Adobe

Extending the Acrobat SaveAsXML Plugin

1/27/21 Adobe, Inc.

About the Mapping Tables

This chapter describes how the SaveAsXML plug-in works. It describes a sample mapping table and its directives, and the steps in editing the mapping tables.

How SaveAsXML interacts with the mapping tables

When the SaveAsXML plug-in registers itself with Acrobat DC, it inspects the set of XML files in the `MappingTables` folder to determine the number of conversion services that are available.

The `MappingTables` folder must be inside the `SaveAsXML` folder, which is at the same level as `SaveAsXML.api`. Files in the `MappingTables` folder are the only ones that are inspected as potential conversion services supported by the plug-in. This folder must not contain any files with the `.xml` extension that are not mapping tables.

If the registration process finds the Root element and its menu-name attribute, which may be a string or a predefined identifier, it adds the menu-name to the list of file format choices available in the Save As dialog box. The menu-name must be unique, or the user may be confused by similarly identified entries among the Save As dialog box's file formats.

When a user selects an applicable file format in the Save As dialog box, the dialog box handler activates the SaveAsXML plug-in. The plug-in reads the associated mapping table and converts it to a binary in-memory format, which it uses to control the processing of the current tagged PDF document.

Sample mapping table

The following sample mapping table, which is simplified and incomplete, demonstrates the basic operations of SaveAsXML processing. The sample is followed by a detailed analysis of the directives.

For more complete examples, see the mapping tables distributed with SaveAsXML. Directives that are currently supported are used in one or more of the distributed tables. For a reference of directives and their attributes, see the following chapter, Mapping Table Elements Reference.

Example: *Sample mapping table*

```
<Root File-format = "Xml-1-00" Menu-name = "Sample Mapping Table"
      Mac-creator = "MSIE" Mac-type = "TEXT" Win-suffix = "xml"
      Encode-out = "Utf-8-out">
  <Emit-string ... >&lt;XML-Doc&gt;</Emit-string>
  <Walk-structure Use-event-list = "Block-events"></Walk-structure>
  <Emit-string ...>&lt;/XML-Doc&gt;</Emit-string>
  <Define-event-list Name = "Block-events">
    <Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
            Node-name = "Div" Alternate-name = "-none-"
            Node-content = "Has-kids" Event-class = "Enter">
      <Emit-string ...>&lt;Div</Emit-string>
```

```
      <Call-proc-list Name = "Block-attributes"></Call-proc-list>
      <Emit-string ...>&gt;</Emit-string>
      <Walk-children Use-event-list = "Inline-events"></Walk-children>
    </Event>
      <Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
              Node-name = "Div" Alternate-name = "-none-"
              Node-content = "Has-kids" Event-class = "Exit">
      <Emit-string ...>&lt;/Div&gt;</Emit-string>
    </Event>
    <Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
            Node-name = "Div" Alternate-name = "-none-"
            Node-content = "Empty" Event-class = "Enter">
      <Emit-string ...>&lt;Div</Emit-string>
      <Call-proc-list Name = "Block-attributes"></Call-proc-list>
      <Emit-string ...>/&gt;</Emit-string>
    </Event>
  </Define-event-list>
  <Define-event-list Name = "Inline-events">
    <Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
            Node-name = "Span" Alternate-name = "-none-"
            Node-content = "Has-kids" Event-class = "Enter">
      <Emit-string ...>&lt;Span</Emit-string>
      <Call-proc-list Name = "Span-attributes"></Call-proc-list>
      <Emit-string ...>&gt;</Emit-string>
      <Walk-children Use-event-list = "Inline-events"></Walk-children>
    </Event>
    <Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
            Node-name = "Span" Alternate-name = "-none-"
            Node-content = "Has-kids" Event-class = "Exit">
      <Emit-string ...>&lt;/Span&gt;</Emit-string>
    </Event>
    <Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
            Node-name = "Span" Alternate-name = "-none-"
            Node-content = "Empty" Event-class = "Enter">
      <Emit-string ...>&lt;Span</Emit-string>
      <Call-proc-list Name = "Span-attributes"></Call-proc-list>
      <Emit-string ...>/&gt;</Emit-string>
    </Event>
    <Event  Inf-type = "Pds-mc" Name-type = "Any" Node-name = "-none-"
            Alternate-name = "-none-" Node-content = "Has-text-only"
            Event-class = "Enter">
      <Proc-doc-text do-br-substitution =
"do-br-substitution"></Proc-doc-text>
    </Event>
  </Define-event-list>
  <Define-proc-list Name = "Block-attributes">
    <Proc-var  Pdf-var = "Alt" Owner = "Structelem" Type = "String"
               Has-enum = "No-enum" Inherit = "Not-inherited" Default =
"-none-"
               Condition = "Has-value">
      <Emit-string ...>alt="</Emit-string>
      <Proc-string></Proc-string>
      <Emit-string ...>"</Emit-string>
    </Proc-var>
  </Define-proc-list>
```

```
  <Define-proc-list Name = "Span-attributes">
    <Proc-var  Pdf-var = "ActualText" Owner = "Structelem" Type = "String"
               Has-enum = "No-enum" Inherit = "Not-inherited" Default =
"-none-"
               Condition = "Always">
      <Emit-string ...>actual-text="</Emit-string>
      <Proc-string></Proc-string>
      <Emit-string ...>"</Emit-string>
    </Proc-var>
  </Define-proc-list>
</Root>
```

