately
remane_file	Rename a specific file
save_file	Save a file
script	Specify a block of ECMAScript client-side scripting logic
send	Send the result to the remote device
session	Begin a session
set_book	Reserve the recording program
set_channel	Set channel
set_speed	Set speed

set_temperature	Set temperature
set_volume	Set volume
start_operation	Declare an operation
var	Declare a variable

1.Control flow and scripting

1.1 Variables and Expressions

RCXML variables are in all respects equivalent to ECMAScript variables: they are part of the same variable space. RCXML variables can be used in a <script> just as variables defined in a <script> can be used in RCXML. Declaring a variable using <var> is equivalent to using a 'var' statement in a <script> element. <script> can also appear everywhere that <var> can appear. RCXML variables are also declared by session items.

The variable naming convention is as in ECMAScript, but names beginning with the underscore character ("_") and names ending with a dollar sign ("$") are reserved for internal use. RCXML variables, including session item variables, must not contain ECMAScript reserved words. They must also follow ECMAScript rules for referential correctness. For example, variable names must be unique and their declaration must not include a dot - "var x.y" is an illegal declaration in ECMAScript. Variable names which violate naming conventions or ECMAScript rules cause an error.

1.1.1 Declaring Variables

Variables are declared by <var> elements:

```
<var name="home_phone"/>
<var name="pi" expr="3.14159"/>
<var name="city" expr="'Sacramento'"/>
```

Variables declared without an explicit initial value are initialized to the ECMAScript undefined value. Variables must be declared before being used either in RCXML or ECMAScript. Use of an undeclared variable results in an ECMAScript error which is thrown as an error. Variables declared using "var" in ECMAScript can be used in RCXML, just as declared RCXML variables can be used in ECMAScript.

In a session, the variables declared by <var> and those declared by session items are initialized when the session is entered. The initializations are guaranteed to take place in document order, so that this, for example, is legal:

```
<?xml version="1.0" encoding="UTF-8"?>
<rcxml version="1.0">
<device appliance="tv">
   <session id="s001">
      <var name="one" expr="1"/>
      <var name="two" expr="one+1"/>
      <start_operation id="op01">
        <power_on/>
      </start_operation>
  </session>
</device>
</rcxml>
```

When the user visits this <session>, the session's initialization first declares the variable one and sets its value to 1. Then it declares the variable two and gives it the value 2.

1.1.2 Variable Scopes

RCXML uses an ECMAScript scope chain to allow variables to be declared at different levels of hierarchy in an application. For instance, a variable declared at document scope can be referenced anywhere within that document, whereas a local variable declared in a catch element is only available within that catch element. In order to preserve these scoping semantics, all ECMAScript variables must be declared. Use of an undeclared variable results in an ECMAScript error which is thrown as an error.semantic.

1.1.3 Referencing Variables

Variables are referenced in cond and expr attributes:

```
   <if cond="city == 'LA'">
      <assign name="city" expr="'Los Angeles'"/>
   <elseif cond="city == 'Philly'"/>
```

```
<assign name="city" expr="'Philadelphia'"/>
<elseif cond="city =='Constantinople'"/>
    <assign name="city" expr="'Istanbul'"/>
</if>
<assign name="var1" expr="var1 +  1"/>
<if cond="i &gt; 1">
    <assign name="i" expr="i-1"/>
</if>
```

The expression language used in cond and expr is precisely ECMAScript. Note that the cond operators "<", "<=", and "&&" must be escaped in XML (to "<" and so on).

Variable references match the closest enclosing scope according to the scope chain given above. You can prefix a reference with a scope name for clarity or to resolve ambiguity. For instance to save the value of a variable associated with one of the fields in a form for use later on in a document:

```
<assign name="document.ssn" expr="dialog.ssn"/>
```

If the application root document has a variable x, it is referred to as application.x in non-root documents, and either application.x or document.x in the application root document. If the document does not have a specified application root and has a variable x, it is referred to as either application.x or document.x in the document.

1.2 Event Handling

The platform throws events when the system does not respond, doesn't respond in a way that the application understands, requests help, etc. The system throws events if it finds a semantic error in a RCXML document. Events are identified by character strings.

