# Proposal to drop bind-XX variables (bug 24510):

# detailed textual changes

revised 2014-03-14, changed paragraphs highlighted.

Section 5.4.4 (extensions to dynamic context)

In the Note below the third bullet, delete the second sentence.

In the table row for current group, delete ", calls on xsl:for-each-group with a bind-group attribute"; add ", and calls on xsl:apply-templates, xsl:call-template, xsl:apply-imports, and xsl:next-match that appear in a streaming context as described in 14.2."

In the table row for current grouping key, delete " with a bind-grouping-key attribute"; add ", and calls on xsl:apply-templates, xsl:call-template, xsl:apply-imports, and xsl:next-match that appear in a streaming context as described in 14.2.

Delete xsl:merge from the "Set by" entries for current-group() and current-grouping-key().

Add two extra components to the dynamic context (each contributing a new bullet item and a new entry in the table):

- The current merge group: this is a map. During evaluation of an xsl:merge instruction, as each group of items with equal composite merge key values is processed, the current merge group is set to a map whose keys are the names of the various merge sources, and whose associated values are the items from each merge source having the relevant composite merge key value.
- The current merge key: this is a sequence of atomic values. During evaluation of an xsl:merge instruction, as each group of items with equal composite merge key values is processed, the current merge key is set to the composite merge key value that these items have in common.

In the table, add two rows:

current merge group | absent | xsl:merge | Calls to stylesheet functions, dynamic function calls, evaluation of global variables, stylesheet parameters, and patterns, evaluation of xsl:apply-templates, xsl:call-template, xsl:apply-imports, and xsl:next-match

current merge key | absent | xsl:merge | Calls to stylesheet functions, dynamic function calls, evaluation of global variables, stylesheet parameters, and patterns, evaluation of xsl:apply-templates, xsl:call-template, xsl:apply-imports, and xsl:next-match

## Section 14.1

- Remove the bind-group and bind-grouping-key attributes of xsl:for-each-group (in the syntax template and from the XSLT 3.0 schema)

## Section 14.2

## The intro to this section (before 14.2.1) is rewritten as follows:

Two pieces of information are available during the processing of each group (that is, while evaluating the sequence constructor contained in the `xsl:for-each-group` instruction, and also while evaluating the sort key of a group as expressed by the `select` attribute or sequence constructor of an `xsl:sort` child of the `xsl:for-each-group` element):
[Definition: The **current group ~~value~~** is the group itself, as a sequence of items].

[Definition: The **current grouping key ~~value~~** is a single atomic value, or in the case of a composite key, a sequence of atomic values, containing the grouping key of the items in the current group ~~value.~~]

~~There are two ways of getting this information. The preferred way in XSLT 3.0 is to bind variables using the `bind-group` and `bind-grouping-key` attributes of the `xsl:for-each-group` instruction.~~

~~If the `bind-group` attribute is present, then its value must be an EQName, and this causes a local variable binding for this name to be visible within the sequence constructor forming the body of the `xsl:for-each-group` instruction, and also within any `xsl:sort` element child of the `xsl:for-each-group` element. The type of the variable is `item()*` (any sequence of items), and its value is the content of the current group value.~~

~~If the `bind-grouping-key` attribute is present, then its value must be a EQName, and this causes a local variable binding for this name to be present within the sequence constructor forming the body of the `xsl:for-each-group` instruction and also within any `xsl:sort` element child of the `xsl:for-each-group` element. The type of the variable is `anyAtomicType*` (any sequence of atomic values), and its value is the current grouping key value, that is the grouping key of the group being processed.~~

~~If the variable names bound in the `bind-group` or `bind-grouping-key` attributes are used in the `select` attribute or the sequence constructor within an `xsl:sort` child of the `xsl:for-each-group` instruction, then they act as references to the group whose sort key is being computed, or the grouping key of that group, respectively.~~

~~Except as noted below, the variable bindings established by the `bind-group` and `bind-grouping-key` attributes, when present, are visible within all descendant elements of the `xsl:for-each-group` instruction on which they are declared, other than elements where the variable binding is shadowed by another variable binding. For more information see *9.9 Scope of Variables*.~~

~~[ERR XTSE3220] It is a static error if a variable bound in the bind-group or bind-grouping-key attribute of an xsl:for-each-group instruction is referenced within an expression in the lang, order, collation, stable, case-order, or data-type attributes of an xsl:sort child of that xsl:for-each-group instruction.~~

~~[ERR XTSE3230] It is a static error if the bind-grouping-key attribute is present on an xsl:for-each-group instruction unless either the group-by or group-adjacent attribute is present.~~

~~For backwards compatibility, XSLT 3.0 also allows~~ Information about the current group ~~value~~ and the current grouping key ~~value to be~~ is held in the dynamic context, and is available using the `current-group` and `current-grouping-key` functions respectively.