Root node

Processing begins with the root node of the mapping table and generally proceeds as a pre-order hierarchical traversal of the control nodes.

```
<Root File-format = "Xml-1-00" Menu-name = "Sample Mapping Table"
      Mac-creator = "MSIE" Mac-type = "TEXT" Win-suffix = "xml"
      Encode-out = "Utf-8-out">
```

In processing the `Root` node of the mapping table, the SaveAsXML processor opens the output file using the path and name of the PDF document to be saved, replacing the file suffix with that specified by the `Win-suffix` attribute in this node. In Mac OS, the `Mac-creator` and `Mac-type` are also used to open the output file. The remaining attributes in the `Root` node are available to the SaveAsXML processor and are used to control or optimize the conversion.

Emit-string

```
<Emit-string ... >&lt;XML-Doc&gt;</Emit-string>
```

The `Emit-string` directive causes its content to be translated to the output encoding specified in the `Encode-out` attribute of the `Root` node, then emits the converted data to the output file. In this sample, it issues the start tag for the document: `<XML-Doc>`. For clarity, the additional attributes of the `Emit-string` directive have been omitted in the sample.

Here, as in any mapping table directive, the following code is used to represent special characters:

`<` represents the less-than (<) character.

`>` represents the greater-than (>) character.

`&` represents the ampersand (&) character.

Walk-structure

```
<Walk-structure Use-event-list = "Block-events"></Walk-structure>
```

The `Walk-structure` directive causes the SaveAsXML processor to walk the first-level structural elements (Kids array of the StructRoot) of the tagged PDF document to be saved. For more information, see "Walk-children" on page 9.

Structural elements are traversed in the order found in the logical structure tree. An event is generated on entering and on exiting each structural element. The event-list specified by the `Use-event-list` attribute of the `Walk-structure` directive is searched for a matching `Event` directive. For more information, see "Define-event-list" on page 8.

If a match is found, the directives within that `Event` directive are processed (which may include the recursive processing of children of the current structural element via a `Walk-children` directive). Searching of the event-list is terminated and the next event is generated.

If no match is found, or when processing is completed on the matching `Event` directive, the next event is generated.

Processing continues until all first-level structural elements (Kids array of the StructRoot) have been traversed, then the directive following the `Walk-structure` directive is processed. In this sample, it is:

```
<Emit-string Emit-space-after = "Emit-space-after" ...>
&#lt;/XML-Doc&gt;
</Emit-string>
```

This `Emit-string` directive issues the end tag: `</XML-Doc>`. Because newlines and spaces are often modified or stripped by various XML tools, the `Emit-space-after` attribute, and the other related attributes of the `Emit-string` directive, guarantees the retention of these characters.

Define-event-list

```
<Define-event-list Name = "Block-events">
```

The `Define-event-list` directive is similar to a macro or subroutine definition in most programming languages. It encapsulates and names a set of event directives. The directives are activated by a `Walk-structure`, `Walk-children`, or `Call-event-list` directive specifying the name of the event list in the `Use-event-list` attribute.

Event

```
<Event  Inf-type = "Struct-elem" Name-type = "Structure-role"
        Node-name = "Div" Alternate-name = "-none-"
        Node-content = "Has-kids" Event-class = "Enter">
```

The `Event` directive includes a set of attributes that are used to determine if the directives within it are to be processed. The directive in the sample is activated by entering (either from a parent element or from the prior peer element) a structural element (`Inf-type = "Struct-elem"`), where the element is role-mapped (`Name-type = "Structure-role"`) to `"Div"` and the element has children.

When an `Event` directive is activated, the directives within it (before its `</Event>` tag) are processed. In this sample, the directive is:

```
<Emit-string ...>&lt;Div</Emit-string>
```

This issues the `"Div"` portion of the output element's start-tag.

Call-proc-list

```
<Call-proc-list Name = "Block-attributes"></Call-proc-list>
```

The `Call-proc-list` directive processes the properties associated with this structural element, using the processing list specified by the `Name` property on the `Call-proc-list` directive.

Although the event-list processing stops on the first match, the proc-list processing continues for every directive in the selected processing list.

The directive:

```
<Emit-string ...>&gt;</Emit-string>
```

issues the closing `">"` on the output element's start-tag.

Walk-children

```
<Walk-children Use-event-list = "Inline-events"></Walk-children>
```

The `Walk-children` directive is functionally identical to the `Walk-structure` directive, except that it walks the first level children of the current structural element.

The `</Event>` tag indicates the end of the processing for this event. Remaining entries in this event-list follow a similar model.

The next `Event` included in this event-list handles events that are generated when exiting `Div` elements that have children. This generates the close tag on the output element.

```
<Event         Inf-type = "Struct-elem" Name-type = "Structure-role"
               Node-name = "Div" Alternate-name = "-none-"
               Node-content = "Has-kids" Event-class = "Exit">
      <Emit-string ...>&lt;/Div&gt;</Emit-string>
</Event>
```

The final `Event` directive included in this event-list handles events that are generated on entering an element which has no children. It does not and should not contain a `Walk-children` directive.

```
  <Event         Inf-type = "Struct-elem" Name-type = "Structure-role"
                 Node-name = "Div" Alternate-name = "-none-"
                 Node-content = "Empty" Event-class = "Enter">
          <Emit-string ...>&lt;Div</Emit-string>
          <Call-proc-list Name = "Block-attributes"></Call-proc-list>
          <Emit-string ...>/&gt;</Emit-string>
  </Event>
</Define-event-list>
```

The `</Define-event-list>` tag ends the list of entries in the `Block-events` event-list.

