Bedework 3.9

Bedework 3.9 Features List

- Written in Java
- Standards-based and Interoperable
- Includes a CalDav server
- Includes Web clients
- Database independence
- Calendars may be shared
- Meeting scheduling
- Event import and export
- Multiple calendars for users and for public events
- Tagging and filtering
- Internationalization
- Customizable calendar feeds
- Portal support
- Timezone support
- Recurring events with overrides
- Container authentication
- Support for other clients and calendar servers

Questions

Questions or comments about Bedework should be posted on the bedework-user email list.

Documentation To-Do List

Introducing Bedework

Bedework is an open-source enterprise calendar system that supports public, personal, and group calendaring. It is designed to conform to current calendaring standards with a goal of attaining strong interoperability between other calendaring systems and clients.   Bedework is built with an emphasis on higher education, though it is used by many commercial enterprises.

You may choose to deploy Bedework for public events calendaring, personal calendaring and scheduling, or both. Bedework is suitable for embedding in other applications or in portals and has been deployed across a wide range of environments.

Features of Bedework

- **Java :** Written completely in Java, Bedework is system independent. Currently it will compile and run in Java 1.6.

- **Standards based and interoperable :**Interoperability with other calendar systems and clients by way of standards compliance is a fundamental design goal of the Bedework system. The following standards are supported:
    - iCalendar support (rfc2445)
    - iTIP (rfc2446)
    - CalDAV (rfc4791)CalDAV scheduling (draft)
    - CardDAV v.4
    - VVenue (draft)

- **CalDAV server :** a full CalDAV server is a core feature of Bedework. It can be used with any CalDAV capable client and has been shown to work with Mozilla Lightning, Apple's iCal, Evolution, and others.

- **CardDAV server :** Bedework provides a CardDAV server providing personal and public contact stores for use in the personal client. A stand-alone JavaScript address book application is included with the personal web client suitable for deployment in other web applications such as email web clients.

- **Web clients :**The Bedework web clients provide access to public events in guest mode and to public and personal events in authenticated mode. All web clients are easily skinned allowing a high degree of customization.
    - **Public calendar suites :** Public events are displayed using "calendar suites" allowing multiple organizations to maintain their

own public views of events with whatever degree of visibility is appropriate. A Bedework public calendaring installation may have one or many calendar suites. A calendar suite provides a customized view of events, custom theming, and control over how events are tagged by event administrators.

  - **Public calendar feeds and embeddable JavaScript widgets,** supporting ical, json, and XML.
  - **Personal calendars :** Bedework provides a web client for personal and group calendaring including scheduling. Using CalDAV desktop clients, users can see a fully synchronized view of their personal and subscribed events between their desktop client and the web client.
  - **Administrative client for public events :** Public event entry and maintenance is carried out through the administrative web client. The system supports three roles: Super Users control global system settings including user and calendar maintenance and the setup of calendar suites. Calendar Suite Owners can modify the settings of their calendar suite, and Event Administrators can add and edit events for the administrative groups to which they belong.
  - **Public event community submission :** Bedework provides a web client for submitting events to a public queue allowing members of your community who are not event administrators to suggest public events.
  - **Highly customizable look and feel and standards based :** The web clients are themed using CSS and a theme settings file, or by deeper maniuplation of XSLT. Designers can theme Bedework for multiple clients and uses, without involving your programming staff. Bedework comes with skins for producing the web clients, data feeds, and displays suitable for handheld devices. Bedework provides a widget builder that makes it easy to embed dynamic event listings on static web sites.

- **Database independence - Hibernate :** The core of the calendar uses Hibernate for all database transactions giving support of many database systems and enterprise level performance and reliability. A number of caching schemes are implemented for Hibernate including clustered systems giving further options for improving availability.

- **Sharing :** Full CalDAV access control is available allowing the sharing of calendars and calendar entities based on authentication status and identity.

- **Scheduling :** Bedework supports scheduling of meetings including invitations and their responses. Caldav scheduling (still in draft) is supported. Freebusy is supported and the busy time is displayed as attendee lists are built. Access control allows users to determine who may attempt to schedule meetings with them.

- **Import and export :** Events can be imported and exported in iCalendar (RFC2445) format. This provides an option for populating the calendar from external sources. A dump/restore utility provides a means to backup and restore xml data files.

- **Calendar subscriptions :** Users may subscribe to calendars to which they have access, including public and personal calendars. iCalendar data feeds are available from the public web client.

- **Multiple calendars :** The core system supports multiple calendars for users and for public events.

- **Tagging & Filtering :** Events and folders can be tagged by any number of categories and event views, feeds, and widgets can be filtered by these.

