Planned Features for uPortal 2.1 and Beyond

At the recent uPortal developers meeting[1], the attendees brainstormed about new features that they would like to see in future versions of uPortal. The ideas from the attendees and others[2] in the uPortal community were compiled into a list and prioritized with respect to what features should go into uPortal 2.1[3] and what features should be left for later releases assuming that a new version of uPortal is released approximately every 6 months.

The features planned for uPortal 2.1 include:

- **Aggregated layouts** – Allows for administration of layout fragments (e.g. tabs, columns) by different entities. This would make it possible for the university to control a tab in every user's layout or for a professor to control a tab for every student in his/her class. It is similar to what Campus Pipeline has developed and termed *Distributed Layouts*. This will require some code refactoring to allow for easier management of user layout node permissions.
- **CAR files** – These **C**hannel **AR**chive files would be used to easily install or deploy a channel and all its resources into a uPortal installation. Packaging standards and GUI tools to deploy these files will need to be developed. Modifications to exiting uPortal tools such as DbLoader may be necessary.
- **Groups/Permissions enhancements** – More sophisticated caching is desired. Database locking issues must be resolved. Permissions policies must be made pluggable. Groups and permissions management channels must be incorporated into the uPortal framework.
- **Publishing/channel administration enhancements** – Groups and permissions UI's as exposed by the Groups and Permissions channels must be incorporated. A channel preview screen is desired at the end of the publishing process. A mechanism for publishing channels that require an arbitrary amount of parameters is necessary. This would enable the publishing of applet and embedded object channels.
- **Frame support and detaching channels** – Some rework to URLs used in uPortal is necessary to resolve current problems with supporting frames and detaching channels.
- **Browser differentiation** – uPortal utilities need to be able to distinguish between browsers of different versions.
- **Channel error handling –** Channels need the ability to automatically restart if they are having problems rendering. Users need the ability to submit reports to an administrator describing errors that they encounter.
- **Layout management –** Tighter integration of layout display and management is desired. For example, the widgets for moving tabs, columns, and channels should

[1] March 11-13, 2002 at the University of New Mexico

[2] Those not present at the meeting were given an opportunity to submit ideas for new features via email

[3] uPortal 2.1 is scheduled to be released in October 2002.

be available on the same screen as the rendering of the tabs, columns, and channels themselves. Users would have to go to the preferences channel less often. It is also necessary to allow for persisting small changes to the data store in a more efficient manner. Currently, the whole entire layout structure is persisted even if the user has made a minor change.

- **Tree/column structure and theme** – An alternative to the tab/column structure that would allow for unlimited levels of channel groupings. This effort was started before uPortal 2.0 and now needs to be completed.
- **Web services extension –** uPortal will expose channels through a web services layer. This allows for web services clients (including a generic uPortal channel) to interact with channels on behalf of their users using SOAP. Interactive Business Solutions (IBS) and Learning Assistant Technologies (LAT) are partnering to develop this functionality.
- **Storage improvements** – Channels and framework components need the ability to request connections to multiple relational databases and/or LDAP servers. A convention to look up and obtain these connections via JNDI must be established. Also, relational database storage implementations currently lack the ability to be configured to interact with different relational databases.
- **Connection pooling** – An infrastructure for swapping in and out connection pooling implementations is necessary, especially since the currently recommended 3rd party tool Poolman[4] has proven to be problematic and is now unsupported.
- **WML theme –** An effort to support WML browsers is partially implemented and needs to be completed. The usability needs improvement and more channels included with the uPortal framework need to support WML.
- **Internationalization** – uPortal needs an example of portal content delivery in a language other than English. A mechanism to let users choose their language is desired.
- **Web proxy enhancements –** The web proxy channel needs to implement caching and multithreaded behavior as well as support for authentication to external web sites.
- **Usage statistics –** The framework must allow for the insertion of statistics-gathering code. It is desired to measure frequency of channel renderings, channel accesses, user logins, etc. These statistics could be fed to flat files, databases, or even reporting software.
- **System-wide channels –** There is a need to allow access to channels that aren't necessarily part of a user's layout. Channels like user preferences and channel admin are currently included in every user's layout as hidden channels. This shouldn't be necessary. System-wide channels are also needed to implement channel previewing during publishing and subscription.
- **Generic subscription channel** – It is necessary to create a reusable UI component for subscribing to channels. As more themes are created, this component will become more and more desirable. The implementation of this component would be similar to the groups channel being used as a "servant".

[4] See http://www.codestudio.com

Features that are planned for uPortal 2.2 and beyond include:

- **Integration of content management systems**
- **Integration of workflow services**
- **Custom channel controls –** Establishing a mechanism for channels to manage the controls buttons (currently help, info, edit, etc) that they use to trigger events
- **Custom class loaders –** Useful for ensuring a “sandbox” security model for channels
- **Composite group service –** Talks to different group stores, e.g. LDAP.
- **Back button support –** Allow users to use the browser back button to navigate back through previous portal screens
- **Wrappers for other portal components –** This would allow code from other portals (portlets, pagelets, modules, etc) to be installed as uPortal channels.
- **Layout fragment subscription –** This is the ability for users to subscribe to tabs or columns (assuming tab/column structure) that are pre-configured with channels.
- **Auto-profiles –** Use of centralized repository of browser information to determine user profiles
- **HTTP caching utility –** Build a portal utility that channels can use to implement HTTP-level caching.
- **Channel development kit –** Build a toolset for rapid development of uPortal channels
- **User admin tool –** Offers ability to add and remove users
- **Runtime admin tool –** Gives information about memory usage, active sessions, and manages caches
- **Multi-JVM synchronization –** Investigate issues associated with synchronization across multiple JVMs
- **Channel keywords –** Associate channel definition with channel keywords so that channel can be located via a keyword search
- **Hierarchical installations –** Distributed aggregated layout fragments

If time permits and sufficient resources are available, some of the features scheduled for uPortal 2.2 and beyond could be moved into uPortal 2.1. Developers can influence the priority of implementing these features by simply volunteering to do the work.