The following event-list handles inline elements and is similar to the one above.

```
<Define-event-list Name = "Inline-events">
  <Event         Inf-type = "Struct-elem" Name-type = "Structure-role"
                 Node-name = "Span" Alternate-name = "-none-"
                 Node-content = "Has-kids" Event-class = "Enter">
          <Emit-string ...>&lt;Span</Emit-string>
          <Call-proc-list Name = "Span-attributes"></Call-proc-list>
          <Emit-string ...>&gt;</Emit-string>
          <Walk-children Use-event-list = "Inline-events">
          </Walk-children>
  </Event>
  <Event         Inf-type = "Struct-elem" Name-type = "Structure-role"
                 Node-name = "Span" Alternate-name = "-none-"
                 Node-content = "Has-kids" Event-class = "Exit">
          <Emit-string ...>&lt;/Span&gt;</Emit-string>
  </Event>
  <Event         Inf-type = "Struct-elem" Name-type = "Structure-role"
                 Node-name = "Span" Alternate-name = "-none-"
                 Node-content = "Empty" Event-class = "Enter">
```

```
            <Emit-string ...>&lt;Span</Emit-string>
            <Call-proc-list Name = "Span-attributes"></Call-proc-list>
            <Emit-string ...>/&gt;</Emit-string>
    </Event>
```

For event-lists that process structural elements that contain text or graphics, an `Event` entry similar to the following is required. The code in the SaveAsXML plug-in that traverses the logical structure tree also reports entering and exiting of the marked content containers (the wrappers around the low-level text and graphic content in the PDF page's marking stream). The labels on these nodes are hidden in the Tags view in Acrobat DC. (The corresponding `Event` for a `Pds-mc` element where the content is `Image` is more complex. See the mapping tables distributed with SaveAsXML for complete examples.)

```
<Event          Inf-type = "Pds-mc" Name-type = "Any" Node-name = "-none-"
                Alternate-name = "-none-" Node-content = "Has-text-only"
                Event-class = "Enter">
```

This `Event` directive processes the low-level marked content containers (`Inf-type = "Pds-mc"`) that actually contain the text (`Node-content = "Has-text-only"`). A corresponding exit directive is not required.

Proc-doc-text

```
<Proc-doc-text do-br-substitution = "do-br-substitution"></Proc-doc-text>
```

The `Proc-doc-text` directive converts the text from the active marked content container in the PDF page's marking stream to the output encoding specified in the `Encode-out` attribute of the `Root` node and then emits the converted data to the output file. The `do-br-substitution` attribute controls whether the LF character is to be converted to a `<BR/>` tag in the output stream, converted to a space, or discarded.

```
    </Event>
</Define-event-list>
```

Define-proc-list

```
<Define-proc-list Name = "Block-attributes">
```

The `Define-proc-list` directive is also a macro or subroutine similar to the `Define-event-list` directive. Whereas the event-list describes how to process transition events in traversing the logical structure tree, the proc-list describes how to process the properties (attributes) of a structural element.

Proc-var

```
<Proc-var       Pdf-var = "Alt" Owner = "Structelem" Type = "String"
                Has-enum = "No-enum" Inherit = "Not-inherited"
                Default = "-none-" Condition = "Has-value">
```

The `Proc-var` directive searches an internal cache of the properties on the current structural element for the value of the property specified by its `Pdf-var` and `Owner` attributes. If inheritance is enabled, it also searches the cached properties of all ancestors of the current structural element for an applicable value. Once it determines if there is (or is not) a value, it uses the remaining attributes to determine if the value should be processed. If it determines it should be processed, then the directives contained in the `Proc-var` directive are processed.

Proc-string

```
<Emit-string ...>alt="</Emit-string>
<Proc-string></Proc-string>
```

The `Proc-string` directive causes the string selected by the containing `Proc-var` directive to be translated to the output encoding specified in the `Encode-out` attribute of the `Root` node, and then emits the converted data to the output file.

```
                <Emit-string ...>"</Emit-string>
    </Proc-var>
</Define-proc-list>
```

The `</Define-proc-list>` tag indicates the end of this proc-list.

The following proc-list has a similar organization for `Block-attributes`.

```
  <Define-proc-list Name = "Span-attributes">
      <Proc-var         Pdf-var = "ActualText" Owner = "Structelem"
                        Type = "String" Has-enum = "No-enum"
                        Inherit = "Not-inherited" Default = "-none-"
                        Condition = "Always">
                <Emit-string ...>actual-text="</Emit-string>
                <Proc-string></Proc-string>
                <Emit-string ...>"</Emit-string>
      </Proc-var>
  </Define-proc-list>

</Root>
```

The `</Root>` tag is the last line of a mapping table file. It indicated the end of the `Root` directive.