- **Internationalization :** Bedework supports full internationalization, including multilingual content (though multilingual content creation is not yet exposed in the web UI).

- **Data feeds:** RSS, Javascript (e.g. json), iCalendar, and XML feeds are natively available, and custom feeds can be developed by writing an XSL skin. Feeds can be filtered by category or creator, and a feed and widget builder is available to help end users and developers design public feeds and embeddable event widgets.

- **Portal support :** Bedework has been shown to work as a JSR168 portlet in Jetspeed, uPortal and Liferay using the portal-struts bridge.

- **Timezone support :** Full timezone support is implemented. There is a set of system defined timezones based upon externally available sets of timezone definitions. In addition users are able to store their own timezone definitions.

- **Recurring events :** Extensive recurring event support is available via CalDAV and the web clients.

- **Event references :** Users may add public event references to their personal calendars. Event references can be annotated by the user.

- **Pluggable group support :** Bedework uses a pluggable class implementation to determine group membership for authenticated users allowing organizations to implement a class which uses an external directory. The default class uses internal tables to maintain group membership. Different implementations can be used for administrative and personal use allowing the separation of any given users roles.

- **Container authentication :** There is no authentication code in Bedework. Rather, Bedework behaves as a standard servlet, and all authentication is carried out through external mechanisms. Standard container authentication (via Tomcat or Jboss) and filter based Yale CAS authentication are used in production. The quickstart comes packaged with the Apache DS server against which the quickstart deployment of Bedework authenticates. This server can be used in production, though many deployers opt to authenticate against their organization's existing directory.

- **Support for other calendar systems and clients :** It is possible to access an entire calendar with a single url. This can be used to subscribe to a Bedework calendar from Google or Outlook. Bedework can also take advantage of the richness of CalDAV capable desktop clients such as Apple's iCal and Mozilla Lightning.

New Features in Bedework 3.9

Bedework 3.9 contains many enhancements. Among them:

- Image uploads (public events):
  public event admininstrators can upload an image to accompany an event, with an auto-generated thumbnail used in the "Upcoming" event listings.
- Event registration system (public events):
  a simple public events registration system is now included with Bedework, allowing end-users to register to attend public events.
- Multi-tenancy (public events):
  allows super-users to configure calendar suites "on the fly" without having to rebuild Bedework.
- Calendar suite resources (public events):
  allows calendar suite owners to manage images, CSS, or XML snippets from within the administrative user interface.
- Notifications:
  notification system for use with calendar sharing and public subscriptions
- Improved calendar sharing (personal calendaring):
  calendars can be shared by referencing a user account, and a notification will be sent to the specified user to accept (or reject) the calendar sharing invitation.
- Improved public subscriptions (public events/personal calendaring):
  users can subscribe to a public calendar (or event) that makes use of the new notification system
- Apache Solr (public events):
  Apache Solr available for both the search engine and for retrieving the event listings. It provides improved performance and allows for paging of upcoming events (to be implemented soon) and for faceted search.
- Miscellaneous bug fixes and enhancements

System Overview

Bedework System Architecture

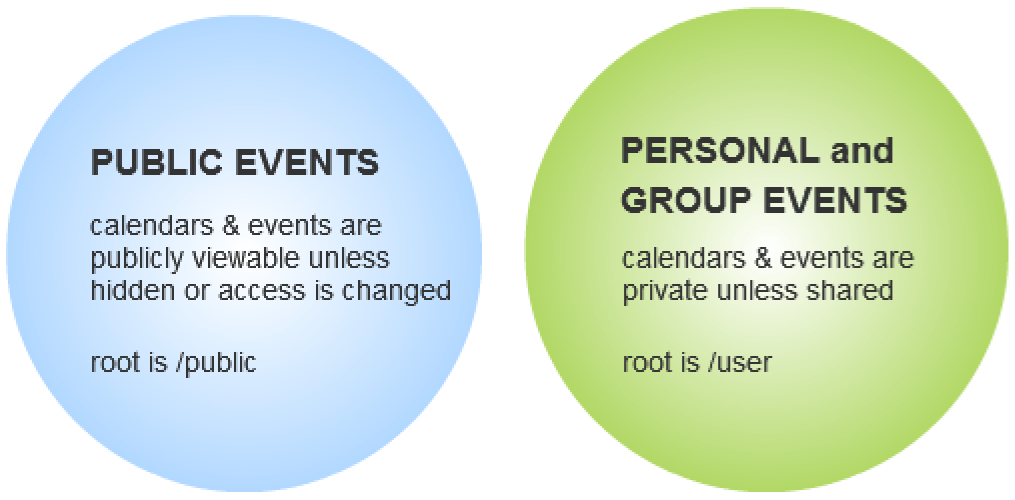
Bedework has a central server architecture and is modular and extensible. It consists of the following components:

