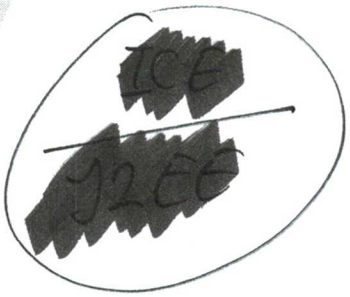
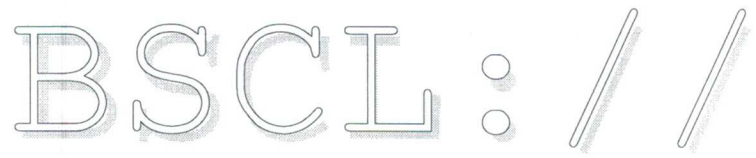
BSCL://



Requirements Specification

Version: 12/5/2000 2:58 PM

Table of contents



Introduction

This document defines and explains the functional requirements for the BackStream Content Logistics (BSCL) system. It is meant to provide the basis for the technical design and implementation of the system.

The requirements should be read by anyone who will be actively involved in the project. This includes –but is not limited to– programmers, designers, marketing, sales and management. In addition the contents of this document might be useful to potential client and investors.

BSCL is a generic content management, transformation and publishing tool. What this means is that content can be input into the system, edited and enriched, transformed into different presentations and ultimately be delivered to various subscribers.

Voor wie is BSCL bedoeld? Goeie vraag.

TODO: Erik, commentaar uit HTML nalopen en evt. overnemen.

Architecture

The BSCL consists of a number of modules, each of which provides a part of the functionality of the system (blaatzin). These modules are known as: the syndicator, the backbone and the publicator, tracking & tracing and billing. The modules work together to provide the required functionality mentioned in the introduction and described further in the remainder of this document.

Content types

BackStream discerns two types of content: textual or binary. Binary content can not be validated by the syndicator module, can not be edited without external programs and the transformations that can be performed by the publisher module are limited.

Textual content is further divided into structured XML-content and unstructured content. Both of these can be edited via browser-based forms, but unstructured content cannot be validated or transformed. BackStream uses two kinds of unstructured content: DTDs and XSL style sheets, since neither of those is normally provided as valid XML-file. Further use of unstructured content should be discouraged since it limits the possibilities of the BSCL system.

Content can also be qualified as being either local or remote. Local content is stored on the BSCL system (most likely in a database). Remote content is stored either in another BSCL system or on a non-BSCL system. For each such item a content reference is included in the local BSCL system. This reference –which is essentially a special kind of XML content– contains al information that BSCL needs to locate the remote content. This can be a URL (HTTP or FTP), a UNC-name or a BSCL://.

Syndicator

The syndicator allows the users to input their content into BSCL. It does this by allowing the user to either enter the content through a browser-based HTML form or by uploading previously prepared content.

Through the use of third party conversion tools and plug-ins it is possible to convert structured content from almost any source to XML-content which can be handled by BSCL. If the content cannot be converted to XML, it can be stored as unstructured content, as binary content or it can be left in its original location and referenced from the BSCL system by including a content reference. In the latter case BSCL can still provide tracking and tracing for the content.

Backbone

The Backbone is used to manage many of the internal processes and lookup lists of BSCL. You can use it to maintain lists of acceptable keywords (vocabularies) for different parts of the system, make sure that the size of the content database stays under control, add new users and give them access to specific parts of the system or check whether the publishing process can handle all subscriptions in a timely fashion.

Publicator

The publicator is the module that delivers the content to the subscribers. The publisher uses use the publicator to determine offers (packages of content offered to subscribers), add new subscribers and manage subscriptions.

Tracking & tracing

Wat doet de tracking & tracing module? Dit is een rapportage-module op de log die door de overige modules wordt bijgehouden.

Billing

Wat doet de billing module?

Keywords

A large part of the value of the Content Logistics solution is in the ability to offer or subscribe to just the contents that you're interested in. When you compare this to the current day search engines, you'll notice that it's almost impossible to find just the content you're looking for. You either get numerous hits in response to your query or you'll get none.

The BSCL allows both syndicators and publishers to enrich the content by specifying keywords that apply to a piece of content. The system maintains a set of allowed keywords, thus limiting the chance of polluting the keywords tree.

A typical search engine will treat keywords as just that: key**words**. In BSCL a keyword is treated like a semantic entity. This means that the BSCL should be able to tell the difference between the former U.S. president Clinton and –for example- a city named Clinton. Of course such intelligent behaviour is only possible if the system is presented with enough additional details, like the user asking for "president Clinton".

A tree most easily represents the way in which BSCL stores its keywords and their relations. This means that each keyword is identified by its path, not just by its name. So the above example of the president and the city Clinton, could be presented by the following two paths:

\Politics\USA\Presidents\Clinton

\Geographic\USA\Cities\Clinton

As you can see, both keywords are now easily distinguished by their path (which describes their location in the tree). Matching the user query for "president Clinton" to the correct node is now a simple task for the system.

TODO: Iets vertellen over "scoring". We werken niet met AND/OR en quoted strings, maar zorgen gewoon dat de quoted string hoger scoort dan een AND die weer hoger scoort dan een OR. Dat is volkomen logisch en veel eenvoudiger voor de gebruiker.

TODO: Uitleggen over relaties tussen keywords. Vnl. synoniemen en antoniemen.

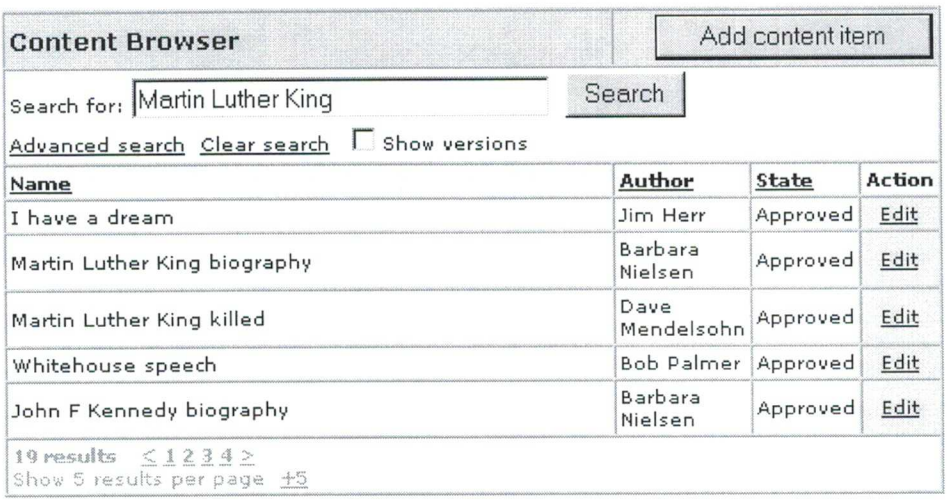
gewicht aan keyword

Syndicator

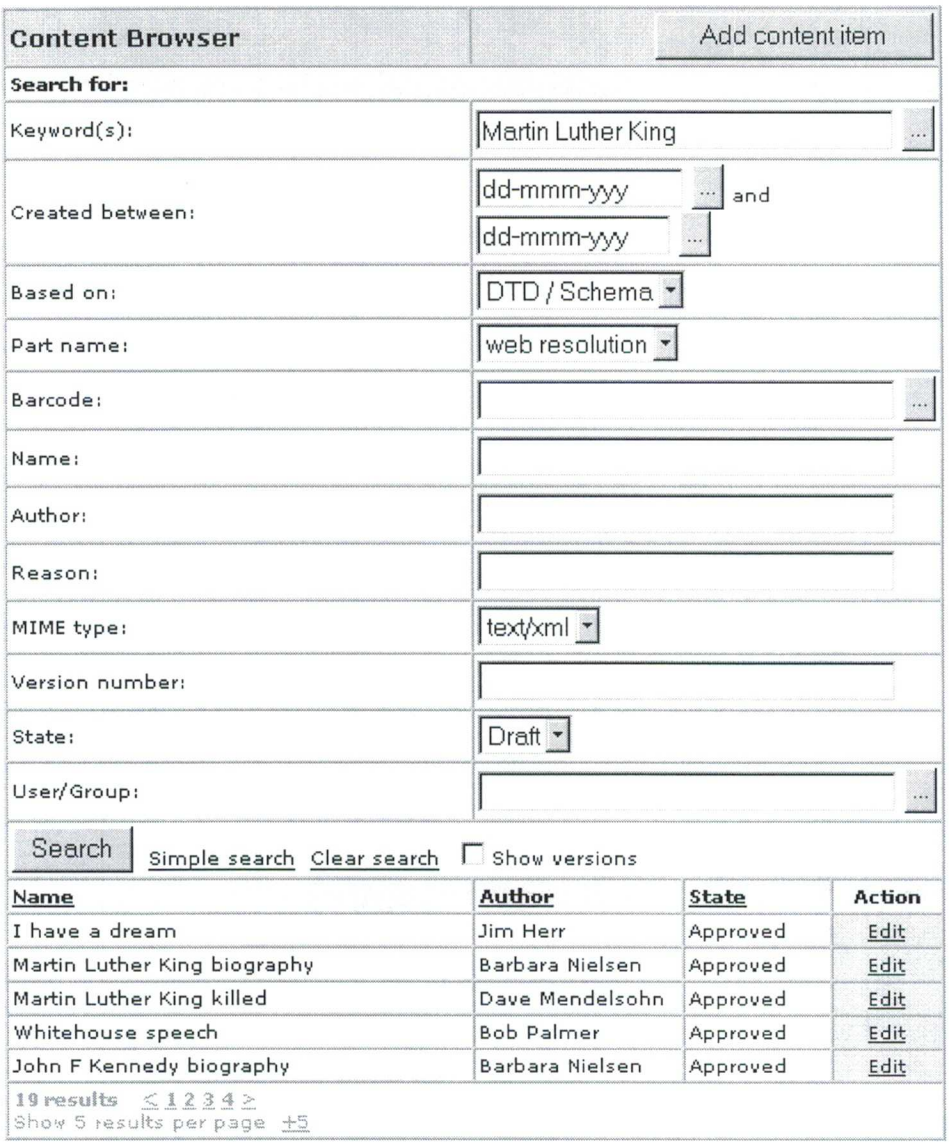
The syndicator handles the input side of BSCL. It allows the content contributor to enter or edit the content in a browser-based form or to provide new or updated content asynchronously by either e-mail or FTP. After the content has been entered in the system, it can be further enriched, updated or aggregated from this module.

Content browser

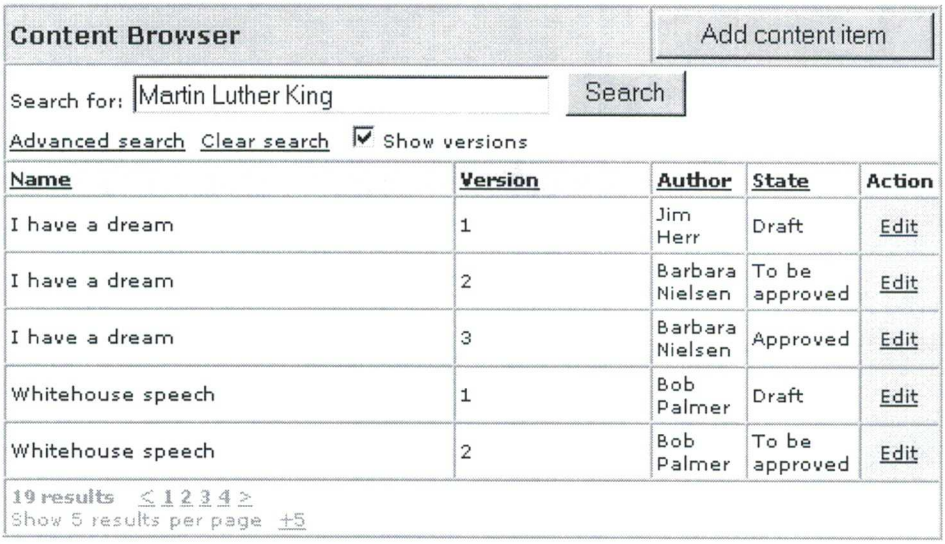
Upon start-up the user is presented with a list of the content items to which he or she has access.



The content browser also allows the user to browse through and search the content (a la BackStream). The illustration above shows the so-called simple search. When the user switches to advanced search, the form looks like this:



In its initial state the content browser shows each content item only once, even when multiple version of the item are available. The checkbox allows the user to opt to view all versions of the content items that are currently visible in the content browser:



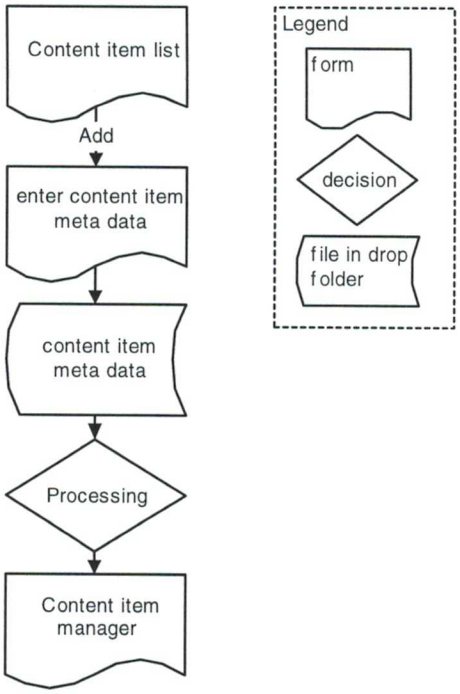
This format shows the state of the various versions, allowing a quick overview of what needs to be done to the various content versions.

