This Way

ConTEXt magazine #1103 MkIV
September 2011

Cross document referencing

Hans Hagen
PRAGMA ADE

Other documents

A straightforward way to refer to something in an other document is by prefixing the reference by a document tag. Take for instance:

```
\in{chapter}[other::whatever]
```

Here, `other` is either a tag or a filename. In the case if a tag, you also need a definition like:

```
\useexternalfile[other][somefilename]
```

Because we load the references of the other file (when present), you can also ask for titles of chapters. In fact, all the following work:

```
\at      {page}[other::whatever]
\in    {chapter}[other::whatever]
\about          [other::whatever]
\goto{location}[other::whatever]
```

given of course that in the other file we have set a reference:

```
\startchapter[reference=whatever,title={Who cares}]
  ...
\stopchapter
```

In MkIV this mechanisms has been extended to deal with products and components. In order not to get clashes between references in multiple chapters, you can do something like this:

```
\setuphead[chapter][referenceprefix=whatever]
```

This will create a namespace for this chapter. A more automated alternative is:

```
\setuphead[chapter][referenceprefix=+]
```

Here the given reference (`whatever`) will automatically become the namespace for that chapter.

Products and components

This is however somewhat cumbersome when we deal with a project structure. There we have the complication that we can process components within a product and although one will only do this for proofing it makes sense at least to deal with references in other components.

In the test suite there are four files demonstrating what is possible. They can be recognized by the name `cross-*.tex`. The product file `cross-100` includes two components:

```
\startproduct cross-100

    \component cross-001
    \component cross-002

\stopproduct
```

In these components there are references to the other component. The cross reference mechanism will automatically use the component's name as namespace but only when you say:

```
\setupreferencing[autofile=yes]
```

A component looks as follows:

```
\setupreferencing[autofile=yes]
\setupinteraction[state=start]

\startcomponent cross-001

\product cross-100

\startchapter[title=One,reference=one]
  ...
\stopchapter

\stopcomponent
```

When a component is processed, the references of the product are also loaded. Actually, some more information fetched so that for instance the chapter number gets set as well as the page number.

Of course this will not guarantee that all referencing turns out right, but it's better than nothing. There are now several ways to refer to something, and as we have quite some fallback heuristics in place all the following will work out well. However, keep in mind that when multiple one's are uses you might end up with the wrong one when no prefix is given.

```
\at      {page}[one]
\in   {chapter}[one]
\about         [one]
\goto{location}[one]

\at      {page}[cross-001:one]
\in   {chapter}[cross-001:one]
\about         [cross-001:one]
\goto{location}[cross-001:one]

\at      {page}[cross-001::one]
\in   {chapter}[cross-001::one]
```

```
\about          [cross-001::one]
\goto{location}[cross-001::one]

\at       {page}[cross-001:::one]
\in   {chapter}[cross-001:::one]
\about          [cross-001:::one]
\goto{location}[cross-001:::one]
```

So what do the (subtle) differences in colons mean? The `cross-001:` prefix is just a prefix. Such a prefix is not always related to a document but it happens that when no other match is found, an extra check takes place to see if it is a component namespace. This is new per September 2011.

The `cross-001::` prefix is the official way to refer to another document and this is no news. However, the `cross-001:::` prefix is new and depending on how the document is run, is either a regular namespace prefix (one colon) or an external reference (two colons). When you use the project structure this might be the best way to go. The reason is that order of looking (and fallbacks) is better defined this way.

So, given that you have a proper usage of product and components, the following method is to be preferred:

```
\at     {page}[other:::one] in \from[other]
\in   chapter}[other:::one] of \from[other] (\about[other:::one])
\goto{details}[other:::one]
```

Keep in mind that in most cases a combination of components and extra prefixes (that is, explicitly set prefixes) work ok. The prefixing mechanism is controlled with:

```
\setupreferencing[prefix=blabla]
```

but you will seldom need this command. In order to prevent clashes you can best use some redundancy:

```
\placefigure[here][fig:foo]{}{}{}
\placetable [here][tab:foo]{}{}{}
```

works out quite well.