- **Bedework core calendar engine**
- **Public events web client**, called a “Calendar Suite”, for display of public events
- **Public events cached feeds and widget builder,** supporting ical, json, and XML and for producing embeddable JavaScript widgets for use on other websites.
- **Public events administration web client** for entering public events, moderating pending events from the submissions client, and configuring calendar suites
- **Public events submission web client** for authenticated members of your community to suggest public events – these become pending events in the admin client
- **Personal and group calendaring web client** with a subscription model to Bedework public calendars, user calendars, and external calendar feeds
- **CalDAV server** for integration with CalDAV capable desktop (and web) clients such as Apple's iCal or Mozilla Lightning.
- **CardDAV server** that supports contacts for scheduling in the personal client.
- A JavaScript-only CardDAV **address book web client** is available for use with the CardDAV server. The address book comes with the Bedework personal web client, and is suitable for use with other web applications (e.g. webmail).

- **TimeZone server** for support of timezone information.
- **Dump/Restore command-line utilities** for database backup, initialization, and upgrades.

Calendar Architecture

The Bedework system is divided into two main spaces: the public events space, and the personal and group calendaring space which are kept separate by design. Public events are stored below a public calendar root folder and personal calendars are below a user calendar root folder.


PUBLIC EVENTS
calendars & events are publicly viewable unless hidden or access is changed
root is /public
PERSONAL and GROUP EVENTS
calendars & events are private unless shared
root is /user

Personal calendar users (and other clients) can subscribe to public events, but users may only add public events using the Administrative and Community Submissions web clients. For more information about Bedework's public and personal event calendaring models, see Public Events Calendaring and Personal & Group Calendaring.

CalDAV Server

Introduction

CalDAV (rfc4791) provides a protocol for interaction between calendar clients and servers, much like iMap provides such a protocol for email. CalDAV is built on top of WebDAV which is an HTTP based protocol. As a result, CalDAV inherits all of the advantages and disadvantages of those protocols.

What CalDAV adds to WebDAV is largely reporting but also some restrictions on the placement and handling of calendaring entities.

WebDAV is a protocol which is oriented towards document sharing. This works well enough as long as we remember the unit of information in CalDAV is not a calendar but a calendaring entity such as an event or task.

So, for example, we cannot use the PUT method to store an entire calendar consisting of many events or tasks. Nor, using GET, can we retrieve an entire calendar. Typically, to date, WebDAV sharing of calendar information has involved viewing an entire calendar as a single document which is retrieved, updated adn then stored. This is not the case with CalDAV.

The Bedework implementation

Bedework, at it's core, is not file system based, but uses a database for storage, retrieval and indexing. Events are not stored as a byte for byte image of rfc2445 calendar components but are stored in a relational database as rows and columns in tables.

CalDAV is not the only method used to access the data and there exists a certain tension between the needs of the different access methods.

Bedework is intended to be a complete implementation of CalDAV. At the moment we support most of the CalDAV operations. As more clients become available and more experience is gained in practical use of the protocol a practical working subset supported by all clients and servers will probably emerge.

The quickstart configuration has two CalDAV servers, a public unauthenticated server and the authenticated version used for personal calendars. As CalDAV is a WebDAV based protocol it is possible to retrieve appropriately permitted personal information via the unauthenticated server. This allows users to share their freebusy information with the world if they so wish.

CalDAV clients

A list of available desktop CalDAV clients is hosted at http://caldav.calconnect.org/implementations/clients.html by CalConnect, the The Calendaring and Scheduling Consortium.

Unsupported features

Some of these unsupported features reflect lack of support for some rfc features – others difficulty in providing support for CalDAV specific features. The list is also incomplete but over time will probably get shorter but more accurate.

Recurrence features

Recurrence id ranges

Recurrence id ranges take the values THISANDFUTURE or THISANDPRIOR. As yet this feature is not supported and CalDAV queries or updates using this feature will have uncertain results.

CardDAV Server

Bedework's CardDAV Implementation

Bedework's CardDav server is a standards-compliant CardDav server that stores* and serves out v4.0 vCards. See http://tools.ietf.org/html/rfc6350. Bedework's CardDav server can be configured to use a variety of vCard data stores.   Out of the box, it looks for private address books in an HSQLDB database that is packaged in the Bedework distribution and looks for public address books (one for people and one for locations) from the directory server (ApacheDS) that is packaged with Bedework. By default the server is available at http://mybedeworkserver.edu/ucarddav.