Each element in which an event can occur has a set of catch elements, which include:

-. <catch>

-. <error>

-. <noresponse>

1.3 Executable Content

Executable content refers to a block of procedural logic. Such logic appears in:

- .The <start_opearion> session item.

Executable elements are executed in document order in their block of procedural logic. If an executable element generates an error, that error is thrown immediately. Subsequent executable elements in that block of procedural logic are not executed.

This section covers the elements that can occur in executable content

1.3.1 var element

This element declares a variable. It can occur in executable content or as a child of <session> or <rcxml>. Examples:

```
<var name="phone" expr="'6305551212'"/>
<var name="y" expr="document.z+ 1"/>
```

If it occurs in executable content, it declares a variable in the anonymous scope associated with the enclosing <block>, <filled>, or catch element. This declaration is made only when the <var> element is executed. If the variable is already declared in this scope, subsequent declarations act as assignments, as in ECMAScript.

If a <var> is a child of a <form> element, it declares a variable in the dialog scope of the <form>. This declaration is made during the form's initialization phase as described in Section 2.1.6.1. The <var> element is not a form item, and so is not visited by the Form Interpretation Algorithm's main loop.

If a <var> is a child of a <rcxml> element, it declares a variable in the document scope; and if it is the child of a <rcxml> element in a root document then it also declares the variable in the application scope. This declaration is made when the document is initialized; initializations happen in document order.

Attributes of <var> include:

name	The name of the variable that will hold the result. Unlike the name attribute of <assign> element (Section 1.3.2), this attribute must not specify a variable with a scope prefix (if a variable is specified with a scope prefix, then an error.semantic event is thrown). The scope in which the variable is defined is determined from the position in the document at which the element is declared.
expr	The initial value of the variable (optional). If there is no expr attribute, the variable retains its current value, if any. Variables start out with the ECMAScript value undefined if they are not given initial values.

1.3.2 assign element

The <assign> element assigns a value to a variable:

```
<assign name="flavor" expr="'chocolate'"/>
```

```
<assign name="document.mycost" expr="document.mycost+ 14"/>
```

It is illegal to make an assignment to a variable that has not been explicitly declared using a <var> element or a var statement within a <script>. Attempting to assign to an undeclared variable causes an error.semantic event to be thrown.

Note that when an ECMAScript object, e.g. "obj", has been properly initialized then its properties, for instance "obj.prop1", can be assigned without explicit declaration (in fact, an attempt to declare ECMAScript object properties such as "obj.prop1" would result in an error.semantic event being thrown).

Attributes include:

name	The name of the variable being assigned to. As specified in Section 1.1.2, the corresponding variable must have been previously declared otherwise an error event is occurred. By default, the scope in which the variable is resolved is the closest enclosing scope of the currently active element. To remove ambiguity, the variable name may be prefixed with a scope name as described in Section 1.1.3.
expr	The new value of the variable

1.3.3 clear element

The <clear> element resets one or more variables, including form items. For each specified variable name, the variable is resolved relative to the current scope according to Section 1.1.3 (to remove ambiguity, each variable name in the namelist may be prefixed with a scope name). Once a declared variable has been identified, its value is assigned the ECMAScript undefined value. In addition, if the variable name corresponds to a form item, then the form item's prompt counter and event counters are reset.

For example:

```
<clear namelist="city state zip"/>
```

The attribute is:

namelist	The list of variables to be reset; this can include variable names other than form items. If an undeclared variable is referenced in the namelist, then an error.semantic is thrown (Section 1.1.1). When not specified, all form items in the current form are cleared.

1.3.4 if, else, elseif elements

The <if> element is used for conditional logic. It has optional <else> and <elseif> elements.

```
<if cond="total > 1000">
  <prompt>This is way too much to spend.</prompt>
  <throw event="com.xyzcorp.acct.toomuchspent"/>
</if>
<if cond="amount &lt; 29.95">
  <assign name="x" expr="amount"/>
<else/>
<assign name="x" expr="29.95"/>
</if>
<if cond="flavor == 'vanilla'">
  <assign name="flavor_code" expr="'v'"/>
<elseif cond="flavor == 'chocolate'"/>
  <assign name="flavor_code" expr="'h'"/>
<elseif cond="flavor == 'strawberry'"/>
  <assign name="flavor_code" expr="'b'"/>
<else/>
  <assign name="flavor_code" expr="'?'"/>
</if>
```

1.3.5 goto element

The <goto> element is used to:

-. transition to another form item in the current form,

-. transition to another dialog in the current document, or

-. transition to another document.