TODO: Uitleggen dat de Content Browser ook kan voorkomen als een Content Selector. Zelfde scherm, maar je kiest hier een item. Deze wordt bijvoorbeeld gebruikt om een DTD te kiezen (?), een XSL/transformation, sub-content, users/groups, destinations, offers en subscriptions.

TODO: Uitleggen dat de Content Browser ook kan voorkomen met een subset van de beschikbare content, bijv. de transformation browser en de user browser.

Adding a content item

The content browser has a button that allows the content contributor to manually add a new content item to the system. The flow that is used when manually adding a new content item is depicted in the following chart:



When the user presses the "add content item" button the system presents a browser-based form, which is used to enter the content item meta-data. This includes the friendly name, the author and the type of the content item.

If the type is XML, a DTD or schema must also be selected. Optional validation and the design of the later forms as well as possible transformations in later modules are all derived from the DTD/schema that is selected in this step.

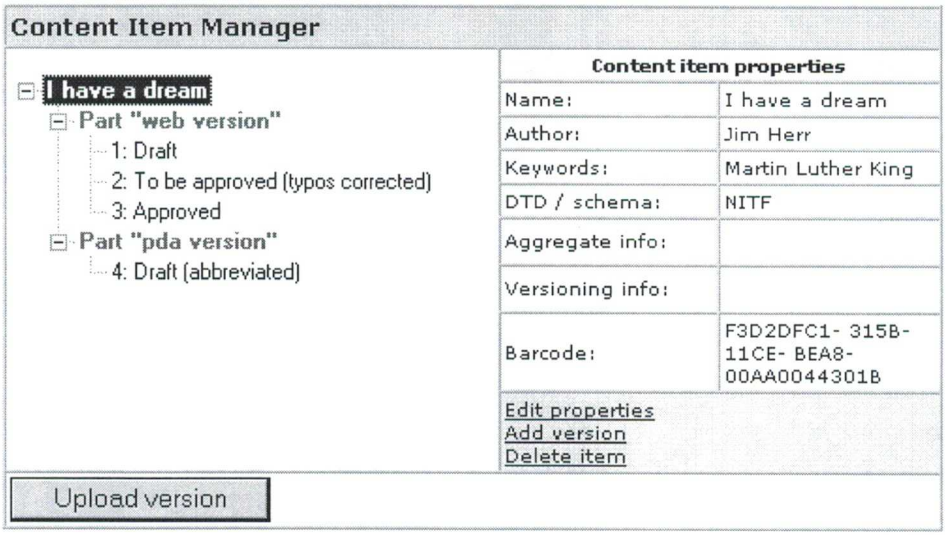
New content item	
Name:	
Content type:	XML Binary
DTD / Schema (for XML only):	NITF
Author:	
Keywords:	"Martin Luther King", "John F Kennedy ...
Custom properties:	
< Back	Next >

When the user presses the keywords button, he or she is taken to the keywords selector described below. This allows the user to associate keywords with the content item. These keywords apply to all version of the content item.

After the content item meta-data has been processed by the system, the user is taken to the content item manager where he or she can start adding versions to the content item.

Content item manager

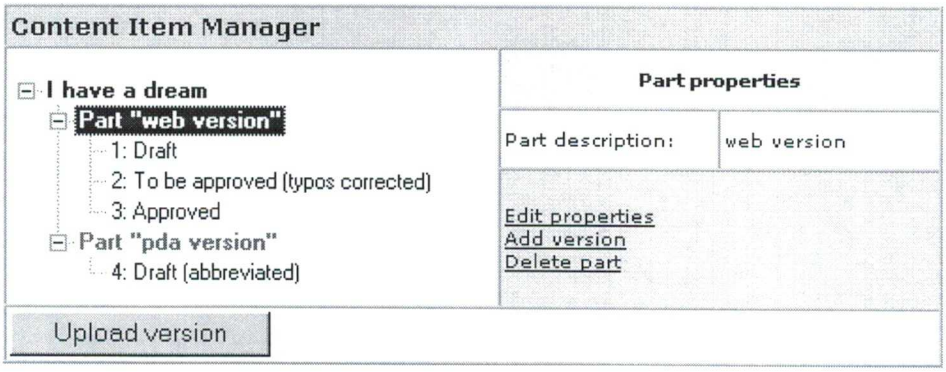
After adding a new content item or selecting a content item in the content browser, the user is taken to the content item manager.



From this content item manager the user can view, edit or delete the content item and add, edit or delete versions. The form above is presented if the user clicks on the content item in the tree on the left and shows the content item meta-data. From this form the user can edit the meta-data of the content item, delete the content item (including its versions) and add a new content version to the item. This last action will be further explained in a later section.

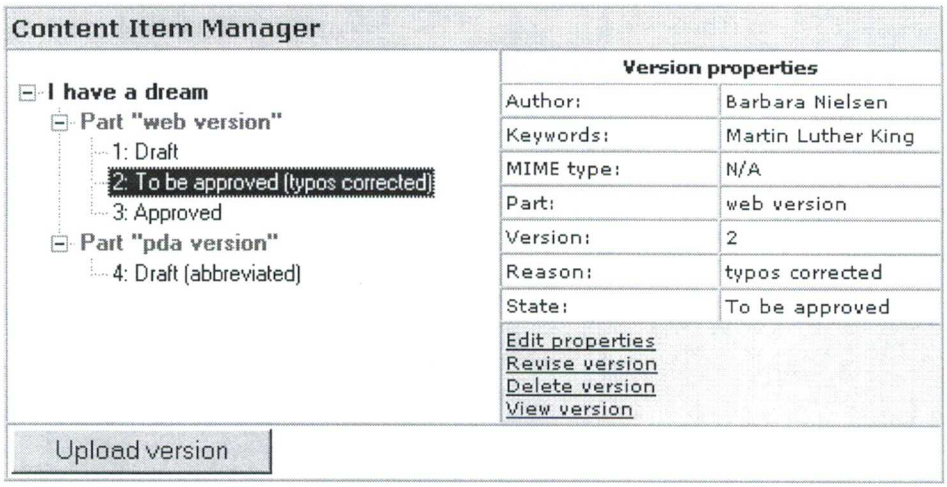
Pressing the upload button allows the user to upload a new content version. This procedure is used for binary content and if the user prepared the content locally. It is further explained below.

Selecting a content part in the tree on the left, results in the part details being displayed:



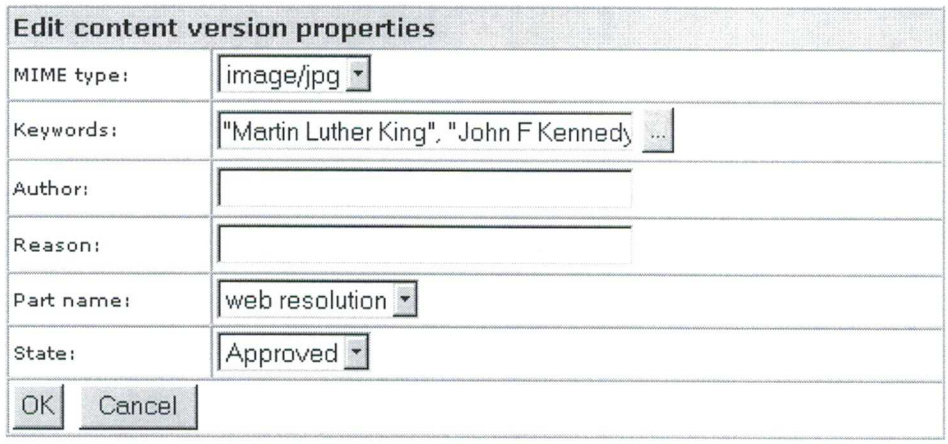
From this form the user can modify the part (which essentially moves all versions in the selected part to a different part), add a content version to the current part or delete all versions in the part.

Selecting a version in the tree, displays the meta-data for that version.



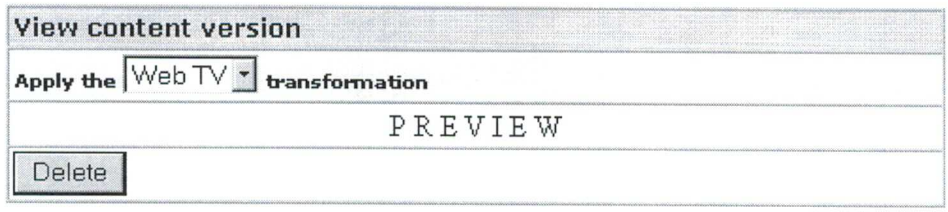
From this form the user can edit the meta-data of the content version, view or delete the version and create a new revision based on this version. Creating a revision creates a new version whose initial content is based on the currently selected version.

Clicking on edit properties shows a form that allows the user to modify the meta-data of the version:



Modifying meta-data doesn't cause a new version to be created, although the changes are written to a log file for auditing purposes. A common type of modification is the changing the status of a version. The state dropdown shows only states that can be reached from the current state by the current user.

When the user selects the view link, the selected content version is shown in a popup.



This popup allows the user to view the content, either in its native form or by applying some kind of transformation to it.

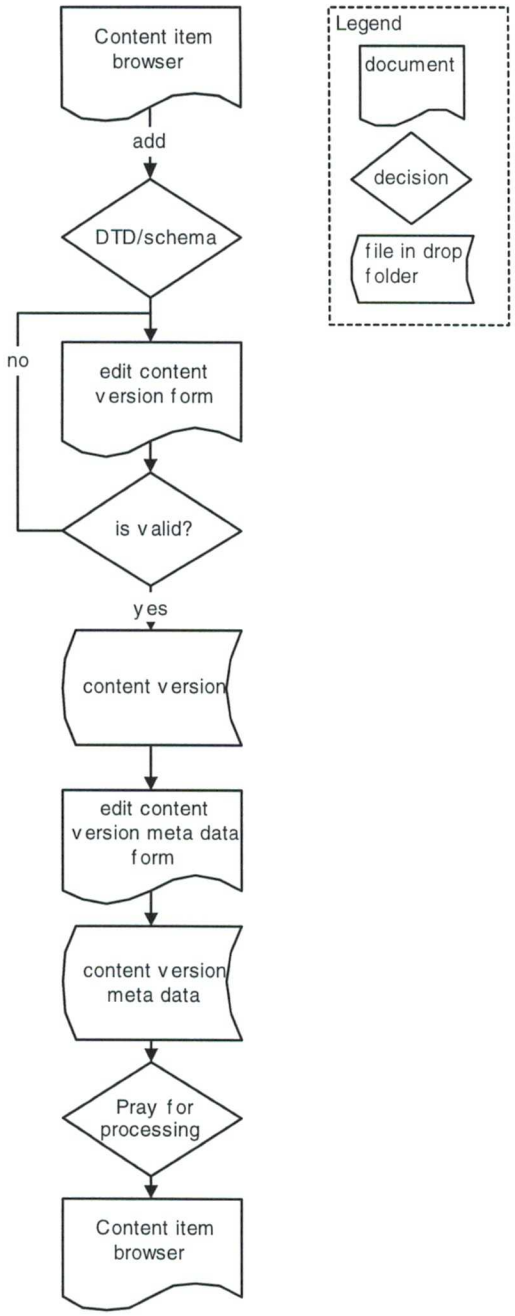
Pressing the delete button will delete this content version and close the popup window. Closing the popup window returns the user to the underlying content item manager.

Adding a content version

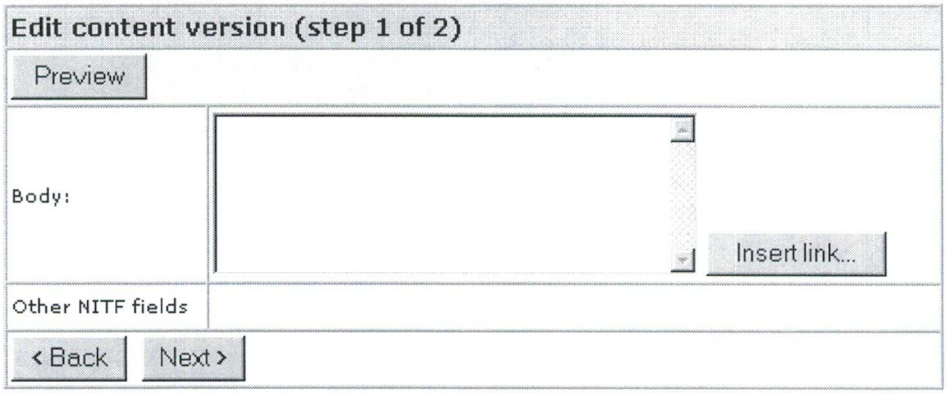
The user can start adding a new content version by selecting one of several links in the content item manager. He or she can select the add version link from the content item properties, the add version link from the part properties or the revise version from the version properties.

All these links take the user through the flow described in this section. The only difference is in the initial values of the various fields in the forms. For example: if the user adds a new version from the content item properties, not initial values will be specified. If the user chooses to revise a version, all fields will initially be set to the same as the previous version.