In XSLT 2.0, the current group and current grouping key were passed unchanged through calls of xsl:apply-templates and xsl:call-template, and also xsl:apply-imports and xsl:next-match. This behaviour is retained in XSLT 3.0 except in the case where streaming is in use: specifically, if the xsl:apply-templates, xsl:call-template, xsl:apply-imports, or xsl:next-match instruction occurs either as a descendant of an xsl:stream instruction, or within a template rule in a streamable mode, then the current group and current grouping key are set to absent in the called template. The reason for this is to allow the streamability of an xsl:for-each-group instruction to be assessed statically, as described in section 19.8.4.19.

~~The difference between using bound variables and using these functions is that the variables have static scope (they can only be used lexically within the `xsl:for-each-group` element), whereas the functions have dynamic scope (they are available in called templates — though not in called functions — as well as within the lexical body of `xsl:for-each-group`). The fact that the functions have dynamic scope makes certain optimizations difficult, and in particular it makes it impossible to satisfy the rules for streamability. When streamed processing is required, therefore, it is necessary to bind variables to the group and grouping key rather than using the `current-group` and `current-grouping-key` functions.~~

~~**Note:**~~

~~The terms current group value and current grouping key value refer to the group and grouping key being processed, regardless whether these are bound to variables or held in the dynamic context. The terms current group and current grouping key refer to the values held in the dynamic context, which are set to hold the current group value and current grouping key value only when these values have not been bound to variables.~~

~~An added benefit of using the `bind-group` and `bind-grouping-key` variables is apparent when `xsl:for-each-group` elements are nested: the grouping variables for the outer instruction remain in scope when processing the inner instruction.~~

~~If the `bind-group` attribute is present on the `xsl:for-each-group` instruction, then the current group (the value accessed by the `current-group` function) is set to absent during the processing of the instruction, which has the effect that any call on `current-group` results in a dynamic error.~~

~~If the `bind-grouping-key` attribute is present on the `xsl:for-each-group` instruction, or if neither the `group-by` nor `group-adjacent` attribute is present, then the current grouping key (the value accessed by the `current-grouping-key` function) is set to absent during the processing of the instruction, which has the effect that any call on `current-grouping-key` results in a dynamic error.~~

Section 14.2.1 (current-group()), Rules, para 3, replace the third sentence by "It is also cleared by a call on xsl:apply-templates, xsl:call-template, xsl:apply-imports, or xsl:next-match that appears in a streaming context (specifically, within an xsl:stream instruction, or within a template rule belonging to a streamable mode)."

Section 14.2.2 (current-grouping-key()), Rules, para 3, delete "or bind-grouping-key", and add: "It is also cleared by a call on xsl:apply-templates, xsl:call-template, xsl:apply-imports, or xsl:next-match that appears in a streaming context (specifically, within an xsl:stream instruction, or within a template rule belonging to a streamable mode)."

Section 14.4: rewrite the examples to use current-group() and current-grouping-key()

Section 15: in the example "Merging all the files in a collection", delete the bind-group attribute, and replace the variable reference $group by the function call current-merge-group(). Similarly, in the example "Merging two heterogeneous files".

Section 15.2. Remove the bind-group and bind-key attributes of xsl:merge from the syntax summary and the XSLT 3.0 schema. Delete the paragraphs that explain these variables.

Section 15.3 (xsl:merge-source)

Remove the bind-source attribute of xsl:merge-source from the syntax summary and the XSLT 3.0 schema. Delete the paragraph that explains this variable.

Add an optional name attribute whose value is an NCName, and associated rules:

The name attribute provides a means of distinguishing items from different items from different merge sources within the xsl:merge-action instructions. If the name attribute is present on an xsl:merge-source element, then it must not be equal to the name attribute of any sibling xsl:merge-source element. If the name attribute is absent, then an implementation-dependent name, different from all

explicitly specified names, is allocated to the merge source.

Section 15.4 Fix the merging example

Section 15.6.

Delete four paragraphs starting from "The static context for the sequence constructor contained", and the paragraph starting "The variable defined in the bind-source attribute".

Fix the example.

Add subsections to define the current-merge-group and current-merge-key functions.

Add: 15.6.1 fn:current-merge-group()

Summary
Returns the group of items currently being processed by an xsl:merge instruction

Signatures
current-merge-group() as item()*
current-merge-group($source as xs:string) as item()*

Properties
determistic, context-dependent, focus-independent

Rules
The current merge group is bound during evaluation of the xsl:merge instruction. If no xsl:merge-group instruction is being evaluated, the current merge group will be absent, that is, any reference to it will cause a dynamic error.

The current merge group is a map. It contains the set of items, from all merge inputs, that share a common value for the merge key. This is structured as a map so that the items from each merge source can be identified. The key in the map is the value of the name attribute of the corresponding xsl:merge-source element, or a system-allocated name for the merge source if the name attribute is absent.