To transition to another form item, use the nextitem attribute, or the expritem attribute if the form item name is computed using an ECMAScript expression:

```
<goto nextitem="ssn_confirm"/>
<goto expritem="(type==12)? 'ssn_confirm' : 'reject'"/>
```

To go to another dialog in the same document, use next (or expr) with only a URI fragment:

```
<goto next="#another_dialog"/>
<goto expr="'#' + 'another_dialog'"/>
```

To transition to another document, use next (or expr) with a URI:

```
<goto next="http://flight.example.com/reserve_seat"/>
```

```
<goto next="./special_lunch#wants_vegan"/>
```

The URI may be absolute or relative to the current document. You may specify the starting dialog in the next document using a fragment that corresponds to the value of the id attribute of a dialog. If no fragment is specified, the first dialog in that document is chosen.

Note that transitioning to another dialog in the current document causes the old dialog's variables to be lost, even in the case where a dialog is transitioning to itself. Transitioning to another document using an absolute or relative URI will likewise drop the old document level variables, even if the new document is the same one that is making the transition. However, document variables are retained when transitioning to an empty URI reference with a fragment identifier. For example, the following statements behave differently in a document with the URI http://someco.example.com/index.xml:

```
<goto next="#foo"/>
<goto next="http://someco.example.com/index.xml#foo"/>
```

According to [RFC2396], the fragment identifier (the part after the '#') is not part of a URI and transitioning to empty URI references plus fragment identifiers should never result in a new document fetch. Therefore "#foo" in the first statement is an empty URI reference with a fragment identifier and document variables are retained. In the second statement "#foo" is part of an absolute URI and the document variables are lost. If you want data to persist across multiple documents, store data in the application scope.

The dialog to transition to is specified by the URI reference in the <goto>'s next or expr attribute (see [RFC2396]). If this URI reference contains an absolute or relative URI, which may include a query string, then that URI is fetched and the dialog is found in the resulting document.

If the URI reference contains only a fragment (i.e., no absolute or relative URI), then there is no fetch: the dialog is found in the current document.

The URI reference's fragment, if any, names the dialog to transition to. When there is no fragment, the dialog chosen is the lexically first dialog in the document.

If the form item, dialog or document to transition to is not valid (i.e. the form item, dialog or document does not exist), an error.badfetch must be thrown. Note that for errors which occur during a dialog or document transition, the scope in which errors are handled is platform specific. For errors which occur during form item transition, the event is handled in the dialog scope.

Attributes of <goto> are:

next	The URI to which to transition.
expr	An ECMAScript expression that yields the URI.
nextitem	The name of the next form item to visit in the current form.
expritem	An ECMAScript expression that yields the name of the next form item to visit.

Exactly one of "next", "expr", "nextitem" or "expritem" must be specified; otherwise, an error.badfetch event is thrown.

1.3.6 exit element

Returns control to the interpreter context which determines what to do next.

<exit/>

This element differs from <return> in that it terminates all loaded documents, while <return> returns from a <subdialog> invocation. If the <subdialog> caused a new document (or application) to be invoked, then <return> will cause that document to be terminated, but execution will resume after the <subdialog>.

Note that once <exit> returns control to the interpreter context, the interpreter context is free to do as it wishes. It may play a top level menu for the user, drop the call, or transfer the user to an operator, for example.

Attributes include:

expr	An ECMAScript expression that is evaluated as the return value (e.g. "0", "'oops!'", or "field1").
namelist	Variable names to be returned to interpreter context. The default is to return no variables; this means the interpreter context will receive an empty ECMAScript object. If an undeclared variable is referenced in the namelist, then an error.semantic is thrown.

Exactly one of "expr" or "namelist" may be specified; otherwise, an error.badfetch event is thrown.

The <exit> element does not throw an "exit" event.