The flow for this process is shown in this illustration:



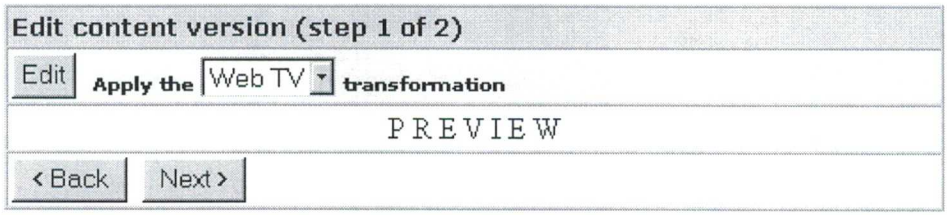
The system shows a form based on the DTD or schema that was specified in the meta-data of the content item. The form might look like this:



TODO: Preview function beschrijven en screenshot toevoegen.

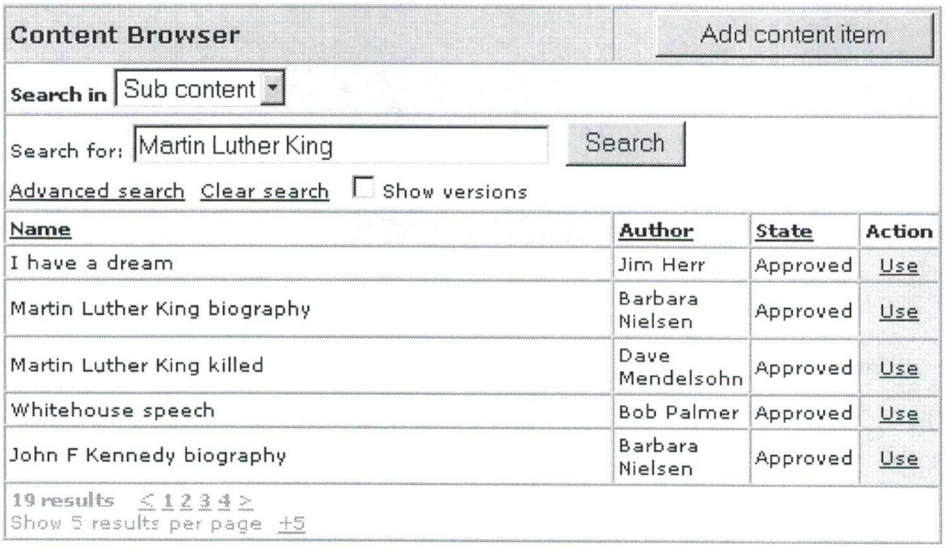
TODO: Voor veel soorten content moeten hier eigenlijk functies als spellings- en grammaticacontrole hebben.

Clicking the preview button, switches the form to preview mode:



TODO: Beschrijven

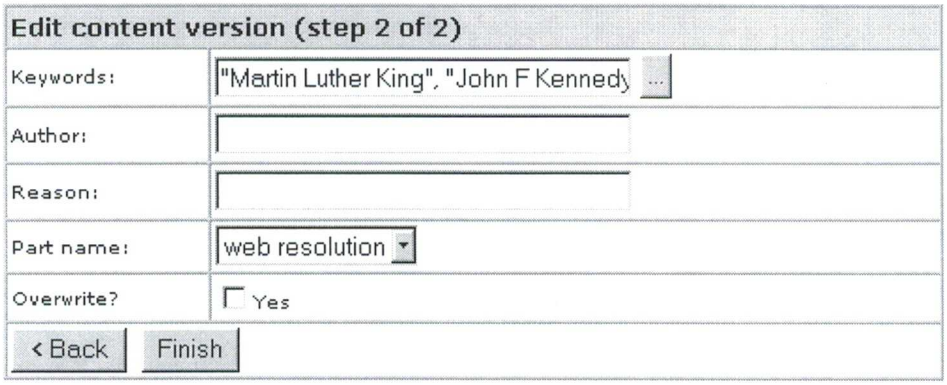
Inserting a link results in the following popup:



TODO: Beschrijven

After submitting the content is validated against the DTD or schema of the content item. If the content is not valid, the user is taken back to the previous form to correct the error.

If the content is valid, the user is presented with a form in which to enter (or modify) the meta-data for the new content version:



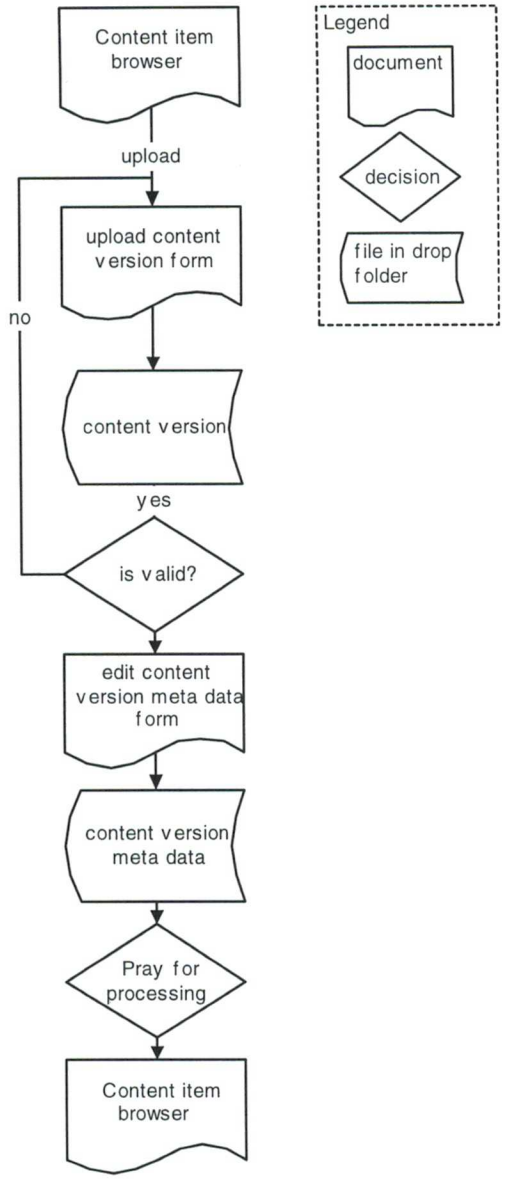
TODO: We zouden de keywords automatisch kunnen selecteren door alle woorden in de versie te proberen te vinden in de lijst van keywords. Het is immers makkelijker voor de editor om keywords te verwijderen dan om ze zelf toe te voegen.

After submitting this meta-data the content version is added to the system and the system returns to the content item manager. The tree in the content item manager will contain the newly added content version, which is automatically selected.

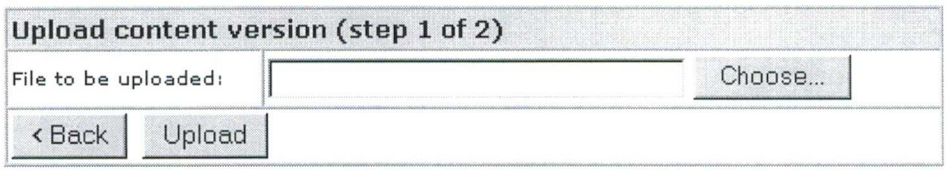
Uploading a content file

It is also possible to add a version to a content item by choosing to upload a content file from the content item manager. For textual content this might be useful if the content file was prepared locally. For binary content, uploading a file is the only way to add new versions.

When adding a new version by uploading a content file, the flow is as follows:

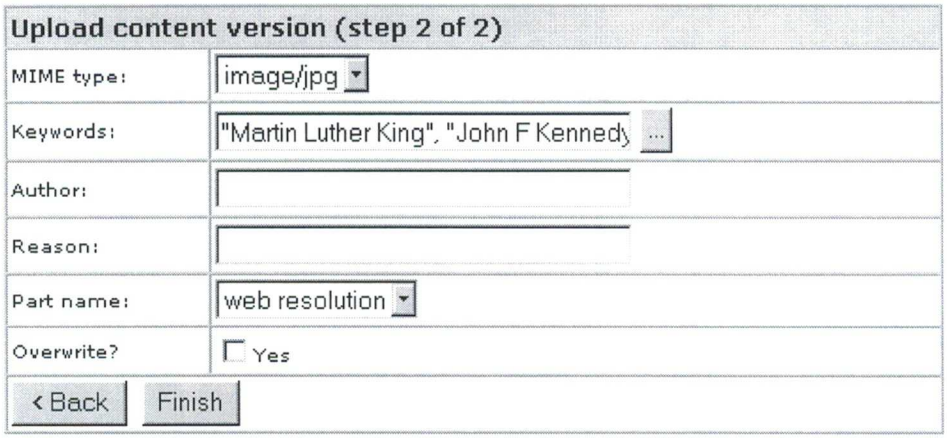


The first form allows the user to choose a local content file, which is to be uploaded to the BSCL system.



After pressing the upload button the file is transferred to the BSCL system and (in the case of XML content) validated against the DTD or schema. If the validation fails, the system returns to the form. The user should modify the local content file and upload it again.

After the content file has been accepted, the system shows a form that allows the user to enter the meta-data for the content version.



Note that this form forces the user to select a MIME type in the case of binary content. This is required since binary content is identified in BSCL not by its filename extension but by the MIME type that is specified in the meta-data of the content version. If the filename extension is present when a file is uploaded, the system will try to map the extension to a MIME type using an internal mapping list.

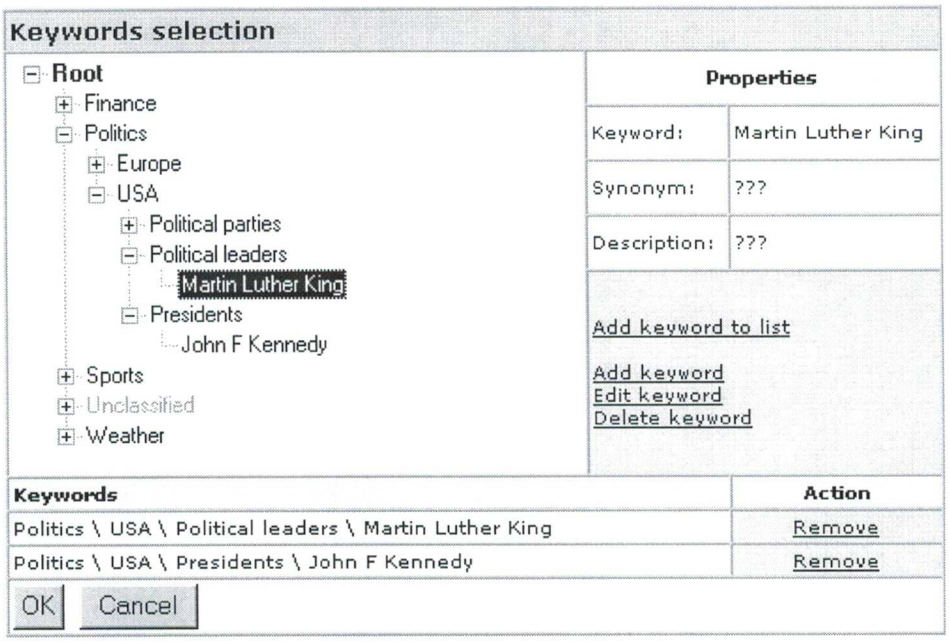
After the version meta-data has been added, the new version is added to the system. After this, the system returns to the content item manager from which the upload was started. The tree in this content item manager will contain the newly added content version, which is automatically selected.

Keyword selector

Content contributors associate keywords with content items and content versions. These keywords help publishers and subscribers to create offers and subscriptions containing just the requested content.

Keywords associated with a content item apply to all versions in the item, while keywords associated with a content version only apply to that specific version.

The associating of keywords with content is accomplished through the keyword selector:



TODO: Hier moet ook nog een search-functie in. Je wilt immers snel een keyword kunnen opzoeken terwijl je 't pad nog niet weet.

TODO: Per keyword moet de user een score kunnen aangeven. Dus: "dit artikel gaat over Clinton en Gore, maar meer over Clinton dan Gore". Deze informatie kunnen we bij searches gebruiken om de meest van toepassing zijnde resultaten bovenaan te zetten.

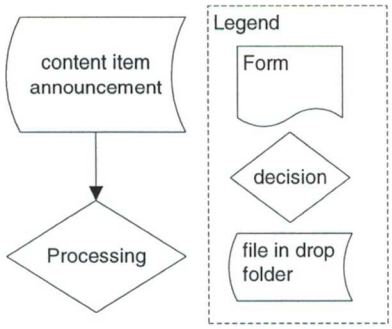
When the user selects a keyword in the tree on the left, its details are displayed on the right. By subsequently clicking "add keyword to list" the keyword is added to the list of keywords at the bottom. This list shows the keywords that are associated with this content item or version.

Uploading a content item

Content can also be added by just transferring files to the BSCL system either via FTP or by email. In this scenario –contrary to when uploading a file while adding content– there is no need to enter any data into browser-based forms. This means all content and meta-data can be prepared offline and transferred to the BSCL system when ready. This method is also typically used by data feeds.

This is accomplished is by the use of a drop folder on the BSCL system. Any files uploaded to ("dropped into") the drop folder will automatically be processed by the system. In addition to the actual content files, the drop folder may also contain announcement files. These files describe new content items and versions to the system and replace the "content item meta-data" form and the "content version meta-data" forms of the manual input.