As a standards-compliant server, it should work well with many of the address book clients available on PC's and phones, however because many of them are based on V3.0 vCards, it may not.

*depending on whether the source is r/w or r/o.

Timezone Server

The Bedework system has a server that implements the latest Timezone standard:
http://tools.ietf.org/html/draft-douglass-timezone-service-04

The timezone server is essential to the running of the Bedework system. The server periodically updates itself from a primary server.

History and Background

Bedework History and Background

Bedework was established in March 2005 in succession to UW Calendar. In December 2006 Bedework received the Andrew W. Mellon Foundation's Technology Collaboration (MATC) Award. Since then the project has prospered, and in early 2009, Bedework became an incubator project of Jasig (http://www.jasig.org/).

Bedework is in use by institutions large and small, educational, governmental, commercial, and non-profit.

Bedework is named after the Venerable Bede, a highly influential monk and scholar from the area of Northumbria in Britain who in 725AD wrote the treatise, "On the Reckoning of Time". Bede is pronounced like “bead”, Bedework is pronounced “bead work”.

Calendaring Standards & Interoperability

Bedework is designed to conform to existing calendar standards. In particular:

- iCalendar: RFC 5545
  Internet Calendaring and Scheduling Core Object Specification
- iTIP: RFC 5546
  iCalendar Transport-Independent Interoperability Protocol
- CalDAV: RFC 4791
  Calendaring Extensions to WebDAV
- CalDAV scheduling (IETF draft)
  Scheduling Extensions to CalDAV
- CardDAV RFC 6352

vCard Extensions to Web Distributed Authoring and Versioning
- VVENUE: draft-norris-ical-venue-00
  Internet Calendaring and Scheduling Venue Component Specification

The Bedework implementation team participates in the standards process as a member of CalConnect, the Calendaring and Scheduling Consortium, which is "focused on the interoperable exchange of calendaring and scheduling information between dissimilar programs, platforms, and technologies."

Interoperability with other calendar systems and clients by way of standards compliance is an important design goal of the Bedework system.

For more information about calendaring and calendaring standards, please see http://www.calconnect.org.

Underlying technologies

The following lists some of the core technologies used by the Bedework platform:

- Apache Struts MVC
- CalDAV
- Cascading Style Sheets (CSS)
- iCAL: RFC 2445
- iCal4j
- Java Servlet API
- Java Server Pages (JSP)
- Portlet API: JSR-168
- XHTML
- XML
- XPath
- XSLT

See also Calendaring Standards

Getting Started

For both testing and production deployment, you should begin with the Bedework Quickstart .

This section provides the information you need to get Bedework up and running quickly.

The Bedework Quickstart

To deploy or try out Bedework, download and run the quickstart package. You can get the most recent release from the Jasig website: http://www.jasig.org/bedework/download.

The Bedework quickstart comes pre-built, allowing you to get familiar with the system quickly and easily, without having to compile and deploy code, and without having to set up a database. It is preloaded with with data to get you started and to provide an example of how you might structure your calendar environment. **It is the foundation from which a production release is built.**

We have attempted to make the quickstart as close to a production-ready system as possible. To move to deployment, you need only move to a production database. You may also choose to use local authentication (against ldap, CAS, Shib, etc), front Bedework with Apache, or make further customizations. In all cases, begin with the Bedework Quickstart. For more information about deploying a production system, see Deploying Bedework – but read this chapter first!

Packaged with the quickstart

1. Bedework: Calendar engine, CalDAV, CardDAV, Timzone servers, and web clients
2. JBoss 5.1 (jboss-5.1.0.GA)
3. Apache DS 1.5 (apacheds-1.5.3-fixed)
4. HSQL 1.8.0.7, packaged in JBoss
5. Apache ActiveMQ 5.3 (activemq-rar-5.3.0), packaged in JBoss
6. Apache Ant 1.7 (apache-ant-1.7.0)

System Requirements

1. Java Development Kit (JDK) 6 must be installed.
   Bedework is built and tested against Oracle's JDK. Other JDKs are not supported.

2. JAVA_HOME environment variable must be set.
   (*See Running Bedework*.)

Running Bedework

Start Bedework

1. *Open two console windows and cd to the quickstart directory in each.*

2. For each console, set JAVA_HOME environment variable. For example:
   - *Linux:* export JAVA_HOME=/usr/java/jdk1.7.0_21
   - *Windows:* set JAVA_HOME=C:\Program Files\Java\jdk1.7.0_21
   - *Mac OS X:* export JAVA_HOME=$(/usr/libexec/java_home)

3. Enter the following, one command line per console:
   a. Console 1: **./bw -quickstart dirstart** Starts ApacheDS ldap directory server for Bedework on port 10389
   b. Console 2: **./startjboss** or **./startjboss -debug** Starts Bedework core services in JBoss on port 8080 (and several others). Add -debug to increase the amount of information logged.