1.3.7 script element

The <script> element allows the specification of a block of client-side scripting language code, and is analogous to the [HTML] <SCRIPT> element. For example, this document has a script that computes a factorial.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<rcxml version="1.0">
    <script> <![CDATA[
        function factorial(n)
        {
          return (n <= 1)? 1 : n * factorial(n-1);
        }
    ]]> </script>
</rcxml>
```

A <script> element may occur in the <rcxml> and <form> elements, or in executable content. Scripts in the <rcxml> element are evaluated just after the document is loaded, along with the <var> elements, in document order. Scripts in the <session> element are evaluated in document order, along with <var> elements and form item variables, each time execution moves into the <session> element. A <script> element in executable content is executed, like other executable elements, as it is encountered.

The <script> element has the following attributes:

src	The URI specifying the location of the script, if it is external.
charset	The character encoding of the script designated by src. UTF-8 and UTF-16 encodings of ISO/IEC 10646 must be supported (as in [XML]) and other encodings, as defined in the [IANA], may be supported. The default value is UTF-8.

Either an "src" attribute or an inline script (but not both) must be specified; otherwise, an error.badfetch event is thrown.

The RCXML <script> element (unlike the [HTML] <SCRIPT> element) does not have a type attribute; ECMAScript is the required scripting language for RCXML.

Each <script> element is executed in the scope of its containing element; i.e., it does not have its own scope. This means for example that variables declared with var in the <script> element are declared in the scope of the containing element of the <script> element. (In ECMAScript terminology, the "variable object" becomes the current scope of the containing element of the <script> element).

Here is a time-telling service with a block containing a script that initializes time variables in the dialog scope of a form:

```
<?xml version="1.0" encoding="UTF-8"?>
<rcxml version="1.0">
  <device>
```

```
    <session>
      <var name="hours"/>
    <var name="minutes"/>
      <var name="seconds"/>
      <script>
        var d = new Date();
        hours = d.getHours();
        minutes = d.getMinutes();
        seconds = d.getSeconds();
       </script>
    </session>
  </device>
</rcxml>
```

The content of a <script> element is evaluated in the same scope as a <var> element (see 1.1.2 Variable Scopes and 1.3.1 VAR).
The ECMAScript scope chain (see section 10.1.4 in [ECMASCRIPT]) is set up so that variables declared either with <var> or inside <script> are put into the scope associated with the element in which the <var> or <script> element occurs.

All variables must be declared before being referenced by ECMAScript scripts, or by RCXML elements as described in Section 1.1.1.

1.3.8 log element

The <log> element allows an application to generate a logging or debug message which a developer can use to help in application development or post-execution analysis of application performance.

The <log> element may contain any combination of text (CDATA) and <value> elements. The generated message consists of the concatenation of the text and the string form of the value of the "expr" attribute of the <value> elements.

The manner in which the message is displayed or logged is platform-dependent. The usage of label is platform-dependent. Platforms are not required to preserve white space.

ECMAScript expressions in <log> must be evaluated in document order. The use of the <log> element should have no other side-effects on interpretation.

```
<log>The card number was <value expr="card_num"/></log>
```

The <log> element has the following attributes:

label	An optional string which may be used, for example, to indicate the purpose of the log
expr	An optional ECMAScript expression evaluating to a string.

1.4 Operation Content

Operation content is a collection for the elements of controlling device. These elements are child nodes of the element <start_operation> and define all function of RCXML.

1.4.1 check_mem element

The <check_mem> element returns the device memory's status. It can be used in the Device which has the Embedded OS.

1.4.2 check_space element

The <check_space> element returns the hard disk's space of device. It can be used in the Device which has the Embedded OS.

1.4.3 delete_file element

The <delete_file> element deletes the saved file in Device's hard disk. This element can be used in the device which has the digital recording equipment.

For Example;

```
<delete_file fname="kbs9news"/>
```

fname	Filename which will be deleted. If the file dosen't exist in the device, this element occurs error.

1.4.4 open_file element

The <open_file> element opens the saved file in Device's hard disk.

For Example;

```
< open_file fname="kbs9news"/>
```

fname	Filename which will be deleted. If the file dosen't exist in the device, this element occurs error.

1.4.5 power_off element

The <power_off> element turns off the device.

1.4.6 power_on element

The <power_off> element turns on the device.

1.4.7 record element

The <record> element starts recording immediately.

1.4.8 rename_file element

The <rename_file> element is changed the existing filename to new name.

For Example;