Reference commands

In MkII the main reference mechanism handled not only user references but also stored section numbers, section titles, captions and all that made sense to refer to. In MkIV we carry around way more information and references are stored in and retrieved from several data structures. Although we keep much more information in memory and store more information in the auxiliary file, we save some too because now (for instance) section titles are stored only once.

The following two commands store an explicit reference, unrelated to a structural component. However, with the page number we also store information about the current section so that we can add a prefix any time we want.

```
\textreference[sometag]{some text}
\pagereference[sometag]
```

Keep in mind that these commands insert a so called node so they can best be attached to some content in order not to dangle around and interfere with spacing. The following works okay:

```
\dontleavehmode\textreference[ward]{Quoting Ward}\input ward
```

A rather low level (not interactive) fetching can be done as follows:

```
\ref[text][sometag]
\ref[page][sometag]
```

We already saw some more advanced commands to retrieve reference data:

```
\at      {page}[one]
\in   {chapter}[one]
\about         [one]
\goto{location}[one]
```

These commands will create a hyperlink when interactivity is turned on.

The \at command typesets the page number and the \in command typesets a number. The \about command deals with the title. In the case of a regular reference the last two commands do a similar thing but the last one adds quotes (by default). The \goto command only has a meaning in interactive documents. It does not add anything to the text.

In interactive mode all these commands will apply a so called contrast color in case the reference refers to the page itself.

There are two commands that relate to current location:

```
\somewere{before}{current}{after}[one]
\atpage[one]
```

The first command typesets one of the three texts, which one depends of the typeset and referred one being on the same page. The second command generates a text automatically.

Although not related to the kind of references we discuss here, you can define symbolic references with:

```
\definereference[symbolic name][real reference]
\resetreference[symbolic name]
```

Using this only makes sense in interactive documents where we can have special operations with arguments and combinations of such references.

Reference formats

You can control the formatting of references in detail using the setup command. For instance you can tweak the way sections numbers are prefixed but as this relates to numbering this will not be discussed here. Reference formats are another way to control the rendering

```
\definereferenceformat[informula]  [left=(,right=),text=formula]
\definereferenceformat[informulas] [left=(,right=),text=formulas]
\definereferenceformat[andformula] [left=(,right=),text=and]
\definereferenceformat[andformulas][left=(,right=),text=and]

\informula [b] and \informula [for:c]
the \informula {formulas}[b] \informula {and} [for:c]
the \informulas {formulas}[b] \informula {and} [for:c]
the \informulas [b] \informula {en} [for:c]
the \informulas [b] \andformula [for:c]
```

Instead of a text, one can specify a label, which should be defined with `\setuplabeltext`.

User references

You can create user references too. This is done with the following command:

```
\setreference[myref][key-1=value-1,key-2=value-2]
```

You can then ask for keys using:

```
\getreference[myref][key-2]
```

In principle you can add filters and rendering variants as well using Lua code but that is rather specialized and often not needed.

source code of this document

```
% language=uk

% author    : Hans Hagen
% copyright : PRAGMA ADE & ConTeXt Development Team
% license   : Creative Commons Attribution ShareAlike 4.0 International
% reference : pragma-ade.nl | contextgarden.net | texlive (related) distributions
% origin    : the ConTeXt distribution
%
% comment   : Because this manual is distributed with TeX distributions it comes with a rather
%             liberal license. We try to adapt these documents to upgrades in the (sub)systems
%             that they describe. Using parts of the content otherwise can therefore conflict
%             with existing functionality and we cannot be held responsible for that. Many of
%             the manuals contain characteristic graphics and personal notes or examples that
%             make no sense when used out-of-context.

\usemodule[mag-01,abr-02]

\startbuffer[abstract]
    The (cross) reference mechanism in \CONTEXT\ is rather complex (in terms of
    code) and provides a lot of functionality. Of course one can ask for page
    numbers, section numbers, titles, or arbitrary text, but also control the
    viewer, go to locations and have chains of actions. In this document we only
    discuss some aspects of cross document referencing. This is not a complete
    manual.
\stopbuffer

\startdocument
  [title={Cross document referencing},
   author=Hans Hagen,
   affiliation=PRAGMA ADE,
   date=September 2011,
   number=1103 \MKIV]

\subject{Other documents}

A straightforward way to refer to something in an other document is by prefixing
the reference by a document tag. Take for instance:

\startscite[tex]
\in{chapter}[other::whatever]
\stopscite

Here, \type {other} is either a tag or a filename. In the case if a tag, you also
need a definition like:

\startscite[tex]
\useexternalfile[other][somefilename]
\stopscite

Because we load the references of the other file (when present), you can also ask
for titles of chapters. In fact, all the following work:

\startscite[tex]
\at      {page}[other::whatever]
\in   {chapter}[other::whatever]
\about         [other::whatever]
\goto{location}[other::whatever]
\stopscite

given of course that in the other file we have set a reference:
```