Editing the mapping tables

You can edit the `.xml` versions of the mapping tables in any XML or SGML editor.

2 Mapping Table Elements Reference

This chapter provides complete details of all mapping table directives and their attributes.

Call-event-list

Inserts the named event-list at this point in the mapping table. This directive is identical to a macro call.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Name`	String	Required. The name of a event list (as in `<Define-event-list>`) to be included at this point in the current event list.

Call-proc-list

Inserts the named proc-list at this point in the mapping table. This directive is identical to a macro call.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Name`	String	Required. The name of a variable processing list (see `<define-proc-list>`) to be included at this point in the current event or proc-list.

Comment

Allows documentation or notes to be included in the mapping table. This directive does no processing.

DTD content rule

```
<TEXT>
```

Conditional-delimiter

Emits the contained text if this `proc-var` is not the first one to be accepted and processed after the start of an event or the first one to be processed after a `conditional-prefix` control element.

DTD content rule

```
<TEXT>
```

Attributes

Name	Type	Description
`Emit-newline-before`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-before` — Emit a newline before emitting any content text. `No-newline-before` — Do not emit a newline before emitting any content text.
`Emit-newline-after`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-after` — Emit a newline after emitting any content text. `No-newline-after` — Do not emit a newline after emitting any content text.
`Emit-space-after`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-after` — Emit a space after emitting any content text. `No-space-after` — Do not emit a space after emitting any content text.
`Emit-space-before`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-before` — Emit a space before emitting any content text. `No-space-before` — Do not emit a space before emitting any content text.

Conditional-prefix

Caches and emits the contained text if any `proc-var` is accepted to be processed before the end of the current event or before the next `Conditional-suffix` control element.

DTD content rule

```
<TEXT>
```

Attributes

Name	Type	Description
`Emit-newline-after`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-after` — Emit a newline after emitting any content text. `No-newline-after` — Do not emit a newline after emitting any content text.
`Emit-newline-before`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-before` — Emit a newline before emitting any content text. `No-newline-before` — Do not emit a newline before emitting any content text.
`Emit-space-after`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-after` — Emit a space after emitting any content text. `No-space-after` — Do not emit a space after emitting any content text.
`Emit-space-before`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-before` — Emit a space before emitting any content text. `No-space-before` — Do not emit a space before emitting any content text.

Conditional-suffix

Emits the contained text if the preceding `Conditional-prefix` within the current event was emitted.

DTD content rule

```
<TEXT>
```

Attributes

Name	Type	Description
`Emit-newline-after`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-after` — Emit a newline after emitting any content text. `No-newline-after` — Do not emit a newline after emitting any content text.
`Emit-newline-before`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-before` — Emit a newline before emitting any content text. `No-newline-before` — Do not emit a newline before emitting any content text.
`Emit-space-after`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-after` — Emit a space after emitting any content text. `No-space-after` — Do not emit a space after emitting any content text.
`Emit-space-before`	Choice	Required. Since XML strips the first and last space in each element and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-before` — Emit a space before emitting any content text. `No-space-before` — Do not emit a space before emitting any content text.

Define-event-list

Event-lists and proc-lists, like macros, allow the user to define a series of processing directives that can be used in multiple locations within the SaveAs mapping table.

Event-lists govern the selection and processing of elements in the layout, metadata, logical structure, or stylesheet trees. Proc-lists govern the processing of attributes or properties associated with a given event or structural element. For more information, see `Define-proc-list`.

DTD content rule

```
( Comment | Event | Call-event-list)+
```

Attributes

Name	Type	Description
`Name`	String	Required. The name to be applied to the event processing list being defined by this element. This is referenced in the `<Walk-*>` elements by the `Use-event-list` attribute. The name must be unique across all Define-event-list elements within a given mapping table file.

Define-proc-list

Proc-lists and event-lists, like macros, allow the user to define a series of processing directives that can be used in multiple locations within the SaveAs mapping table.

Proc-lists govern the processing of attributes and properties associated with a given event or structural element. Event-lists govern the selection and processing of elements in the layout, metadata, logical structure, or stylesheet trees. (See `Define-event-list`.)

DTD content rule

```
(Comment | Proc-var | Walk-proplist | Call-proc-list)+
```

Attributes

Name	Type	Description
`Name`	String	Required. The name to be applied to the variable processing list being defined by this element. This is referenced in the `<Call-proc-list>` element via its `Name` attribute. The name must be unique across all `Define-proc-list` elements within a given mapping table file.

Element-name

Outputs the Element-name, which is used in the XML output filter to generate the user-supplied element tag.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`Node-type`	Choice	Required. Specifies whether to get the structural element name to emit directly from the `/S` key of the StructElem (`Structure-user-label`) or from the result of processing that key via the RoleMap (`Structure-role`). One of: `Structure-role` — Use the result of processing the StructElem's `/S` key via the RoleMap. `Structure-user-label` — Use the StructElem's `/S` key.