Memory settings:
JVM settings can be passed as parameters to the startjboss script. Enter "startjboss -usage" or see "Reviewing JVM parameters" for more information.

If you have the memory, try increasing the heap size for better performance, e.g. **./startjboss -heap 2G**.
If JBoss has difficulty coming up, try decreasing the heap size, e.g. **./startjboss -heap 512M**.

Windows Users:
Throughout this manual, you can replace all commands of the form “./command” with “command” or “command.bat”. For example, to start Bedework's directory server, enter: “bw -quickstart dirstart”, and to start Bedework enter "startjboss".

Helper files

goBedework-3.9.bat is a command line script that will launch all the above in Windows (edit the file to point at the location of your quickstart files).

Explore Bedework

1. Access the web applications at http://localhost:8080/bedework
   Note: this link will only work on the system on which the quickstart is running. If running the quickstart on a remote server, replace "localhost" with your server's hostname.

   This page also provides information about connecting to Bedework's CalDAV server.

Stop Bedework

1. When finished, you can stop Bedework by entering CTRL-C in the console windows.

Logging

Logging in Bedework

Log messages largely appear in the JBoss log, <jboss-dir>/server/default/log/server.log.   Bedework uses log4j for most application logging. The JBoss log4j configuration can be found in <jboss-dir>/server/default/conf/jboss-log4j.xml. It is configured to maintain a rolling log file (server.log) and also append output to the console. Remember to start JBoss with debugging on, if you want debugging output in the logs: "./startjboss -debug". (*See Running Bedework*).

If you are familiar with log4j, you can also dynamically change logging properties within the JBoss jmx-console, http://localhost:8080/jmx-console/

in jboss.system -> service=Logging,type=Log4jService

Authentication and user accounts

User accounts in Bedework are maintained either

- in the Apache DS ldap directory shipped with the Bedework Quickstart, or
- in your local enterprise directory.

You can perform authentication against either the Apache DS directory, your local directory, CAS, Shibboleth, or other service. If you plan on using a local service for authentication, please see Setting up authentication.

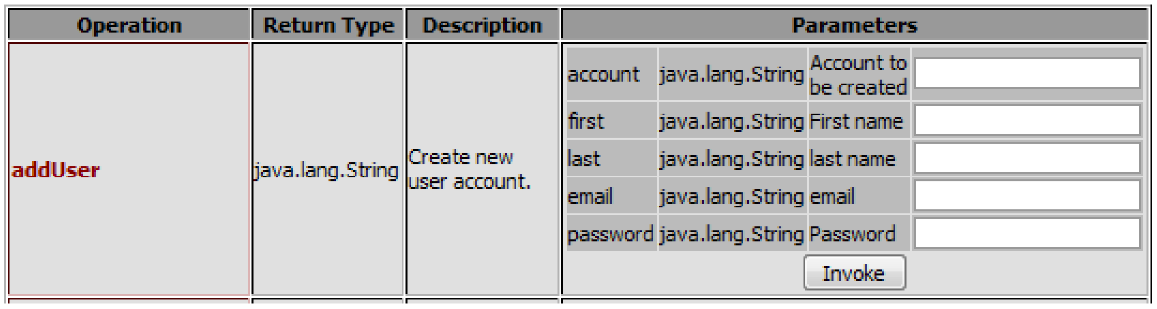
If you plan on using the Apache DS directory shipped with the quickstart, the information below explains how to add and edit user accounts in that system.

Default quickstart users
The quickstart ships with a number of default users and a super user "admin". The default user names and passwords are found on the Quickstart Jump Page, http://localhost:8080/bedework, under each application (or by looking in the quickstart Apache DS directory itself). You can modify, add, and remove users from the Apache DS directory using the information below.