source code of this document

```
\startscite[tex]
\startchapter[reference=whatever,title={Who cares}]
  ...
\stopchapter
\stopscite

In \MKIV\ this mechanisms has been extended to deal with products and components.
In order not to get clashes between references in multiple chapters, you can do
something like this:

\startscite[tex]
\setuphead[chapter][referenceprefix=whatever]
\stopscite

This will create a namespace for this chapter. A more automated alternative is:

\startscite[tex]
\setuphead[chapter][referenceprefix=+]
\stopscite

Here the given reference (\type {whatever}) will automatically become the
namespace for that chapter.

\subject{Products and components}

This is however somewhat cumbersome when we deal with a project structure. There
we have the complication that we can process components within a product and
although one will only do this for proofing it makes sense at least to deal with
references in other components.

In the test suite there are four files demonstrating what is possible. They can
be recognized by the name \type {cross-*.tex}. The product file \type {cross-100}
includes two components:

\startscite[tex]
\startproduct cross-100

    \component cross-001
    \component cross-002

\stopproduct
\stopscite

In these components there are references to the other component. The cross
reference mechanism will automatically use the component's name as namespace but
only when you say:

\startscite[tex]
\setupreferencing[autofile=yes]
\stopscite

A component looks as follows:

\startscite[tex]
\setupreferencing[autofile=yes]
\setupinteraction[state=start]

\startcomponent cross-001

\product cross-100

\startchapter[title=One,reference=one]
  ...
```

source code of this document

```
\stopchapter

\stopcomponent
\stopscite

When a component is processed, the references of the product are also loaded.
Actually, some more information fetched so that for instance the chapter number
gets set as well as the page number.

Of course this will not guarantee that all referencing turns out right, but it's
better than nothing. There are now several ways to refer to something, and as we
have quite some fallback heuristics in place all the following will work out
well. However, keep in mind that when multiple \type {one}'s are uses you might
end up with the wrong one when no prefix is given.

\startscite[tex]
\at      {page}[one]
\in   {chapter}[one]
\about         [one]
\goto{location}[one]

\at      {page}[cross-001:one]
\in   {chapter}[cross-001:one]
\about         [cross-001:one]
\goto{location}[cross-001:one]

\at      {page}[cross-001::one]
\in   {chapter}[cross-001::one]
\about         [cross-001::one]
\goto{location}[cross-001::one]

\at      {page}[cross-001:::one]
\in   {chapter}[cross-001:::one]
\about         [cross-001:::one]
\goto{location}[cross-001:::one]
\stopscite

So what do the (subtle) differences in colons mean? The \type {cross-001:} prefix
is just a prefix. Such a prefix is not always related to a document but it
happens that when no other match is found, an extra check takes place to see if
it is a component namespace. This is new per September 2011.

The \type {cross-001::} prefix is the official way to refer to another document
and this is no news. However, the \type {cross-001:::} prefix is new and
depending on how the document is run, is either a regular namespace prefix (one
colon) or an external reference (two colons). When you use the project structure
this might be the best way to go. The reason is that order of looking (and
fallbacks) is better defined this way.

So, given that you have a proper usage of product and components, the following
method is to be preferred:

\startscite[tex]
\at     {page}[other:::one] in \from[other]
\in   chapter}[other:::one] of \from[other] (\about[other:::one])
\goto{details}[other:::one]
\stopscite

Keep in mind that in most cases a combination of components and extra prefixes
(that is, explicitly set prefixes) work ok. The prefixing mechanism is controlled
with:
```