Emit-all-metadata

Copies the full set of XAP metadata to the output file.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`Emit-newline-after`	Choice	Required. XML strips the first and last space and most newlines from the parsed result, so it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-after` — Emit a newline after emitting any content text. `No-newline-after` — Do not emit a newline after emitting any content text.
`Emit-newline-before`	Choice	Required. XML strips the first and last space in each element and most newlines from the parsed result, so it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-before` — Emit a newline before emitting any content text. `No-newline-before` — Do not emit a newline before emitting any content text.
`Emit-space-after`	Choice	Required. XML strips the first and last space in each element and most newlines from the parsed result, so it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-after` — Emit a space after emitting any content text. `No-space-after` — Do not emit a space after emitting any content text.
`Emit-space-before`	Choice	Required. XML strips the first and last space in each element and most newlines from the parsed result, so it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-before` — Emit a space before emitting any content text. `No-space-before` — Do not emit a space before emitting any content text.

Emit-string

Emits the text contained in this mapping table element.

DTD content rule

```
<TEXT>
```

Attributes

Name	Type	Description
`Emit-newline-after`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-after` — Emit a newline after emitting any content text. `No-newline-after` — Do not emit a newline after emitting any content text.
`Emit-newline-before`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-newline-before` — Emit a newline before emitting any content text. `No-newline-before` — Do not emit a newline before emitting any content text.
`Emit-space-after`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-after` — Emit a space after emitting any content text. `No-space-after` — Do not emit a space after emitting any content text.
`Emit-space-before`	Choice	Required. Since XML strips the first and last space and most newlines from the parsed result, it is necessary to have this set of flags to control explicit insertion of these control codes. One of: `Emit-space-before` — Emit a space before emitting any content text. `No-space-before` — Do not emit a space before emitting any content text.

Evaluate-var

Does the same processing as `Proc-var`, except it does not make the data value available to the other contained processing directives. For more information, see "Proc-var" on page 29.

DTD content rule

```
(Comment | Conditional-delimeter | Emit-string | Conditional-prefix |
Element-name | Proc-string | Proc-integer | Proc-fixed | Proc-length |
```

```
Proc-pixels | Proc-enum | Proc-var | Walk-proplist | Call-proc-list |
Proc-graphic-content | Proc-image-content | Proc-doc-text | Walk-children |
Walk-metadata | Emit-all-metadata | Walk-cached-property-sets |
Walk-structure | Walk-layout | Conditional-suffix )+
```

Attributes

Name	Type	Description
`Compare`	String	Optional. The value used to determine `Diff-from-value`, `Matches-value`, `Less-than-value`, or `More-than-value`. This should be the same type (Fixed, Int32, Atom, String) as the property.
`Condition`	Choice	Required. Indicates whether the directives that are children of the `Proc-var` directive are to be executed. One of: `Always` — Always execute the children of this `Proc-var` directive. `Has-value` — Execute the children of this `Proc-var` directive if a value is found on this node (either explicit or Default). `Diff-from-default-for-event` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by Default. `Diff-from-ancestor` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by searching the inheritance tree for any ancestor. `Diff-from-parent` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by examining the inheritance cache of the parent. `Diff-from-predecessor` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by examining the inheritance cache of the preceding peer. `Diff-from-value` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by `Compare`. Can be used with any type. `Matches-value` — Execute the children of this `Proc-var` directive if a value is found and that value matches that specified by `Compare`. Can be used with any type. `Less-than-value` — Execute the children of this `Proc-var` directive if a value is found and that value is less than that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String. `Less-equal-value` — Execute the children of this `Proc-var` directive if a value is found and that value is less than or equal to that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String. `More-than-value` — Execute the children of this `Proc-var` directive if a value is found and that value is greater than that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String. `More-equal-value` — Execute the children of this `Proc-var` directive if a value is found and that value is greater than or equal to that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String.

Name	Type	Description
`Default`	String	Optional. The value to be used if the property is not found on this element (or through inheritance). This should be the same type (Fixed, Int32, Atom, String) as the property.
`Inherit`	Choice	Optional. Whether the property value can be inherited from a parent. One of: `Inheritable` — This property can be inherited. `Not-inherited` — This property cannot be inherited (Default).
`Owner`	Choice	Required. The owner of the property dictionary. One of: `Metadata` — This is a pseudo-owner for entries in the document's metadata. `Structelem` — This is a pseudo-owner for properties specified directly in the StructElem's obj dictionary. `Layout` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by Layout. `Link` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by Link. `List` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by List. `Table` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by Table. `Auto-span` — This is a pseudo-owner generated by the SaveAs processor for each span it synthesizes by consolidating Tj operators having common styling properties (font, size, color, etc.) `Inline-markup` — This is a pseudo-owner generated by the SaveAs processor when the following inline marking is encountered: `/Span << ... >> BDC (abbrev.) Tj EMC`
`Pdf-var`	String	Required. The name of a property in a given property dictionary (see `Owner`) to be processed or evaluated.
`Type`	Choice	Required. The primary PDF datatype of the property (see `Has-enum` for a possible secondary datatype). One of: `Fixed` — Fixed-point number. `Int32` — A signed integer. `Atom` — A PDF key (/XYZ). `String` — A PDF string. `Color` — An RGB color (array of three Fixed values). `BBox` — A bounding box (array of four Fixed values).

Event

Governs the processing of a node in the layout, logical-structure, metadata, or stylesheet trees. Specifies the processing that is to be performed on entering or exiting the named node.