Using the Apache DS ldap directory shipped with the Bedework Quickstart

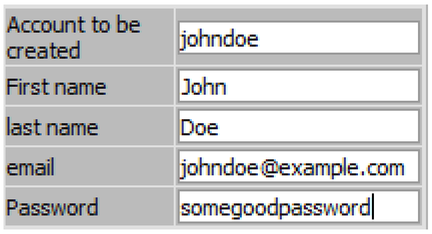
To add user accounts to the Apache DS ldap directory:

1. sign into the jmx-console (e.g. http://localhost:8080/jmx-console/)
2. click "org.bedework" in the left-most menu
3. click service=Selfreg
4. you should see the following form at the top of the table of Operations:



Operation	Return Type	Description	Parameters			
addUser	java.lang.String	Create new user account.	account	java.lang.String	Account to be created	
			first	java.lang.String	First name	
			last	java.lang.String	last name	
			email	java.lang.String	email	
			password	java.lang.String	Password	

5. add a user by filling in the fields and clicking "Invoke". For example:

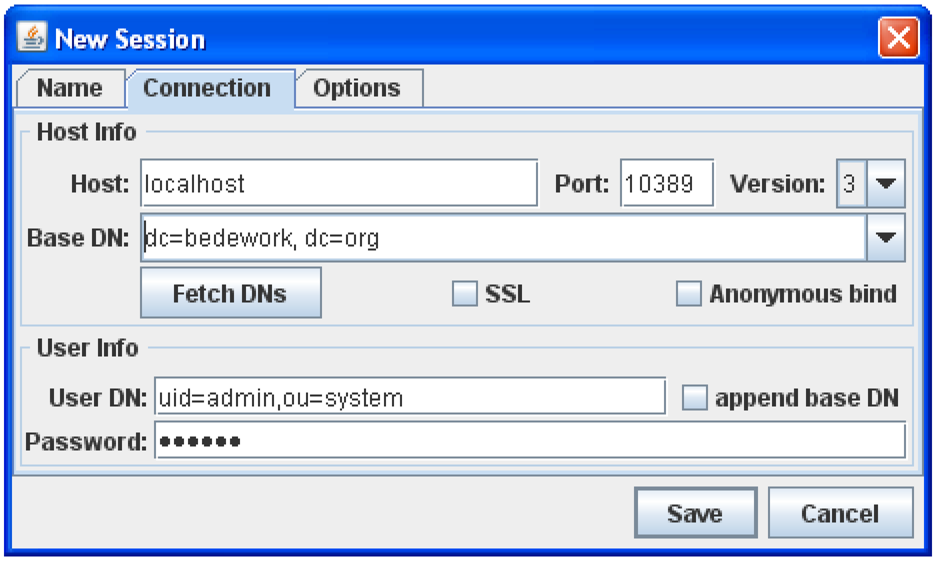


Account to be created	johndoe
First name	John
last name	Doe
email	johndoe@example.com
Password	somegoodpassword

If you plan to use Apache DS in production, secure it!
Before you move into production, you should change the LDAP password for the admin user. Please see "Securing Bedework".

Accessing the Apache DS LDAP Server

With Apache DS running, you can connect to the Apache DS server with any LDAP client using the following settings (password = "secret"):



Host Info
Host: localhost
Port: 10389
Version: 3
Base DN: dc=bedework, dc=org
SSL: ☐
Anonymous bind: ☐
User Info
User DN: uid=admin,ou=system
append base DN: ☐
Password: ●●●●●●

PASSWORD: secret

Dumping and restoring the database

Bedework provides utilities to dump and restore data in XML format. This provides a simple way to back up your work and get a feel for working with Bedework data.

Bedework Quickstart Data Files

The quickstart ships with two initialization files:
<jboss-dir>/server/default/data/bedework/dumprestore/**initbedework.xml**
<jboss-dir>/server/default/data/bedework/dumprestore/**initbedework-sparse.xml**

The “initbedework.xml” data contains a basic calendar structure, many categories, two calendar suites, and the handful of users & groups that allow the quickstart demonstration to run. It is a good starting point for most deployments. No events are included.

The “-sparse” version contains no events, no categories, no locations, no contacts, and the barest of calendar structures. It is a good starting point if you want to build up a system from scratch.

JMX Console

Bedework uses the JMX console built into JBoss to dump and restore data. In the default quickstart distribution, you get to the JMX console by navigating to “JMX Console” from the root of the JBoss server at http://localhost:8080/ or by directly visiting http://localhost:8080/jmx-console/

User name: admin
Password: bWjmx00 (if this password doesn't work, or to change the password (encouraged!), see Securing Bedework)

To dump a data file:

1. With Bedework running, go to the JMX console: http://localhost:8080/jmx-console/
2. Select “org.bedework” from the “Object Name Filter” menu to the left of the page.
3. Select “service=DumpRestore”
4. You will be presented with a form allowing you to manage the dump/restore process.
5. The “DataOut” attribute tells Bedework where to put the file. By default, this is in <jboss-dir>/server/default/data/bedework/dumprestore/
6. If you've changed any attributes, click “Apply Changes”
7. Click “Invoke” for the “dumpData” operation.
8. You should find a file with a name similar to “bwdata20100205T130857.xml” in your DataOut directory.