source code of this document

```
\startscite[tex]
\setupreferencing[prefix=blabla]
\stopscite

but you will seldom need this command. In order to prevent clashes you can best
use some redundancy:

\startscite[tex]
\placefigure[here][fig:foo]{}{}{}
\placetable [here][tab:foo]{}{}{}
\stopscite

works out quite well.

\subject{Reference commands}

In \MKII\ the main reference mechanism handled not only user references but also
stored section numbers, section titles, captions and all that made sense to refer
to. In \MKIV\ we carry around way more information and references are stored in
and retrieved from several data structures. Although we keep much more
information in memory and store more information in the auxiliary file, we save
some too because now (for instance) section titles are stored only once.

The following two commands store an explicit reference, unrelated to a structural
component. However, with the page number we also store information about the
current section so that we can add a prefix any time we want.

\startscite[tex]
\textreference[sometag]{some text}
\pagereference[sometag]
\stopscite

Keep in mind that these commands insert a so called node so they can best be
attached to some content in order not to dangle around and interfere with
spacing. The following works okay:

\startscite[tex]
\dontleavehmode\textreference[ward]{Quoting Ward}\input ward
\stopscite

A rather low level (not interactive) fetching can be done as follows:

\startscite[tex]
\ref[text][sometag]
\ref[page][sometag]
\stopscite

We already saw some more advanced commands to retrieve reference data:

\startscite[tex]
\at      {page}[one]
\in   {chapter}[one]
\about         [one]
\goto{location}[one]
\stopscite

These commands will create a hyperlink when interactivity is turned on.

The \type {\at} command typesets the page number and the \type {\in} command
typesets a number. The \type {\about} command deals with the title. In the case
of a regular reference the last two commands do a similar thing but the last one
adds quotes (by default). The \type {\goto} command only has a meaning in
```

interactive documents. It does not add anything to the text.

In interactive mode all these commands will apply a so called contrast color in case the reference refers to the page itself.

There are two commands that relate to current location:

```
\startscite[tex]
\somewere{before}{current}{after}[one]
\atpage[one]
\stopscite
```

The first command typesets one of the three texts, which one depends of the typeset and referred \type {one} being on the same page. The second command generates a text automatically.

Although not related to the kind of references we discuss here, you can define symbolic references with:

```
\startscite[tex]
\definereference[symbolic name][real reference]
\resetreference[symbolic name]
\stopscite
```

Using this only makes sense in interactive documents where we can have special operations with arguments and combinations of such references.

```
\subject{Reference formats}
```

You can control the formatting of references in detail using the setup command. For instance you can tweak the way sections numbers are prefixed but as this relates to numbering this will not be discussed here. Reference formats are another way to control the rendering

```
\startscite[tex]
\definereferenceformat[informula]  [left=(,right=),text=formula]
\definereferenceformat[informulas] [left=(,right=),text=formulas]
\definereferenceformat[andformula] [left=(,right=),text=and]
\definereferenceformat[andformulas][left=(,right=),text=and]

\informula [b] and \informula [for:c]
the \informula {formulas}[b] \informula {and} [for:c]
the \informulas {formulas}[b] \informula {and} [for:c]
the \informulas [b] \informula {en} [for:c]
the \informulas [b] \andformula [for:c]
\stopscite
```

Instead of a text, one can specify a label, which should be defined with \type {\setuplabeltext}.

```
\subject{User references}
```

You can create user references too. This is done with the following command:

```
\startscite[tex]
\setreference[myref][key-1=value-1,key-2=value-2]
\stopscite
```

You can then ask for keys using:

```
\startscite[tex]
\getreference[myref][key-2]
```

source code of this document

```
\stopscite

In principle you can add filters and rendering variants as well using \LUA\ code
but that is rather specialized and often not needed.

\stopdocument
```

source code of this document