If $G is this map, then:

- the single-argument form of this function returns the value of the expression `if (map:contains($G, $source)) then $G($source) else error()`. Informally, if there is an xsl:merge-source element whose name attribute matches $source, then it returns the items in the current merge group that are contributed by this merge source; otherwise it raises a dynamic error (see below).

- the zero-argument form of the function returns the value of the expression `map:keys($G)!$G(.)`. Informally, it returns all the items in the current merge group regardless of which merge source they derive from.

Notes

Because the current merge group is cleared by function calls and template calls, the current-merge-group() function only has useful effect when the call appears as a descendant of an xsl:merge-action element.

If an xsl:merge-source element has no name attribute, then it is not possible to discover the items in the current merge group that derive specifically from that source, but these items will still be present in the current merge group and will be included in the result when

current-merge-group is called with no arguments.

Error Conditions
The function raises a dynamic error if the current merge group is absent.
The function raises a dynamic error if the value of $source does not match any xsl:merge-source element.

Notes
As for fn:current-group

[Aside. In examples, where there is currently a reference to a variable declared in bind-source, e.g. bind-source="master", the reference $master changes to current-merge-group("master"). Where there is currently a reference to a variable declared in bind-group at the xsl:merge level, it changes to current-merge-group()]

Add 15.6.2 fn:current-merge-key

Summary
Returns the composite merge key of the items currently being processed using the xsl:merge instruction

Signature
current-merge-key() as xs:anyAtomicType*

Properties
determistic, context-dependent, focus-independent

Rules
The evaluation context for XPath expressions includes a component called the current merge key, which is a sequence of atomic values. The current merge key is the composite merge key shared in common by all the items within the current merge group.

The current merge key is bound during evaluation of the xsl:merge instruction. If no xsl:merge instruction is being evaluated, the current merge key will be absent, which means that any reference to it causes a dynamic error.

Because the current merge key is cleared by function calls and template calls, the current-merge-key() function only has useful effect when the call appears as a descendant of an xsl:merge-action element. Within that element, it identifies the common value for the composite merge key shared by all items in the current merge group.

Error Conditions
As for fn:current-grouping-key

Notes
As for fn:current-grouping-key

Section 15.7

Fix the examples

Section 18.1.2

Fix the exmaple titled "Using xsl:stream with xsl:for-each-group".

Section 19

In the para starting "The rules in this section generally...", delete the last sentence (referring to bind-XXX attributes)

In the final paragraph, delete the text starting from "with one exception:...".

Section 19.8.4.19 (Streamability of xsl:for-each-group)

Delete rules 5 and 6.

Rewrite the Note as follows:

The above rules do not explicitly mention any constraints on the presence or absence of call on the current-group function. In practice, however, this plays an important role. In the most common case, the `select` expression of `xsl:for-each-group` is likely to be striding, for example an expression that selects all the children of a given element. Any call on current-group associated with this xsl:for-each-group instruction will ordinarily be striding and consuming, which is consistent with streaming provided there is only one such call, and if it appears in a suitable context (for example, not within a predicate). If there is more than one call, or if it appears in an unsuitable context (for example, within a predicate), then this will have the same effect as multiple appearances of other consuming expressions: the construct as a whole will be free-ranging. These rules are not spelled out explicitly, but rather emerge as a consequence of the general streamability rules.

Section 19.8.4.35 Streamability of xsl:stream

Change rule 1 to:

If the contained sequence constructor contains, at any depth, a call on the current-group function whose nearest containing `xsl:for-each-group` instruction is itself an ancestor of the `xsl:stream` instruction, then roaming and free-ranging.

If the contained sequence constructor contains, at any depth, a call on the current-merge-group ~~or current-merge-source-group~~ function whose nearest containing `xsl:merge` instruction is itself an ancestor of the `xsl:stream` instruction, then roaming and free-ranging.

Section 19.8.7.10

All variable references are now grounded and motionless.

Section 19.8.7.14

All inline function declarations are now grounded and motionless.

Section 19.8.8.4 Streamability of the current-group function

The rules are taken from the existing rules for bind-group variable references in 19.8.7.10.

Section 19.8.8.5 Streamability of the current-grouping-key function

The function is grounded and motionless
Section 19.8.9 Classifying Patterns

Delete rule 3 ("The pattern does not contain...."). (Note: current-group and current-grouping key are banned within patterns).

Section 19.9, "Streamed Grouping" example

Fix the example, including the analysis.

Add sections for streamability of current-merge-group and current-merge-key.

The rules for current-merge-group() are taken from the existing rules for bind-group and bind-source variable references in 19.8.7.10.

A call to current-merge-key() is grounded and motionless.