To restore a data file:

1. Stop the Bedework system if running.
2. Delete (or rename) the following directories
    a. <jboss-dir>/server/default/data/bedework/hypersonic/CalDb3p7.*

    b. <jboss-dir>/server/default/data/activemq
    c. <jboss-dir>/server/default/data/activemq-data
    d. <jboss-dir>/server/default/data/bedework/lucene
3. Start the Bedework system. The first three directories above should be recreated, and you **should see** an error in the log “Schema 'SA' does not exist”, which is HSQL complaining about not finding the Bedework schema that you've just removed.
4. Go to the JMX console: http://localhost:8080/jmx-console/
5. Select “org.bedework” from the “Object Name Filter” menu to the left of the page.
6. Select “service=DumpRestore”
7. You will be presented with a form allowing you to manage the dump/restore process. The “DataIn” attribute should point at the xml datafile you wish to restore. By default, this is the initbedework.xml file.
8. Initialize the schema:
    a. Set the attribute “Export” to True
    b. Set the attribute “Create” to True
    c. Click “Apply Changes” to set the values.
    d. Click “Invoke” for the “schema” operation.
9. Restore the data:
    a. Navigate back to the DumpRestore service (click “Back to MBean”)
    b. Click “Invoke” for the “restoreData” operation.

To create an empty system:

To install a system that contains no events, no categories, no locations, no contacts, and the barest of calendar structures, follow the steps above to restore initbedework-sparse.xml.

Building the Quickstart release

1. Open a console window and cd to the quickstart directory.
2. Set JAVA_HOME environment variable.
3. Enter the following command:

```
./bw -quickstart deploy
```

For details about the bw command and its options, see Deploying Bedework. You can also see a "usage" message by entering "./bw" with no parameters on the command line ("bw" for Windows users).

Deploying Bedework

This section explains how to deploy the Bedework system, including how to change the database, set up authentication, and secure the system.

To learn how to architect your public or personal calendaring environment, see Public Events Calendaring and Personal & Group Calendaring.

Readying your installation environment

Install the Bedework quickstart and test your environment

Before attempting any customization, please test your environment by running the quickstart release. See Getting Started. It is always wise to test your changes incrementally; test each small change to make certain you understand its effects. Doing these two things will help you understand the system and will provide useful information to the Bedework support community if you run into trouble and wish to ask for help.

Production builds begin with the Quickstart environment; use the Quickstart as the basis for your production release.

Satisfy the requirements

- JDK 6
- JAVA_HOME environment variable must be set in your command/shell windows
- Hardware for testing: most current desktop or laptops will be adequate.
- Hardware for production: A server class machine generally with at least 2Gigs allocated to the JVM (more is better).
- An adequate database system. In particular, the database should support Unicode. Most Bedework installations use PostgreSQL, MySQL or Oracle. The quickstart ships with HSQL. It isn't recommended for production systems. **Our recommended database is PostgresSQL.**

It is helpful to understand the following technologies:

- Servlet containers (e.g. Tomcat, JBoss) – JBoss, with embedded Tomcat, is the default container in Bedework.

Though not required, you may also find it useful to understand the following:

- Java servlets
- Some Java, should you need to interface to local systems not already supported by Bedework.

Install or ensure access to a supported database

Bedework uses Hibernate as a persistence engine. Bedework therefore should run on any database supported by Hibernate (see the listing at http://www.hibernate.org.) The list below reflects the systems on which Bedework has been successfully deployed and run:

- PostgreSQL version 8
- MySql version 5
- Oracle Version 10 and perhaps Version 9 with Version 10 jdbc drivers.
- HSQL (as shipped in the quickstart), although we don't recommend it for production systems.
- Apache Derby, although we don't recommend it for production systems.

Setting up the databases

Switch to another DBMS

The quickstart ships with HSQL. HSQL provides a good way to try out the system, but isn't recommended for production. To begin the process of setting up a production system, create four empty bedework databases (read below) using PostgreSQL, Oracle, or MySQL* – all known to work well. If you're considering using another sql database system, please review the notes regarding "Unsupported databases" below.

***DBMS Recommendation**
Note: as of Bedework 3.7, **PostgreSQL** is the preferred database system used by Bedework.
While MySQL is fully supported, there is a long-standing bug in MySQL that can lead to performance issues in Bedework.

Create five databases; check the character set

Create five databases, one for the Calendar server, the Timezone server, the Synchronization engine, the CardDAV server, and the Event Registration system.