DTD content rule

```
(Comment | Emit-string | Conditional-prefix | Element-name | Proc-var |
Walk-proplist | Call-proc-list | Conditional-suffix | Proc-graphic-content |
Proc-image-content | Proc-doc-text | Walk-children | Walk-metadata |
Emit-all-metadata | Walk-cached-property-sets | Walk-structure | Walk-layout |
Evaluate-var)+
```

Attributes

Name	Type	Description
`Event-class`	Choice	Required. Identifies which transition into or out of the node is to be processed using this event description. One of: `Enter` — Node is being entered from either parent or peer. `Enter-from-parent` — Node is being entered from parent, but not from peer. `Enter-from-peer` — Node is being entered from peer, but not from parent. `Exit` — Node is being exited to either parent or to peer. `Exit-to-parent` — Node is being exited to parent, but not to peer. `Exit-to-peer` — Node is being exited to peer, but not to parent. `Begin-children` — Node is being exited to begin processing its children. `End-children` — Node is being re-entered after processing its children.
`Node-content`	Choice	Required. One of: `Empty` — Node has no children or direct content. `Has-text-only` — Node has only text content (no other elements). `Has-kids` — Node has child elements (including possible text-only spans. `Graphic` — Node contains (vector) graphic data. `Image` — Node contains bitmap image data. `Other` — Node is something other than those listed above.

Name	Type	Description
`Node-name`	String	Required. Name of the element or role to match, in order to select this event descriptor for processing.
`Node-type`	Choice	Required. The `Node-name` attribute is matched against either the `/S` key of the StructElem (`Structure-user-label`) or against the result of processing that key via the RoleMap (`Structure-role`). One of: `Any` — Attempt to match on `Structure-user-label` then on `Structure-role`. Also used for matching within metadata and stylesheet construction. `Structure-role` — Compare `Node-name` to the result of processing the StructElem's `/S` key via the RoleMap. `Structure-user-label` — Compare `Node-name` to the StructElem's `/S` key.

Proc-doc-text

Emits the text contained in the current structural element.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`do-br-substitution`	Choice	Required. One of: `do-br-substitution` — Emit a ` ` for every newline found in the doc text. `do-xml-br-substitution` — Emit a ` ` for every newline found in the doc text. `no-substitution` — Disregard newlines in doc text.

Proc-enum

If the data cached by the containing `Proc-var` directive is a string or an atom, searches for a match among the `proc-enum` choice elements that are children of this control element. If a match is found, issues the `Value-out` value of the matching `Proc-enum-choice` directive as a string.

DTD content rule

`Proc-enum-choice+`

Proc-enum-choice

Specifies the choice and output values for a `Proc-enum` directive.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`Value-in`	String	Required. This value is compared to the value cached by the containing `proc-var` directive.
`Value-out`	String	Required. This value is emitted as a string if a match against `Value-in` is found.

Proc-fixed

If the data cached by the containing `Proc-var` directive is a FixedPoint number, emits the text representation of the value. This value is scaled using the attributes of this directive as follows:

1. The original value is multiplied by the value of the `Mul` attribute.
2. The value of the `Add` attribute is added to the result of step 1.
3. The result of step 2 is divided by `Div`.
4. The result of step 3 is converted to a string. `Frac-len` controls the number of digits to the right of the decimal point. `Frac-dlm` is the fraction-radix character to be issued if `Frac-len` is greater than 0.

`Proc-fixed`, `Proc-length`, and `Proc-pixels` vary only in the default values for `Mul`, `Div`, and `Add`.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`Add`	String	Optional. Default is 0.
`Div`	String	Optional. Default is 1.
`Frac-dlm`	String	Optional. Default is "."
`Frac-len`	String	Optional. Default is 2.
`Mul`	String	Optional. Default is 1.

Proc-graphic-content

Processes the content of the current structural element as a vector graphic.

DTD content rule

`Void?`

Proc-hex

If the data cached by the containing `Proc-var` directive is an Int32, an Uns32, or a Fixed, emits the text representation of the integer portion of the value, after the scaling algorithm is applied. This value is scaled using the attributes of this directive as follows:

1. The original value is multiplied by the value of the `Mul` attribute.
2. The value of the `Add` attribute is added to the result of step 1.
3. The result of step 2 is divided by `Div` and the fraction is discarded.
4. The result of step 3 is converted to a string.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`Add`	String	Optional. Default is 0.
`Div`	String	Optional. Default is 1.
`Mul`	String	Optional. Default is 1.
`Num-digits`	String	Optional. Default is 2.

Proc-image-content

Processes the content of the current structural element as a bitmapped graphic.

DTD content rule

`Void?`

Proc-integer

If the data cached by the containing `Proc-var` directive is an Int32 or an Uns32, emits the text representation of the value. This value is scaled using the attributes of this directive as follows:

1. The original value is multiplied by the value of the `Mul` attribute.
2. The value of the `Add` attribute is added to the result of step 1.
3. The result of step 2 is divided by `Div` and the fraction is discarded.
4. The result of step 3 is converted to a string.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Add`	String	Optional. Default is 0.
`Div`	String	Optional. Default is 1.
`Mul`	String	Optional. Default is 1.