The process of adding a new content item by uploading a file is depicted in the following chart:



The content item announcement file has the following structure:

```
<ITEM_ANNOUNCEMENT>
    <CREDENTIALS username="un" password="pw"/>
    <ITEM>
          [<BARCODE>requested barcode</BARCODE>]
          [<NAME>friendly name</NAME>]
          <CONTENT_TYPE>xml | binary</CONTENT_TYPE>
          [<SCHEMA schema="guid" [version="version"]/>]
          [<AUTHOR>author</AUTHOR>]
          [<KEYWORDS>
                <KEYWORD>keyword</KEYWORD>
          </KEYWORDS>]
          [<AGGREGATE target="guid" xslt="guid"]
          [<VERSIONING arguments…/>]
                <Note: version nr schema is configurable, see NewsML>
          [<FEEDBACK email="email-address" ftp="hostname,un,pw,type">]
    </ITEM>
    <ITEM>…</ITEM>
</ITEM_ANNOUNCEMENT>
```

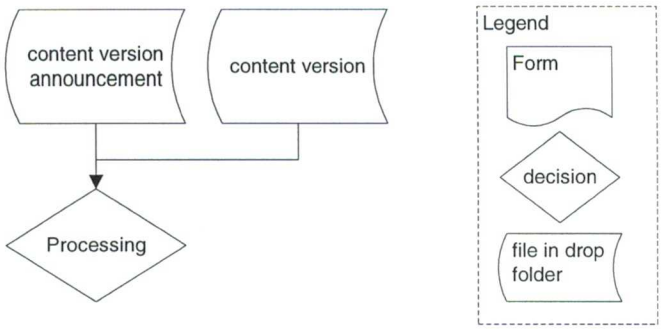
Note that the username and password in the announcement files are not encrypted and included only for identification purposes. Authentication should be used at the connection level to make content submission secure.

The user may announce the preferred barcode for the new content item in the content item announcement file.

Upon processing this file, the system adds a new content item with the meta-data from the file. After this has been completed, the announcement file is deleted from the drop folder.

Uploading a content version

After a content item has been added the content file can be uploaded, followed by a version announcement file. This process is illustrated as follows:



The version announcement file has the following structure:

```
<VERSION_ANNOUNCEMENT>
    <CREDENTIALS username="un" password="pw"/>
    <VERSION>
          <BARCODE>barcode of content item</BARCODE >
          [<PART>part description</PART>]
          <FILE>filename</FILE>
          [<REASON>description</REASON>]
          [<AUTHOR>author</AUTHOR>]
          [<KEYWORDS>
                <KEYWORD>keyword</KEYWORD>
          </KEYWORDS>]
          [<MIME_TYPE>mime type</MIME_TYPE>]
          [<OVERWRITE/>]
          [<FEEDBACK email="email-address" ftp="hostname,un,pw,type">
    </VERSION>
    <VERSION>…</VERSION>
</VERSION_ANNOUNCEMENT>
```

The system interprets the version announcement, inserts the content file and it's meta-data as a content version into the database and removes the files from the upload folder.

To assure that each file is unique on the target system it should be given a GUID-filename.

Uploading a content version with embedded content

If a content version was prepared outside of BSCL, it will most likely not use GUID file names. This can present a problem if the content references several other pieces of sub content (e.g. images), which should also be added to the BSCL.

This problem and its solution are best illustrated with an example:

The content contributor locally prepared an article on Martin Luther King, `mlking.xml`. This article references two local files, binary files: `mlking.jpg`, which contains a picture of Mr. King, and `ihada.wav`, which contains a sound byte to accompany the article.

The original content file looks like this:

```
<ARTICLE>
    <P>I am happy to join with you today in…</P>
    <P>Five years ago, a great America… </P>
    <IMAGE>images/mlking.jpg</IMAGE>
    <AUDIO>audio/ihada.wav</AUDIO>
</ARTICLE>
```

In this case it is necessary to prepare the files by first renaming the two image files to a GUID filename. These files can then be added by following the procedures outlined in "uploading a content item" and "uploading a content version".

Since the image files have been renamed, the `article.xml` needs to be modified to refer to the new filenames.

```
<ARTICLE>
    <P>I am happy to join with you today in...</P>
    <P>Five years ago, a great America... </P>
    <IMAGE>F3D2DFC1-315B-11CE-BEA8-00AA0044301B.jpg</IMAGE>
    <AUDIO>A227D147-158A-09A3-0001-34AD73221B0E.wav</AUDIO>
</ARTICLE>
```

Note that the references to subdirectories have been removed from the filenames, since all files are stored in a single drop folder on the BSCL system. This should never pose a problem, since the GUID filename will always uniquely identify a single file in the drop folder.

Also note that the filename extensions have left intact in the GUID filenames. Doing so allows BSCL to determine the MIME type from the filename and allows the file to keep working on the client side. If a filename extension had been removed the MIME type should have been specified in the corresponding version announcement file.

Aggregated content

Content can also be aggregated which means it is merged with existing content. For instance a single value of a stock market symbol can be appended to a content item already containing previous values. The new content (aggregatee) is merged with the existing aggregate content using an XSLT transformation.

For example if we've got the following XML-file containing a snapshot of some quotes.

```
<QUOTE when="date time">
    <STOCK symbol="MSFT">
    100.7
    </STOCK>
    <STOCK symbol="CSCO" >
    205.4
    </STOCK>
</QUOTE>
```

Single versions (quotes) of this content item have little meaning after their initial creation. Therefore it makes sense to disable versioning of this content item and make older stock quotes available in a single history item per symbol.

```
<STOCK_HISTORY symbol="MSFT">
    <QUOTE when="date time">
    100.6
    </QUOTE>
    <QUOTE when="date time">
    100.5
    </QUOTE>
</STOCK_HISTORY>
```

This last XML-file stores multiple quotes in a single version. It is possible to store all quotes in one version. But it might make sense to periodically create a new version, for example a version with the quotes for september 2000, a version with the quotes for octobre 2000, etc.

With this scenario, you need to use the following announcement files for the quote and history items.

```
<ITEM_ANNOUNCEMENT>
    <ITEM>
          <BARCODE>STOCK_HISTORY_GUID</BARCODE>
          <NAME>Stock history</NAME>
          <CONTENT_TYPE>xml</CONTENT_TYPE>
          <SCHEMA schema="STOCK_HISTORY_SCHEMA_GUID"/>
          <VERSIONING versioning="no"/>]
    </ITEM>
    <ITEM>

          <NAME>Stock quote</NAME>
          <CONTENT_TYPE>xml</CONTENT_TYPE>
          <SCHEMA schema="stock quote schema"/>
          <AGGREGATE
                target="STOCK_HISTORY_GUID"
                xslt="MERGE_MSFT_QUOTE_WITH_STOCK_HISTORY_XSLT_GUID"
          />
          <VERSIONING versioning="no"/>]
    </ITEM>
</ITEM_ANNOUNCEMENT>
```

TODO: Eigenlijk moet MSFT een parameter zijn voor de transformatie. Voor elke andere symbol zal de transformatie immers bijna gelijk zijn.

Note that the stock quote specifies that it is aggregated into the history item. The way the history is versioned, is specified in the announcement of the history item itself. In this example neither of the items is versioned, meaning we're only storing the latest quote and one complete history of all quotes.

File types and filename extensions

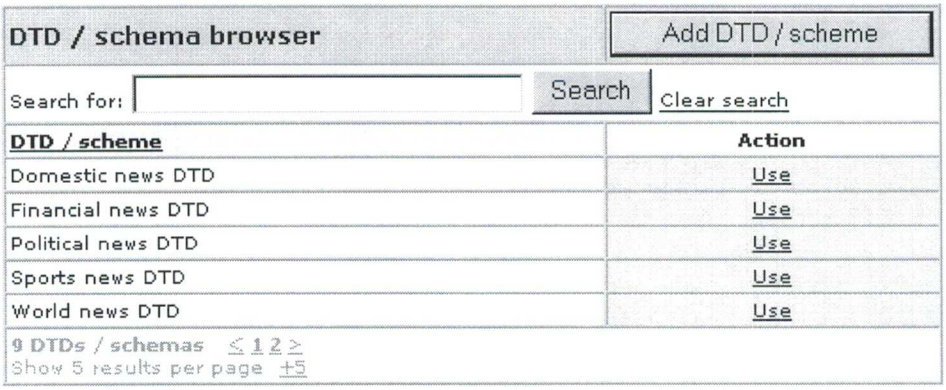
BSCL supports extension mapping, meaning that file extensions are automatically mapped to MIME types upon submission. This means that a file named "mlking.wav" will automatically be recognized as being of MIME type "audio/x-wav" when uploaded.

Files without an extension can be uploaded (this is usually the case on the Apple Macintosh). In this case the announcement for the file must contain the appropriate MIME type.

Most web browsers and other display devices require an extension in order to display e.g. graphics correctly. Upon publishing MIME types can be translated back to extensions if the target requires this.

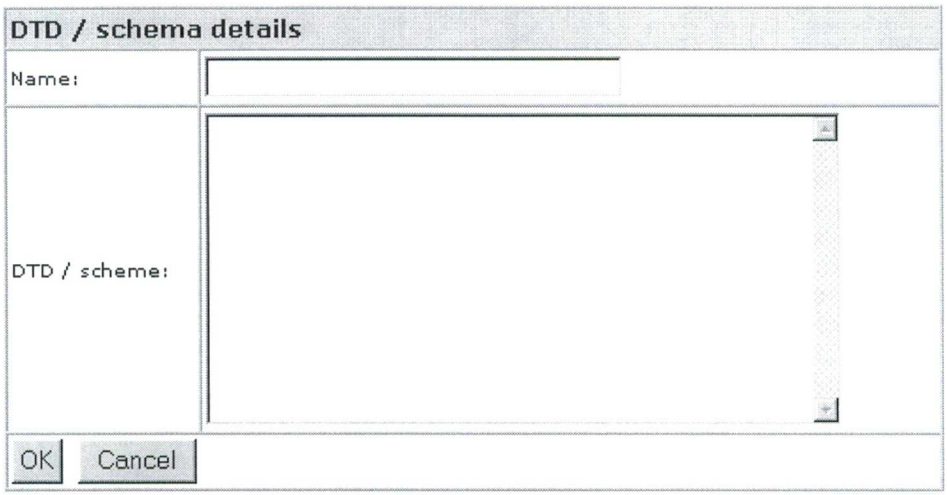
DTD/Schemas

All XML content in the BSCL system is based on (a version of) a DTD or schema. Those DTDs and schemas are managed using the following form:



Essentially this is just a normal content browser, with a filter limiting its contents to just DTDs and schema. The form shown above is the way it is displayed when you need to select a DTD or schema. When you're actively managing the DTDs (adding, editing and deleting them), more actions are available.

Editing a DTD or schema is done using the following form.



This is just a normal content version editing form in its most simple form. The contents are displayed in a single text area.

Standard schemas and transformations

BSCL itself uses schemas to define many different parts of the system. For example: there is a schema to define external content, one to define real-time feeds and there are schemas to define offers and subscriptions.

By defining these parts of the system as schema, it allows BSCL to treat them in much the same way as normal content. This implies for example that offers and subscriptions automatically have the ability to be versioned.

The following schemas are used by BSCL and therefore should be defined and always be present:

- Schema
- DTD

- XSL/XSLT
- External content
- Real-time feed
- Content item announcement
- Content version announcement
- Search
- Offer
- Subscription
- Form

The BSCL system also uses a number of default XSL-transformations internally. For example: the list of content items is first built in XML from the content item meta-data. The system then applies a transformation to this XML, to determine what is actually displayed to the user. By using an XSL-transformation for this purpose, both the structure and te look of the output can easily be modified.

The transformations used by the system include:

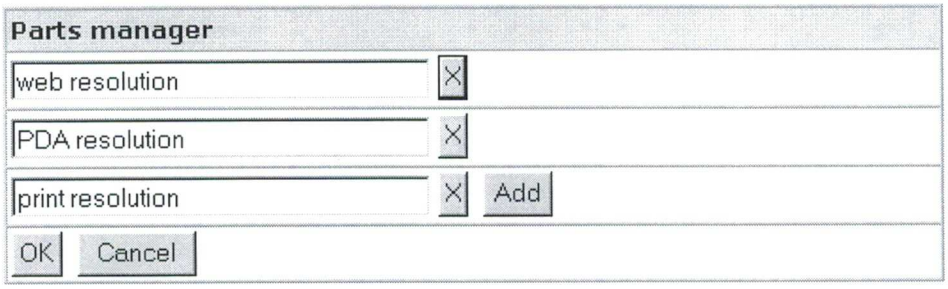
- Content item meta-data to content browser
- Content version meta-data to content browser
- Content item meta-data to content item manager tree
- Content version meta-data to content item manager tree
- Content item meta-data to content item manager properties page
- Content version meta-data to content item manager properties page

This list is by no means meant to be complete and should serve as an indication of the ways in which XSL-transformations are used by BSCL.

Content parts

A content item may be subdivided into one ore more content parts. Although the use of parts is generic, they are intended to be used for classifying content for output to different types of devices. The output transformations determine which version of a content item to use for their intended output device by looking at the parts in the content item.