```
< rename_file oldname="untitle01" newname="kbs9news "/>
```

oldname	Filename which will be changed. If the file dosen't exist in the device, this element occurs error.
newname	New filename.

1.4.9 save_file element

The <save_file> element will save the recording file.

1.4.10 set_book element

The <set_book> element sets the scheduled recording.

For Example;

```
<set_book ch_no="11" date="20050110" stime="2100" duration="45m"
mode="sd"/>
```

This example shows the device will record the channel 11 for 45 minutes from 9 pm on Jan. 10th 2005.

ch_no	Channel No for recording
date	Scheduled date (yyyy/mm/dd)
stime	Scheduled time (hh/mm)
duration	Duration (should be shown in minutes)
mode	Recording mode. hd/sd/auto

1.4.11 set_channel element

The <set_channel> element will change the channel. This element has three different modes – auto, manual, reserve. The 'auto' mode will change the channel up and down automatically. If user press the 'up' button, the channel will be changed to upper channel from the current channel. The upper channel follows the device setting. The 'manual' mode changes the channel by user's input. And the 'reserve' mode will change the channel by scheduling. This mode's usage is shown on the below example.

For Example:

```
<set_ channel mode="reserve" no="11" datetime="20050110 2100"/>
```

This example shows how to use the 'reserve' mode in <set_channel>. Once this element set, the channel will be automatically changed to channel 11 on 9 pm Jan. 10th. If at that moment user watches channel 11, nothing happens.

mode	One of auto, manual or reserve.
dir	Either up or down. Only using in auto mode.
no	Channel no to be changed.
datetime	Scheduled time (yyyy/mm/dd hh/mm).

1.4.12 set_speed element

The <set_speed> element controls the speed. It will be applied to the playing devices such as VCR, PVR and tape recorder. The maximum and the minimum value follow the device's setting.

For Example:

```
<set_speed dir="fastforward"/>
```

dir	Either fast forward or rewind

1.4.13 set_temperature element

The <set_temperature> element controls the temperature. Oven, Gas Range, Air-Conditioner can have this element. The maximum and the minimum value follow the device's setting.

For Example:

```
<set_temperature mode="auto" dir="up"/>
```

mode	Either auto or manual.
dir	Either up or down. Only applying in auto mode.
temp	The temperature to be set.

1.4.14 set_volume element

The <set_volume> element controls the volume.

For Example:

```
<set_volume dir="mute"/>
```

dir	One of up, down or mute.

1.4.15 bookmark element

The <bookmark> element inserts the bookmark to the recorded file.

For Example:

```
<bookmark mode="set" fname="mbc9news" bname="Diplomatic goal: correct Japan history" starttime="05m00s" endtime="07m30s"/>
```

This example shows that insert bookmark to the mbc9news file. The bookmark should be named 'Diplomatic goal : correct Japan History' and duration will be two minutes and thirty seconds.

mode	Either set or release.
fname	Filename would be bookmarked.
bname	Bookmark title.
starttime	Only using in 'set' mode. It appears 'XXmXXs'. This time calculates from the file's starting point.
endtime	Same as 'starttime' attribute. If 'endtime' is set and 'endtime' is longer than the actual file's time length, it will automatically be set the file's end time.

1.5. Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE rcxml PUBLIC "-//W3C//DTD REMOTECONTROLXML 2.1//EN"
  "http://www.w3c.org/xml/rcxml.dtd">
<rcxml version="0.5">
    <device appliance="pvr">
        <power_on/>
```

```
            <session id="s001">
                <start_operation id="op1">
                    <set_book ch_no="11" date="20041213"
stime="2100" duration="45m" mode="sd"/>
                </start_operation>
                <if cond="op1=='sucess'">
                    <send id="s001"
target="url:www.aaa.com?sid=s001& user=user1" result="success"/>
                    <else/>
                    <send id="s001"
target="url:www.aaa.com?sid=s001& user=user1"
result="fail"errorDescription="op1"/>
                </if>
            </session>
            <power_off/>
    </device>
</rcxml>
```