Proc-length

If the data cached by the containing `Proc-var` directive is a FixedPoint number, emits the text representation of the value. This value is scaled using the attributes of this directive as follows:

1. The original value is multiplied by the value of the `Mul` attribute.
2. The value of the `Add` attribute is added to the result of step 1.
3. The result of step 2 is divided by `Div`.
4. The result of step 3 is converted to a string. `Frac-len` controls the number of digits to the right of the decimal point. `Frac-dlm` is the fraction-radix character to be issued if `Frac-len` is greater than 0.

`Proc-fixed`, `Proc-length`, and `Proc-pixels` vary only in the default values for `Mul`, `Div`, and `Add`.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Add`	String	Optional. Default is 0.
`Div`	String	Optional. Default is 72.
`Frac-dlm`	String	Optional. Default is "." (decimal point).
`Frac-len`	String	Optional. Default is 2.
`Mul`	String	Optional. Default is 72.

Proc-pixels

If the data cached by the containing `Proc-var` directive is a FixedPoint number, emits the text representation of the value. This value is scaled using the attributes of this directive:

1. The original value is multiplied by the value of the `Mul` attribute.
2. The value of the `Add` attribute is added to the result of step 1.
3. The result of step 2 is divided by `Div`.
4. The result of step 3 is converted to a string. `Frac-len` controls the number of digits to the right of the decimal point. `Frac-dlm` is the fraction-radix character to be issued if `Frac-len` is greater than 0.

`Proc-fixed`, `Proc-length`, and `Proc-pixels` vary only in the default values for `Mul`, `Div`, and `Add`.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Add`	String	Optional. Default is 36.
`Div`	String	Optional. Default is 72.
`Frac-dlm`	String	Optional. Default is "." (decimal point).
`Frac-len`	String	Optional. Default is 0.
`Mul`	String	Optional. Default is 96.

Proc-property

Processes an arbitrary property. This is similar to `proc-var`, except that it does not select or filter which properties are processed, but simply takes each property owned by the current owner in turn.

DTD content rule

```
(Comment | Conditional-delimeter | Emit-string | Property-name |
Property-type)+
```

Proc-string

If the data cached by the containing `Proc-var` directive is a string or an atom, emits the text content of the string or a text representation of the atom's name.

DTD content rule

```
Void?
```

Proc-var

Specifies the formatting and conversion of the named attribute or property (PDF-variable).

This directive also caches the data value and type of the value specified for use by various processing directives within this element.

DTD content rule

```
(Comment | Conditional-delimeter |Emit-string | Conditional-prefix |
Element-name | Proc-string | Proc-integer | Proc-fixed | Proc-length |
Proc-pixels | Proc-enum | Proc-doc-text | Proc-graphic-content |
Proc-image-content | Conditional-suffix )+
```

Attributes

Name	Type	Description
`Compare`	String	Optional. The value used to determine `Diff-from-value`, `Matches-value`, `Less-than-value`, or `More-than-value`. This should be the same type (Fixed, Int32, Atom, String) as the property.
`Condition`	Choice	Required. Indicates whether the directives that are children of the `Proc-var` directive are to be executed. One of: `Always` — Always execute the children of this `Proc-var` directive. `Has-value` — Execute the children of this `Proc-var` directive if a value is found on this node (either explicit or Default). `Diff-from-default-for-event` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by Default. `Diff-from-ancestor` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by searching the inheritance tree for any ancestor. `Diff-from-parent` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by examining the inheritance cache of the parent. `Diff-from-predecessor` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by examining the inheritance cache of the preceding peer. `Diff-from-value` — Execute the children of this `Proc-var` directive if a value is found and that value differs from that specified by `Compare`. Can be used with any type. `Matches-value` — Execute the children of this `Proc-var` directive if a value is found and that value matches that specified by `Compare`. Can be used with any type. `Less-than-value` — Execute the children of this `Proc-var` directive if a value is found and that value is less than that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String. `Less-equal-value` — Execute the children of this `Proc-var` directive if a value is found and that value is less than or equal to that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String. `More-than-value` — Execute the children of this `Proc-var` directive if a value is found and that value is greater than that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String. `More-equal-value` — Execute the children of this `Proc-var` directive if a value is found and that value is greater than or equal to that specified by `Compare`. Can only be used with: Fixed, Int32, Atom, String.

Name	Type	Description
`Default`	String	Optional. The value to be used if the property is not found on this element (or through inheritance). This should be the same type (Fixed, Int32, Atom, String) as the property.
`Inherit`	Choice	Optional. Whether the property value can be inherited from a parent. One of: `Inheritable` — This property can be inherited. `Not-inherited` — This property cannot be inherited (Default).
`Owner`	Choice	Required. The owner of the property dictionary. One of: `Metadata` — This is a pseudo-owner for entries in the document's metadata. `Structelem` — This is a pseudo-owner for properties specified directly in the StructElem's obj dictionary. `Layout` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by Layout. `Link` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by Link. `List` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by List. `Table` — Properties in the StructElem's Attribute dictionary list within the dictionary owned by Table. `Auto-span` — This is a pseudo-owner generated by the SaveAs processor for each span it synthesizes by consolidating Tj operators having common styling properties (font, size, color, etc.). `Inline-markup` — This is a pseudo-owner generated by the SaveAs processor when the following inline marking is encountered: `/Span << ... >> BDC (abbrev.) Tj EMC`
`Pdf-var`	String	Required. The name of a property in a given property dictionary (see Owner) to be processed or evaluated.
`Type`	Choice	Required. The primary PDF datatype of the property (see `Has-enum` for a possible secondary datatype). One of: `Fixed` — Fixed-point number. `Int32` — A signed integer. `Atom` — A PDF key (/XYZ). `String` — A PDF string. `Color` — An RGB color (array of three Fixed values) `BBox` — A bounding box (array of four Fixed values)

Property-name

Processes the name and key portion of an arbitrary property.