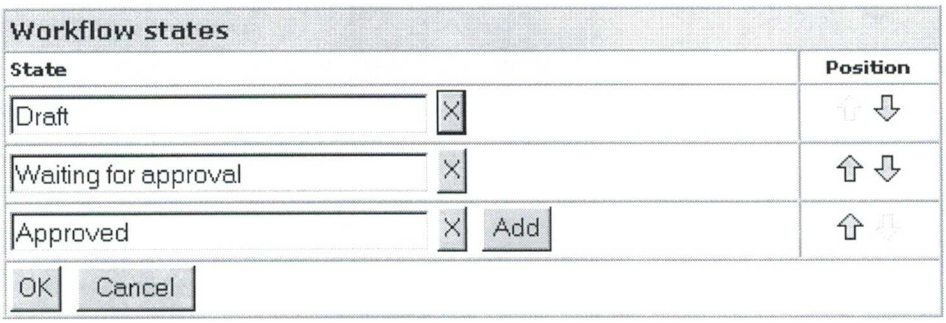
Parts are managed through the following form:



Since parts are not themselves not content items, this is a custom browser/editor-form.

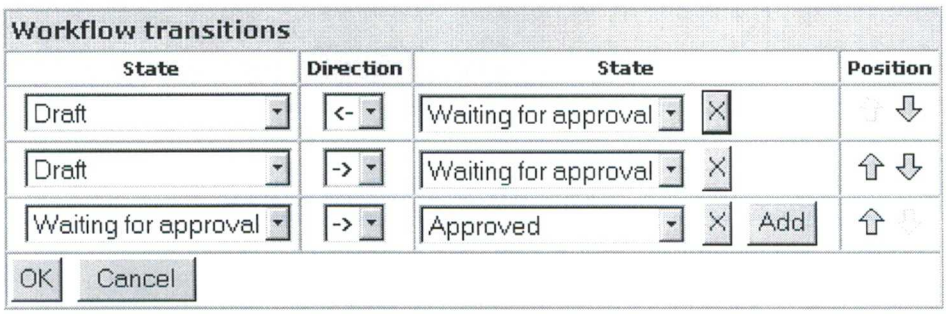
Workflow

The various states a content version can be in, are defined using the following form:



Note that this form allows the user to order the states. This ordering is used when displaying states and state transitions in the other forms.

After defining the states it is possible to define the allowed transitions between those states in the following form:



It is also possible to define which user of group of users has the right to perform a transition. This is described in more detail and illustrated in the section on user rights in the chapter describing the BackBone module.

Backbone

Content sink

Process flow

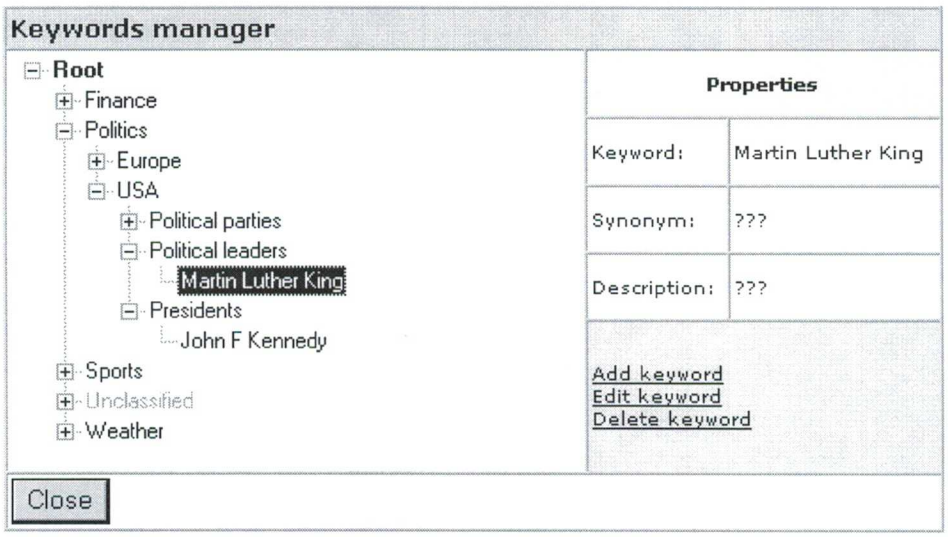
Drop directory

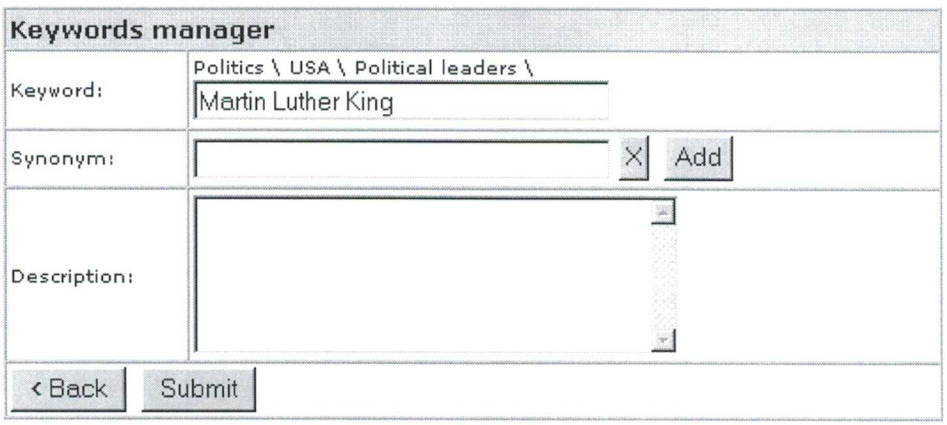
Announcements

Triggers

Keyword manager

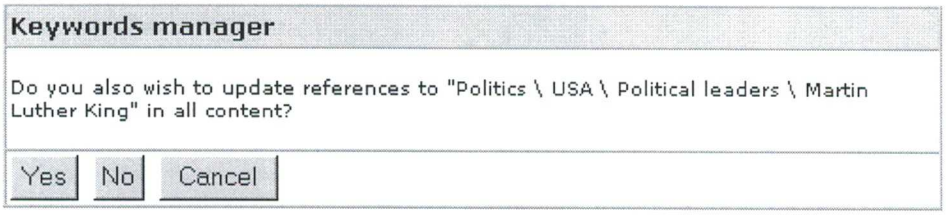
De keyword manager wordt door system administrator gebruikt om keywords toe te voegen en te bewerken.



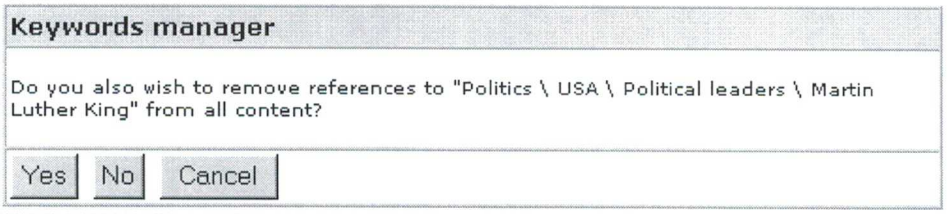


De synoniemen zijn een lijst. Klik op Add levert een extra synoniem op.

Als je een keyword wijzigt en je drukt op submit, krijg je dit:



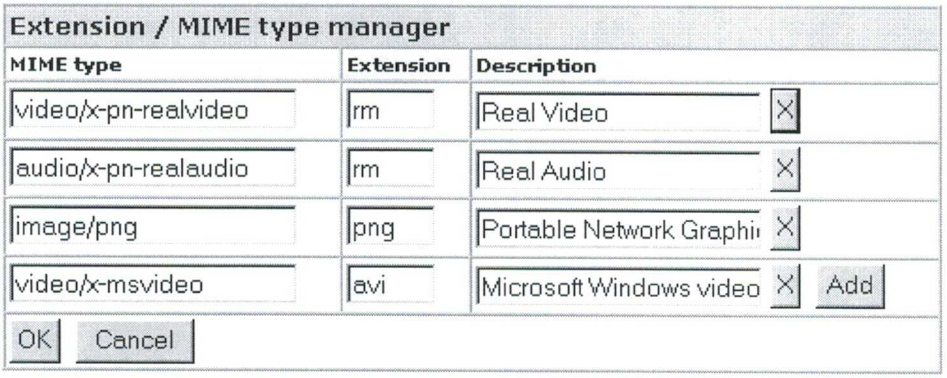
Als je op Delete keyword klikt, krijg je dit:



TODO: Deze moet direct na de keywords manager tree form

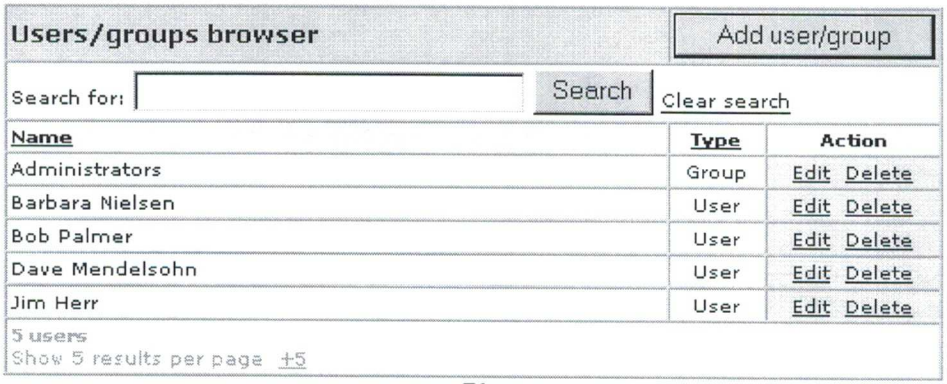
Aggregator

MIME types

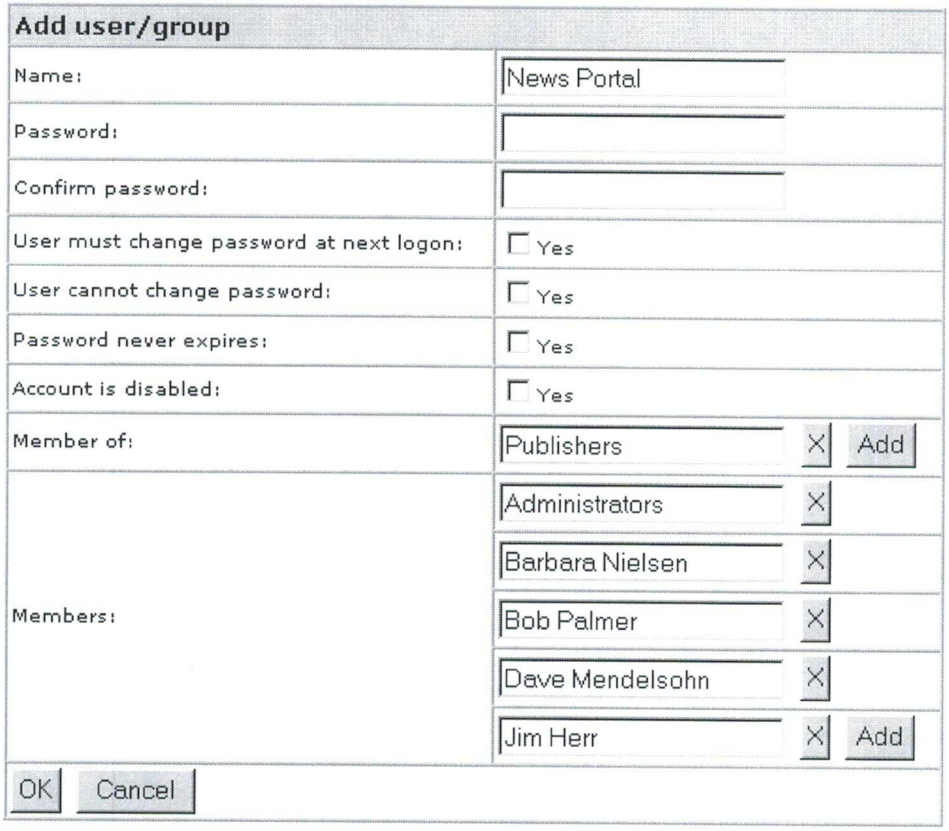


Dit is een custom browser en editor voor MIME types.

Users/Groups



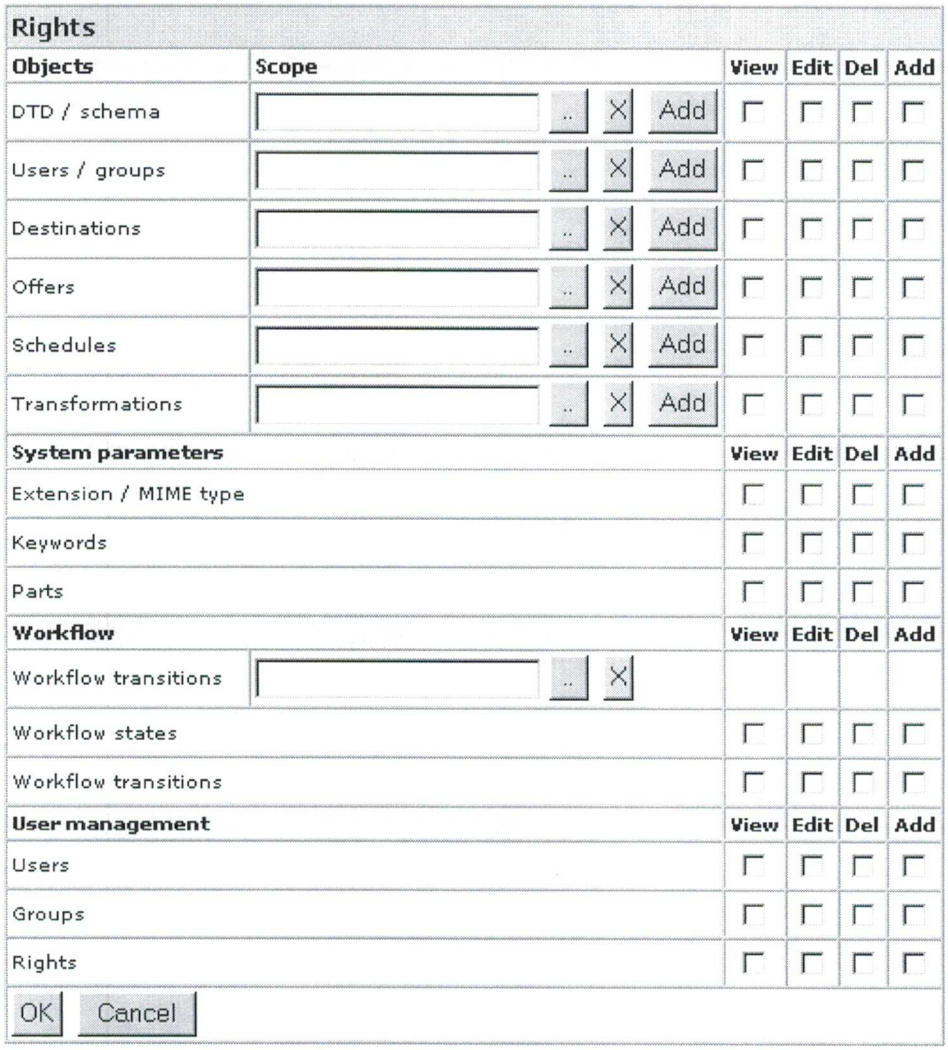
Dit ziet d'ruit als een content browser, maar is eigenlijk gewoon een custom browser. Het feit dat 't een groep is wordt afgeleidt uit het feit dat er users naar deze "user" verwijzen. Click op edit levert:



Je kunt ook voor een groep alle gegevens invullen. Hiermee zou je dus eventueel als groep kunnen inloggen. Ik weet 't nut d'r niet van, maar 't leek Erik wel leuk.

Rights

Rechten worden beheerd vauit het volgende form:

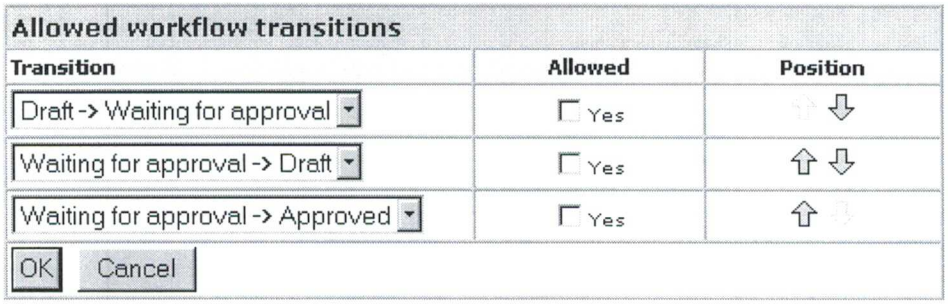


Dit is een custom browser/edit form, maar dat had je vast wel gezien. De onderkant toont wat specifieke systeemtabllen, de bovenkant wordt hieronder beschreven.

TODO: In een lopend verhaal opnemen.

The scope attribute is used to limit the view, edit, delete and add rights to specific content items and/or versions. The scope is defined by using the (advanced) search of the associated content browser. The resulting search result ("where clause") is stored and used to determine whether the user/group has access to a content item or version. Many of the content browsers use a "where clause" in order to show the correct content items/version. Technically speaking the users/groups rights system does exactly the same: it uses a "where clause" to filter content items/version.

Klik op de browse button voor workflow transitions, brengt je naar:



Hier geef je aan welke transition een gebruiker mag uitvoeren. Hiermee bepaal je dus de inhoud van de State-dropdown in het edit content version meta-data form.

Publicator

The publicator contains a process that takes care of sending all necessary content out to the various subscribers. It is also possible for the subscriber to pull the content from the BSCL system.

TODO: Als een content item al klaarstaat voor verzending (dus in tblOfferContents), wat doen we dan als er toch nog iets wijzigt? Concreet: als de versie van APPROVED afgehaald wordt omdat'ie toch nog niet klaar was.

TODO: Een digest zou je kunnen opvragen door de BSCL:// (of de bar-code) van de subscription op te geven. Dat is vooral handig, omdat digests niet worden opgeslagen.

TODO: waterval. Zie ook DB-model.

1. Als een offer geactiveerd wordt (door de scheduler), dan kijkt deze welke content er binnen het offer valt. Deze content items/versions worden in tblOfferContents geschreven.
2. Als een subscription geactiveerd wordt (alweer door de scheduler), kijkt deze welke content uit tblOfferContents binnen de subscription valt. Deze content items/versions worden in tblSendQueue gezet.
   NB: Wat als tijdens het uitvoeren van deze task, ook de offer net wordt ververst? Even uitstellen.
3. De publishing server pakt steeds een item uit de send queue en probeert deze te versturen naar de destination (uit de subscription). Lukt dat, dan wordt 't record verwijderd.
   NB: De content wordt natuurlijk eerst flink getransformeerd (op basis van de subscription en de offer), voordat deze verstuurd wordt.
4. Er is een housekeeping-thread die kijkt of er in de send queue records staan die over de houdbaarheidsdatum heen zijn. Zo ja, dan worden deze verwijderd.

Destinations

Vanuit dit scherm beheer je de destinations waarnaar content wordt verstuurd.

Destination details	
Name:	
Protocol:	FTP
Host name or IP address:	
SMTP settings	
E-mail address:	
FTP settings:	
User name (leave empty for anonymous FTP):	
Password (leave empty for anonymous FTP):	
Anonymous FTP?	Yes
Transfer type:	Passive
OK Cancel	

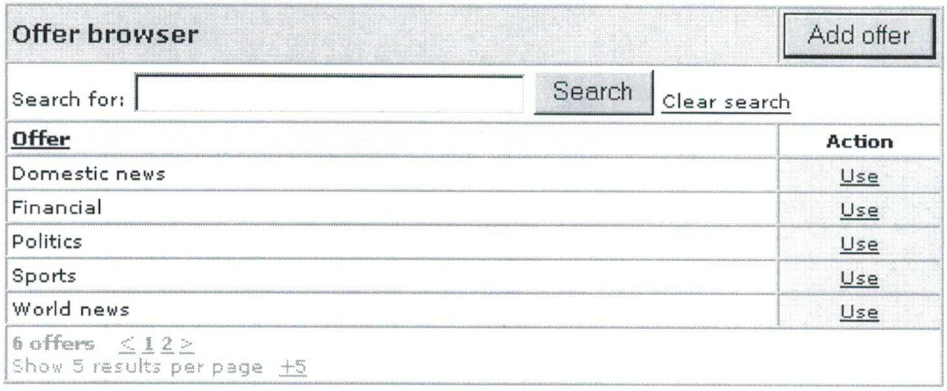
Destination browser		Add destination	
Search for:	Search	Clear search	
Name	**Host name**	**Protocol**	**Action**
Jupiter-01	apollo.backstream.com	FTP	Use
Financial	jupiter01.client.backstream.com	HTTP	Use
Politics	jupiter02.client.backstream.com	HTTPS	Use
Sports	portal.backstream.com	SMTP	Use
World news	192.32.234.1	SSH	Use
8 destinations < 1 2 > Show 5 results per page +5			

TODO: Plak browser voor de editor.

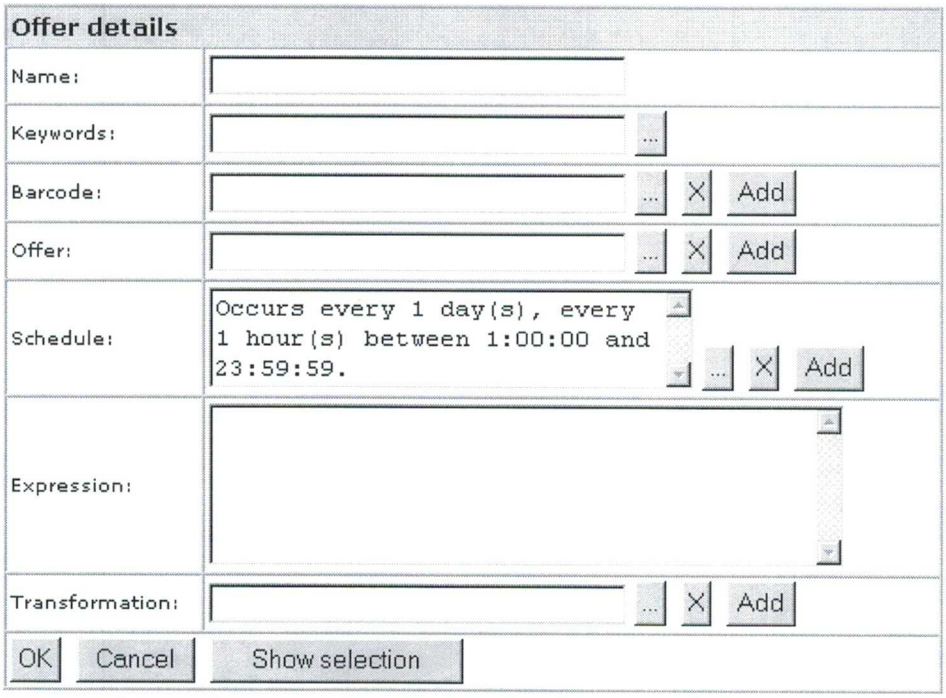
Dit is een content browser met een filter op de BSCL-Destination DTD.

Offers

Offers worden gemanaged vanuit het volgende scherm:



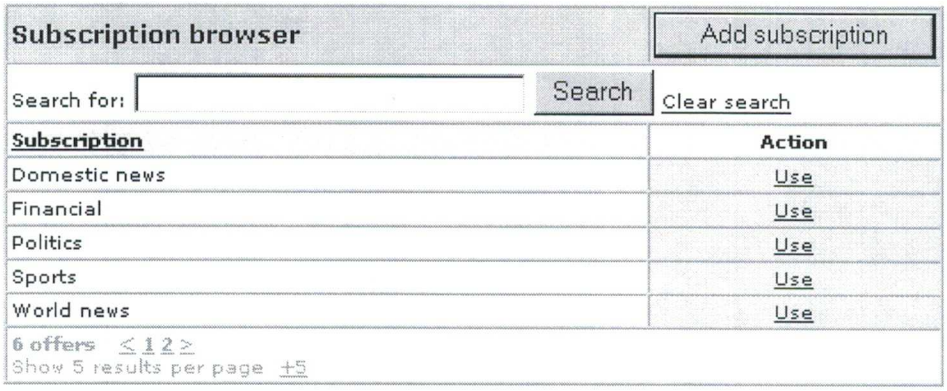
Dit is een content browser met een filter op BSCL-Offer DTDs.



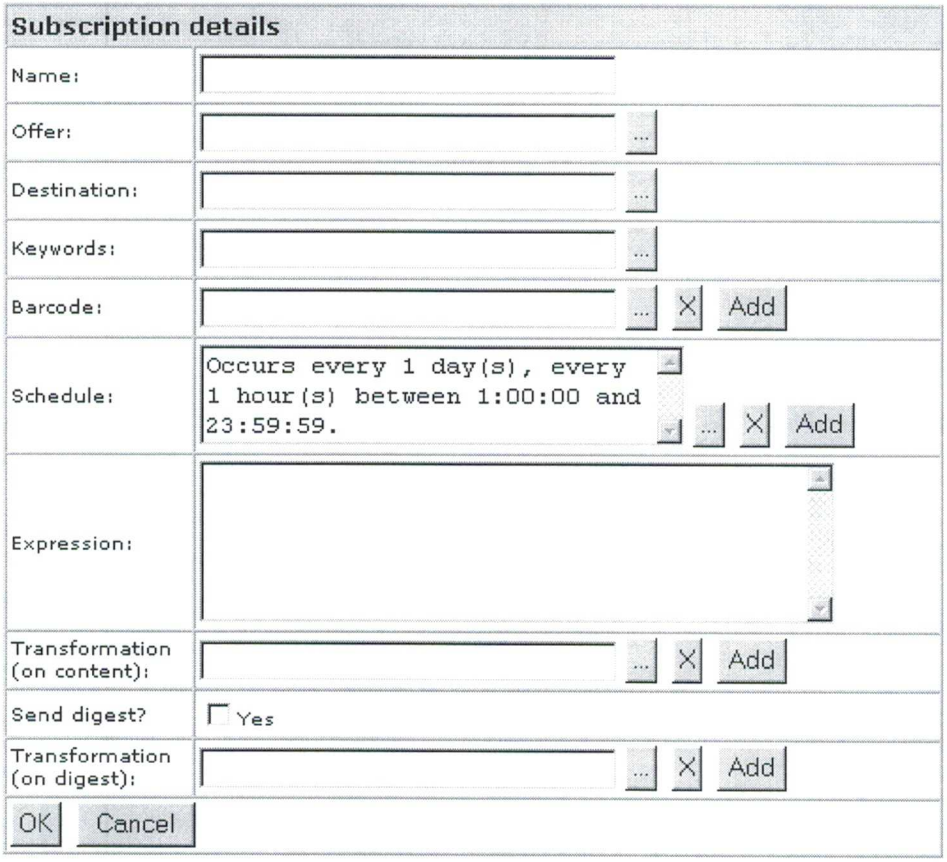
Een offer kan meerdere BSCL://, offers en schedules bevatten. De offers worden niet echt "bevat", er is dus geen sprake van een master detail relatie.