DTD content rule

```
Void?
```

Property-type

Processes the data portion of an arbitrary property.

DTD content rule

```
(Comment | Conditional-delimeter | Emit-string | Proc-string | Proc-integer |
Proc-fixed | Proc-length | Proc-pixels | Proc-enum | Proc-doc-text |
Proc-graphic-content | Proc-image-content)+
```

Attributes

Name	Type	Description
`Type`	Choice	Required—The primary PDF datatype of the property. One of: `Fixed` — Fixed-point number. `Int32` — A signed integer. `Atom` — A PDF key (/XYZ). `String` — A PDF string. `Color` — An RGB color (array of three Fixed values). `BBox` — A bounding box (array of four Fixed values).

Root

The root node of a mapping table. Its attributes specify the name of the filter to appear in the menu and information necessary to properly generate the output file name and type information.

DTD content rule

```
(Comment | Emit-string | Define-event-list | Define-proc-list |
Walk-metadata | Emit-all-metadata | Walk-cached-property-sets |
Walk-structure | Walk-layout)+
```

Attributes

Name	Type	Description
`Encode-out`	Choice	Required. The encoding of the output file. One of: `Utf-8-out` — The file is encoded in UTF-8 (8-bit Unicode). `Utf-16-out` — The file is encoded in UTF-16 (16-bit Unicode). `Ucs-4-out` — The file is encoded in UCS-4 (32-bit Unicode). `Iso-latin-1-out` — The file is encoded as ISO-Latin-1. All Unicode values above 0x00FF are output as numeric character entities (&#xFFFF;). `Html-ascii-out` — The file is encoded as 7-bit ASCII. All Unicode values above 0x007F are output as numeric character entities (&#xFFFF;).

`File-format`	Choice	Required. Internal unique name that describes the format of the output file. The following formats are provided:
		`Html-3-02`
		`Html-4-01-with-css-1-00`
		`Xml-1-00`
		`Plain Text`

`Mac-creator`	String	Required. The file creator field for a Mac OS file.
`Mac-type`	String	Required. The file type field for a Mac OS file.

`Menu-name`	String or Identifier	Required. The text string describing the file format that appears in the Save As dialog box's pulldown menu. The following predefined identifiers, which provide localized menu name strings, can be used in place of a string:
		`$IDS_HTML_3_2_MENU_NAME - localized string "HTML 3.2"`
		`$IDS_HTML_4_01_CSS_1_0_MENU_NAME - localized string "HTML 4.01 with CSS 1.0"`
		`$IDS_XML_1_0_MENU_NAME - localized string "XML 1.0"`
		`$IDS_PLAIN_TEXT_MENU_NAME - localized string "Text (Plain)"`

`Win-suffix`	String	Required. The three-letter file type suffix for the Windows environment. Also used on Mac OS files.

Void

This node is used to avoid the `<empty/>` syntax of XML and force the `<name></name>` syntax of SGML, which allows editing on any SGML editor as well as any XML editor.

Many elements have the content rule "Void?". However, the Void element should never be specified, thereby leaving the containing node empty.

DTD content rule

```
<EMPTY>
```

Walk-cached-property-sets

Directs the SaveAs processor to construct a stylesheet cache and walk the stylesheet data.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Use-event-list`	String	Required. The name of an event processing list (as in `<define-event-list>`) to be used to process the events generated by walking the stylesheet data (ClassMap and class information) of the document.

Walk-children

Directs the SaveAs processor to walk the kids list of the current structural element.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Use-event-list`	String	Required. The name of an event processing list (see `<define-event-list>`) to be used to process the events generated by walking the first-level children of the current structural element.

Walk-layout

This directive is not supported in this version of SaveAsXML.

Walk-metadata

Directs the SaveAsXML processor to walk the `DocInfo` metadata portion of the PDF document.

DTD content rule

```
Void?
```

Attributes

Name	Type	Description
`Use-proc-list`	String	Required. The name of an event processing list (as in `<define-proc-list>`) to be used to process the attributes found by walking the metadata portion of the document.

Walk-proplist

Directs the SaveAs processor to walk the specified generic property list (property lists owned by XML, HTML-3.20, and HTML-4.01). This is used to process arbitrary, user-supplied attributes of the current structural element.

DTD content rule

```
(Comment | Conditional-delimeter | Emit-string | Proc-property)+
```

Attributes

Name	Type	Description

`Owner`	Choice	Required. Selects the attribute list owner. One of:
		`Xml`
		`Html-3.20`
		`Html-4.01`
		`Css-1.00`
		`Css-2.00`

Walk-structure

Directs the SaveAsXML processor to walk the logical structure tree and associated content of the PDF document.

DTD content rule

`Void?`

Attributes

Name	Type	Description
`Use-event-list`	String	Required. The name of an event processing list (as in `<define-event-list>`) to be used to process the events generated by walking the structure tree of the document.

Index