Subscriptions

Subscriptions worden beheerd vanuit het volgende scherm. Iedere subscirber ziet hierin natuurlijk alleen zijn eigen subscriptions.



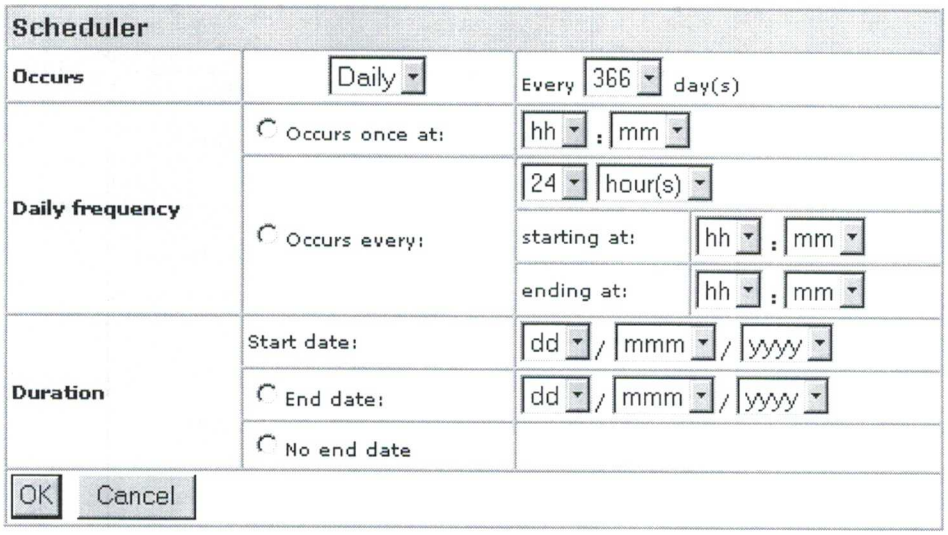
Een content browser met een filter op BSCL-Subscription DTDs.



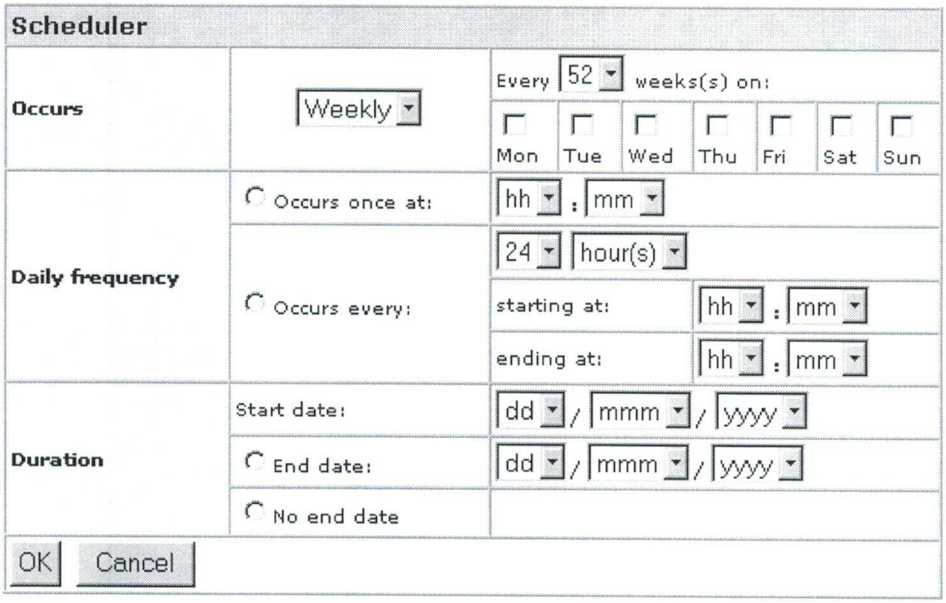
Je kunt dus zowel de content versions als de digest version krijgen. Voor beiden geldt een andere transformatie.

Schedules

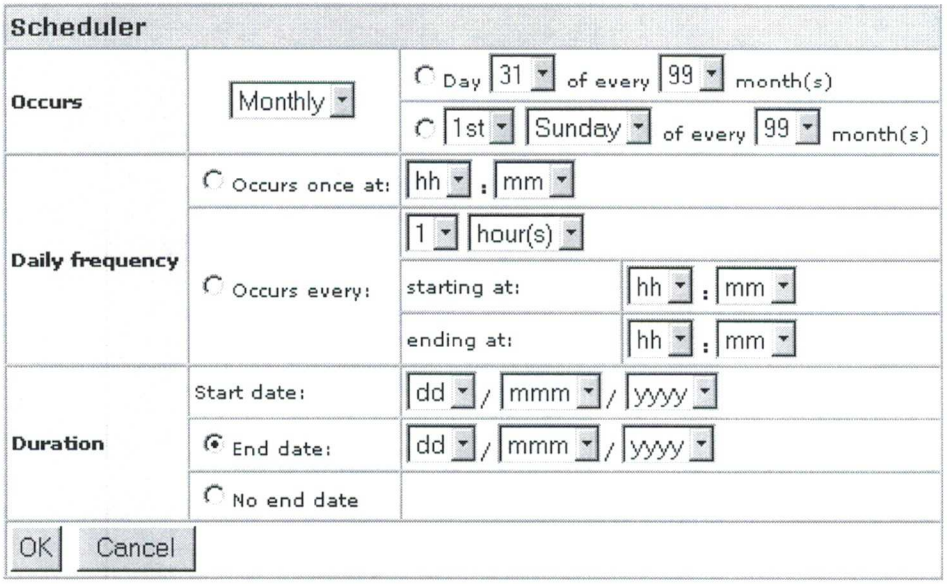
Je kunt schedules niet browsen, omdat je ze ook niet echt los opslaat. Het zijn wel gewoon content items met een BSCL-Schedule DTD.



Dit is een daily schedule (bovenste frame). Je kiest vervolgens de intra-day frequency (middelste frame) en hoe lang de schedule geldt (onderste frame).



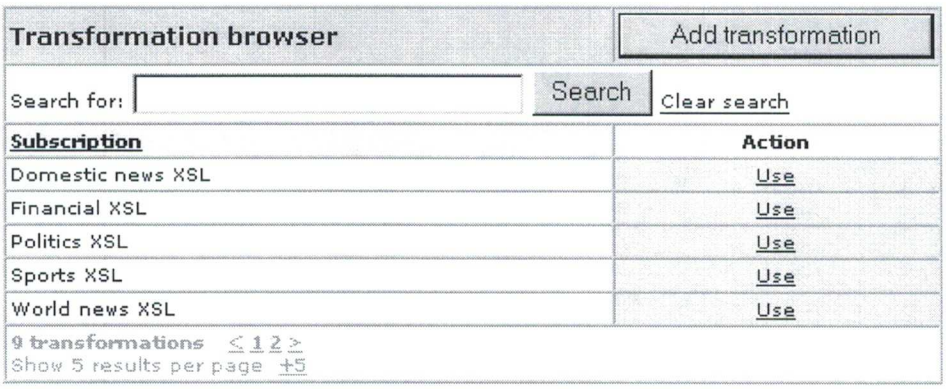
Die is een weekly schedule. 't Enige verschil met de daily schedule zit in het bovenste frame.



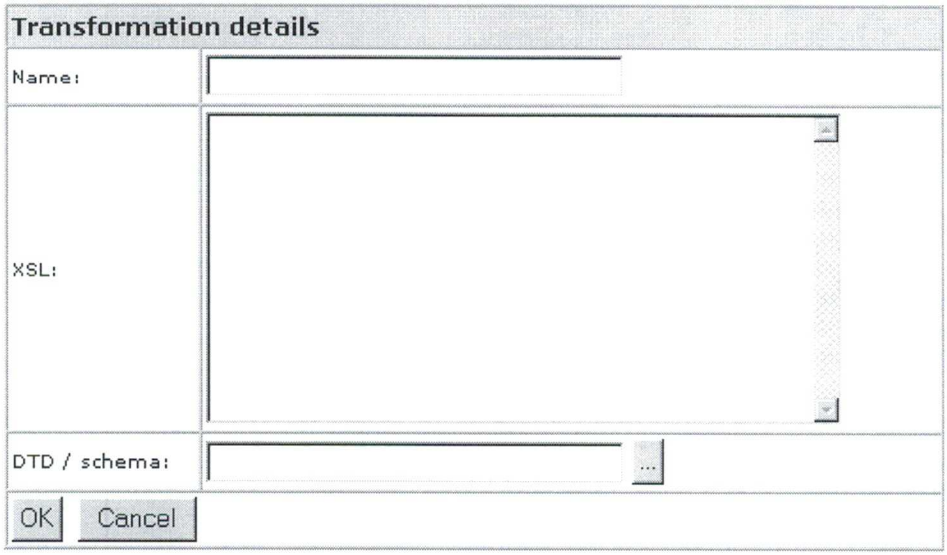
Dit is een monthly schedule. 't Verschil zit weer in 't bovenste frame.

Transformations

Transformations worden beheerd vanuit het volgende scherm:



Dit is een content browser met een filter op de XSLT-DTD.



Je geeft hier aan bij welke DTD/schema deze transformatie is toegestaan. Dit wordt op andere plaatsen gebruikt om de lijst in de transformation browser (of dropdown) te beperken.

Tracking & tracing

Je moet de volgende objecten kunnen tracen:

- Content items
  Dit omvat schedules, XSLs, DTDs en schema's, offers en subscriptions, destinations,
- Content versions
- Users
- Workflow-onderdelen

De syndicator houdt bij

- wanneer een item of version is binnengekomen, bijgewerkt, verwijderd
- wanneer een user wordt toegevoegd, gewijzigd, verwijderd, wanneer hij inlogt
- wanneer keywords worden toevoegd, gewijzigd, verwijderd
- workflow
- content parts
- extensions

De publicator houdt bij

- wanneer een offer of subscription is aangemaakt, gewijzigd, verwijderd, uitgevoerd.
- wanneer een item of version (of een digest) is verzonden of opgevraagd, inclusief failures
- wanneer een subscriber is aangemaakt, gewijzigd, verwijderd
- wanneer een verzonden version, wordt bekeken. Hiertoe wordt dus in de version een link naar een BSCL/ opgenomen (met BSCL:// en subscription GUID).

Billing

Semantic definitions/glossary

- **BSCL**
  The BackStream Content Logistics system.
- **BSCL://**
  A BackStream Content Locator. This is a notation that allows us to identify any kind of content in a BSCL system. This can include the barcode of a content item, a part name, a version number, the name (or IP address) of a BSCL system.
- **Content**
  Any type of data that can be stored in BSCL. All content is qualified as either binary or textual and as being either local or remote.
- **Binary content**
  All content that doesn't qualify as being textual content. This includes images and sound or video files. Binary content cannot be edited using BSCL without using external programs or plug-ins.
- **Textual content**
  Content that uses a textual presentation. This type of content is further subdivided into XML content (also known as structured content) and unstructured content
- **Structured content**
  See: XML-content
- **Unstructured content**
  Textual content, which doesn't qualify as valid XML. This can either be because the content is not well-formed, or because it doesn't have a DTD or schema associated with it or it doesn't meet the requirements of it's DTD or schema.
- **XML content**
  All content adhering to a specified DTD. This includes news articles, headlines or any other XML file based on a valid DTD or schema. XML content can be edited using browser-based forms.
- **Remote content**
  Content that is not stored on the current BSCL system. This content is described on the local system by a content reference.
- **Local content**
  Content that is local to the current BSCL system.
- **Content reference**
  A reference to local or remote content that is stored in the current BSCL system. Since a reference is specified as a special kind of XML-file it can be stored in the local system and used to gain access to the remote content. Alternatively you can use references to provide multiple barcodes to a piece of local content.
- **Barcode**
  The Id of a content item in a BSCL system.
- **DTD**
  A document type definition, specifying the structure of a specific set of XML-files.
  See also: Schema

- **Schema**
- **Publisher**
  Publishers maintain, edit, enrich, index, tag and categorize content. They also define the offers i.e. the packages in which the content is presented to the subscribers.
- **Subscriber**
  Subscribers receive content from the BSCL by indicating which offers they wish to receive. The content is either pushed to the subscriber or pulled from the BSCL system by the subscriber.
- **Subscription**
  A subscription defines an offer to which a subscriber is entitled, which publication schedules are used and which data transformations should be applied to the content.
- **User**
  A BSCL user is a content contributor, publisher, subscriber, administrator or any combination of these roles.

- **Announcement**
  An XML document that signals to BSCL that a new content item or new content version must be created. Usually this document takes the form of an XML file in de BSCL drop folder.

- **Content item manager**
  The content item manager is used to view and edit a content item. It allows the user to access the content item meta-data, the different parts of the content item and all of the versions.

- **Drop folder**
  A directory on the BSCL system, which is continuously scanned for new files to be processed and added to the system. Files sent to the BSCL system by FTP or e-mail are placed in this folder. Content added using a browser-based form, may also be written to a file that is added to this folder.

- **Content file**
  The file containing the actual content of a version. A content file is typically uploaded to the BSCL system and put into the drop folder to await further processing.

-

Appendix: Case studies

Case study: sub-content in content item manager

Content Item Manager

- I have a dream
  - Part "web version"
    - 1: Draft
    - 2: To be approved (typos corrected)
    - 3: Approved
  - Part "pda version"
    - 4: Draft (abbreviated)
- Sub content
  - Martin Luther King biography
    - 1: Draft

Version properties	
Author:	Barbara Nielsen
Keywords:	Martin Luther King
MIME type:	N/A
Part:	N/A
Version:	1
Reason:	
State:	Draft

Edit properties
Revise version
Delete version
View version

Upload version

Case study: preview in content item manager

Content Item Manager

- Mobile Phone
  - 1: Approved

Version properties	
Author:	Barbara Nielsen
Keywords:	Mobile phone
MIME type:	image/jpg
Part:	N/A
Version:	1
Reason:	
State:	Approved

Edit properties
Delete version
View version

Upload version

BackStream

White Paper

Content Processing Platform

Content is King if it is billable

Building blocks

BackStream CPP is a platform with a rich set of building blocks delivering advanced content processing and business process integration functionality. The building blocks offer functionality to set up, maintain and manage the input, storage, syndication, aggregation, categorization, indexing, versioning, archiving and publishing of content. To assist the content supply chain BackStream CPP offers extensive (customizable) workflow functionality and strong role-based user management. Scalability is realized by the inherent capability of the building blocks (multi-threaded services) to be distributed among multiple severs. This architecture allows BackStream CPP to process massive amounts of content in real-time.

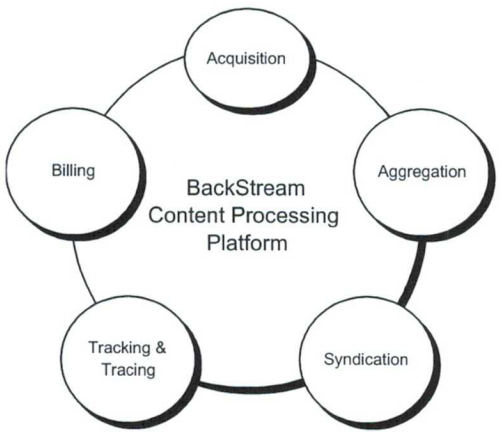


Fig. 1 – BackStream Content Processing Platform

Tracking and tracing

BackStream CPP enables organizations to track and trace content during the life cycle. It does so by assigning a "barcode" to each content item which uniquely identifies a piece of content, whether it is text, a photo or a video stream. BackStream CPP provides the means to track and trace an item not only within the BackStream CPP platform but also outside of it (e.g. after it has been published to a web site). The extensive amount of information BackStream CPP gathers about the usage of a content item enables it to value content on implicit quality (by evaluating criteria like number of times read, creation date, attention span duration, etcetera). In this way BackStream CPP can support various content-for-fee business models like pay-per-view, bulk, revenue sharing, click-thru, etcetera.

CPP takes XML to the max

Content types

BackStream CPP works with any kind of content, whether it is structured text (e.g. XML) or unstructured data (like a word processor document). It handles text, pictures, sound and video in any imaginable format. Content does not need to be stored locally (inside the BackStream CPP data store) but can also be located elsewhere on the Internet or within an enterprise. The BackStream CPP data store can be hooked up to legacy databases, external data sources or even file systems.

Dynamic channels

When structured content enters the system, BackStream CPP transforms it to XML. During the (multi-channel) publishing stage XML content can be transformed to e.g. another XML dialect, to HTML, to WML or to PDF. Unstructured data (like Microsoft Word documents) can be converted to XML using third-party plug-ins.

OSI-layer model positioning

When looking at the OSI-layer model, BackStream CPP is positioned in the 5th (protocol), 6th (presentation) and 7th (application) layer:

7	BackStream CPP tracking, tracing and billing services	Application
6	BackStream CPP presentation layer XML / XSL(T)	Presentation
5	GUI IPC Tracking & Tracing	Protocol
4	HTTP:// Sockets TCP/IP	Transport
3	Exodus, Level3, PSINet, UUNet	Network
2	3COM, Cisco, Nortel	Data link
1	Ethernet	Physical layer

The presentation layer is entirely driven by XML and XSL(T) transformations. This creates a quickly customizable and device-independent user interface that runs on any device equipped with a web browser. The presentation layer is built on top of the protocol layer which handles the user interface over HTTP and the inter-process communication between the BackStream CPP building blocks via sockets. The protocol layer also handles the tracking and tracing of content items. The XML-based inter-process communication protocol enables developers to interface with the building blocks. Supported interfacing protocols are ICE, SOAP, RMI and JAXM. An extensive Java API and COM interface is also available.

CPP uses and supports the following interfacing protocols:

- ICE
- SOAP
- RMI
- JAXM

To keep an eye on the flow of content at the hardware level (making it impossible to circumvent tracing) BackStream CPP is creating interfaces with router hardware (positioned in the 2nd (data link) layer) and Content Delivery Networks (CDNs) from vendors like Cisco and Nortel. This creates an unprecedented accuracy in the way content is tracked and traced.

CPP leverages content processing applications by providing a powerful platform

Application solutions

- **Content providers (originators (suppliers) and publishers of new content):** content providers (e.g. news agencies) can use the full BackStream CPP suite to author, store and publish content.
- **Content management tool vendors:** CMS vendors can license building blocks to build (parts of) their CMS solution.
- **Content aggregators and syndicators (collectors and (re)distributors of existing content):** syndicators often rely on custom-built software to acquire, author, store and publish content. BackStream CPP allows these companies to replace some or all of these applications by BackStream CPP building blocks, freeing the syndicator from costly application maintenance.
- **Software companies:** "traditional" software companies can employ BackStream CPP technology to enhance their networked applications with content delivery functionality.

Functionality and characteristics

- BackStream CPP can be used to offer and package content (text and media assets) for syndication to partners. Syndicated content can be shipped to multiple destinations and platforms.
- Content enters BackStream CPP using data feeds via HTTP, FTP, SMTP or file systems. Manual input –supported by a flexible role-based workflow system– takes place in a powerful browser-based environment.
- BackStream CPP can manage, store and publish content with links (embedded references) to media servers (e.g. streaming video), legacy databases or even file systems. BackStream CPP can also integrate with the top ERP and CRM offerings.
- Content offers are dynamic, based on criteria like keywords, meta data and expressions. Publishers (who create offers) and subscribers (who subscribe to offers) can both fine-tune the content offers and delivery schedules. The intuitive, browser-based user interface makes it easy to give access to the content repository, to create delivery packages and schedules, to set up data connections to servers or display devices and to create transformation schemas.
- Offers can be embedded within offers and extensive publication schedules can be associated to offers. Publishers and subscribers can apply different schedules to the offers and subscriptions they create.

The CPP components provide the high cohesion and loose coupling developers are seeking

- BackStream CPP supports push (by provider or publisher) and pull (by subscriber or affiliate) exchange methods over HTTP, SMTP, FTP and local or wide area networks. New protocols such as ICQ, Exchange, SMS or client-specific protocols can be added quickly because of the open plug-in architecture of all the BackStream CPP components.

- The pull exchange method provides an interactive browsing environment that makes it easy for subscribers to search, locate, acquire and use content. The available content is constrained by the offers created by the publisher.

- It can mold content into a different look-and-feel (branding) for every destination by applying presentation template transformations to content. This approach also enables BackStream CPP to ship content to multiple devices (e.g. wireless internet and iTV devices)

- From the ground up, BackStream CPP has been designed to be deployed as an ASP (Application Service Provider) service. Inherent in its design are capabilities to separate user domains and content domains.

- The tracking and tracing capabilities of BackStream CPP support a wide range of content-for-fee business models like pay-per-view, bulk, revenue sharing, click-thru, etcetera.

- BackStream CPP does not impose a single DTD or schema. Instead it allows any content item stored in the central content exchange to be based on its own schema. The browser-based forms, used to input content, are also based on schemas.

- During the content acquisition and normalization process structured content is parsed and scanned for meta data information like keywords.

Component benefits

There are several compelling reasons to use BackStream CPP as the basis for professional content processing applications:

- Faster to market with pre-built, pre-tested logic
- Cost is a fraction of the development time
- BackStream CPP contains extensive content management domain knowledge

[...a very rich platform for XML integration...]

Architecture

From a technical perspective, BackStream CPP can be seen a heterogeneous collection of (Java) services (server-side threaded processes), interlinked by a socket-based inter-application messaging protocol. The design allows the components to be distributed among multiple servers. This architecture enables the creation of real-time, high-volume content processing applications.

Areas of XML usage

According to a Giga Information Group study, the usage of XML is diversified, as shown by the following chart:

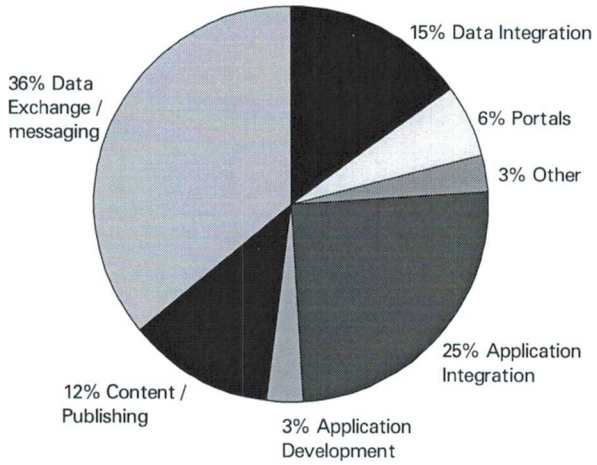
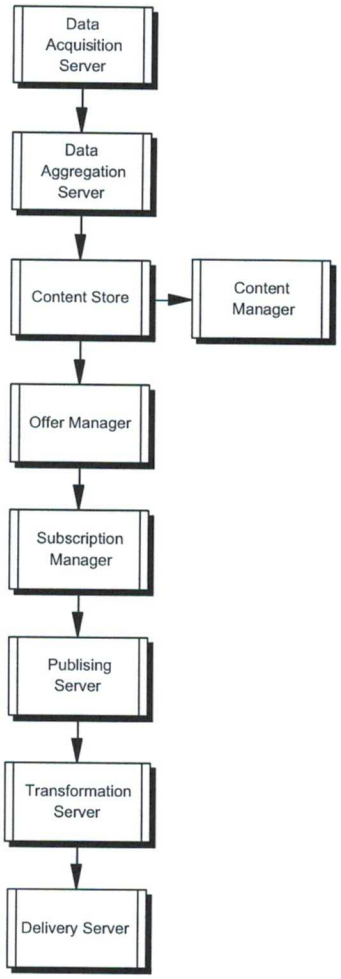


Fig. 2 – Areas of XML usage

Since BackStream CPP can be used in all areas it is regarded as a very rich platform for XML integration.



BackStream CPP architecture comprises the following objects:

Servers	
Data Acquisition Server	Handles automatic data acquisition via HTTP, FTP, SMTP and (networked) file systems. During acquisition content can be tagged automatically with (globally maintained) keywords.
Data Aggregation Server	Transforms XML, EDI docs and other structured text to XML based on user-definable schemas (DTDs).
Content Store	Content and meta data can be distributed among different servers from different vendors. The Content Store supports SQL Server, Oracle, Sybase, MySQL, DB2 and others.
Publishing Server	Manages the automatic scheduling of content delivery. Content is either published automatically based on pre-defined rules or can be retrieved on demand.
Transformation Server	Performs automatic transformation during acquisition and publishing of XML content to various input and output formats (XML, WML, HTML, EDI, ASCII, etcetera). Content can be delivered as separate files or as a single file digest ("headlines").
Delivery Server	Selects and delivers the most suitable version of content based on the destination's characteristics and the subscriber's preferences. This allows e.g. the delivery of low-bandwidth images to portable devices and high(er) bandwidth images to the web.
Tracking Server*	Tracks and traces content items during their journey through CPP (even externally stored content). It can process usage and click-thru logs delivered by web servers and digital rights management (DRM) systems.
Report Server*	Delivers extensive browser-based reporting and analysis on content usage. Provides data export functionality for further processing.
Data Integration Server*	Enables the connection between enterprise legacy databases and file systems and the Content Store.
Billing Server*	Provides interfaces with billing software (iPAY, QPass, invoicing software) to enable the billing of content on aspects like input, storage, delivery and usage.
Managers	
Content Manager	Provides a web-based forms environment based on schemas with automatic versioning, role-based workflow functionality and digital content management for binary content (images, sounds, video). Content can be maintained, revised, enriched, tagged and categorized.
Offer Manager	Publishers can define dynamic content offers based on keywords, search criteria, version and expressions.
Subscription Manager	Subscribers can subscribe to offers created by publishers. Subscribers can fine-tune the offer contents to meet exact requirements and manage destination settings like host, protocol, etcetera.
Platform Manager*	The Platform Manager is the CPP application management tool which gives access to e.g. user management, workflow settings, schema definitions, etc.
Cluster Manager*	Allows the setup and administration of clustered CPP systems to form one homogenous content repository.
* not listed in diagram	

Dependencies

Requirement	Vendor/version
Java virtual machine (JVM)	Version 1.3 (available for all major operating systems)
Servlet engine	Version 2.1 (available for all major web and application servers)
Database engine	Sun JDBC (available for all major operating systems)