Name them something like: *bedework3p9_cal*, *bedework3p9_tz, bedework3p9_synch, bedework3p9_card*, *bedework3p9_reg*.
Provide full access rights to an appropriately named user such as *bedework*.

Your database should be set to use a Unicode format, generally UTF-8. For most European and American systems this requirement may not appear necessary at first, but eventually problems will occur with special characters introduced by users or by events from other calendar systems. For most non-European languages Unicode or some other multi-byte support is an absolute necessity.

In Postgres the Bedework databases might look like this:
```
# \list
                                     List of databases

       Name       |  Owner   | Encoding |  Collation  |    Ctype    |   Access privileges

------------------+----------+----------+-------------+-------------+-----------------------

 bedework3p9_cal  | bedework | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/bedework

                                                                    : bedework=CTc/bedework

 bedework3p9_card | bedework | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/bedework

                                                                    : bedework=CTc/bedework

 bedework3p9_reg  | bedework | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/bedework

                                                                    : bedework=CTc/bedework

 bedework3p9_synch| bedework | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/bedework

                                                                    : bedework=CTc/bedework

 bedework3p9_tz   | bedework | UTF8     | en_US.UTF-8 | en_US.UTF-8 | =Tc/bedework
```

```
                                                          : bedework=CTc/bedework
```

Unsupported databases

Because we use Hibernate, we don’t support databases they don’t support. While Hibernate should make it possible to run Bedework on any Hibernate-supported database, in reality, there are problems with some systems that may require hand-editing of the schema. We expect in time to discover a workable solution to such problems.

Databases may be unsupported at the moment (or permanently) but it may still be possible to massage the schema enough to make Bedework run. We try to indicate why they are not supported and some will eventually move into the supported category.

- MS SQL Server: Partially supported but requires at least hand-editing of the schema. Has not been tried with 3.3.1 onwards. At least one problem remains, Sql Server does not follow the ANSI standard for unique indexes in that null=null for Sql Server but null is never equal to null in ANSI standard databases. This breaks some of the unique indexes in Bedework.
- MySQL version 4: Has problems in a number of areas. These are unlikely to ever be resolved. Version 4 is now old.

Developer notes

There are no checks that the application, e.g the calendar, has the same settings as the database. Databases will interpret the byte stream according to their configuration even if that does not match the configuration of the calendar server. Care in matching up all the components is critical. Those components are at least:

- The client (browser, caldav client etc)
- The servlet container (jboss, tomcat)
- The servlet (bedework)
- Jdbc settings
- The database

Configuring Bedework

Introduction

Preparing Bedework for production involves copying configuration files, modifying them for your site, and building Bedework. Bedework uses ant for the build and deploy process and a number of property files are used to control that process.

Copy the Configuration Directories

Copy the distributed configurations directory to your home directory

Bedework uses a configuration directory to store multiple configurations. The quickstart copy of this directory is at *<quickstartDirectory>/bedework/config/bwbuild*.

Copy the entire directory structure – bwbuild on down – into your home directory (*/home/userid* on many Unix systems and *C:\Users\userid* on a typical Windows system).   So, for example, the bwbuild directory in windows would live here: *C:\Users\userid\bwbuild* or here: */home/userid/bwbuild*

Inside ***bwbuild*** you will find a number of example configurations in subdirectories, for example ***default*** and ***jboss-mysql***. ***default*** is configured to use HSQL.

Do I need to copy the whole bwbuild configuration tree?
Yes. There are hidden directories – .platform and .defaults – that must be present in your configuration directory.   After you copy the bwbuild tree, you can remove any configurations you aren't using.

Can I copy the configs outside of /home?
Yes. You may choose to copy the bwbuild directory structure to another directory. Some prefer to keep the configuration files in the vicinity of their Bedework files. If you do so, add -bwchome <config-directory> to your Bedework build commands. This will become clearer as you read on.

Copy one of the configurations to create your own

Choose the sample configuration that is the best fit – bwbuild/jboss-postgresql, for example, because you plan to use postgresql – and make a copy of the entire directory tree. You might name your copy for your organization: bwbuild/*mycompany* or bwbuild/*myuniversity*.

> **bwbuild/myconfig**
> Your configuration directory will be referred to as **bwbuild/*myconfig*** for the remainder of this documentation.

Link to Your Bedework Databases

Edit the five bedework-*server*-ds.xml files to connect Bedework to your database

If you have based your configuration on one of the samples, you need to edit these five configuration files:

1. bwbuild/*myconfig*/***bedework-card-ds.xml***: connection information for the Bedework CardDAV data source
2. bwbuild/*myconfig*/***bedework-ds.xml***: connection information for the Bedework calendar server data source
3. bwbuild/*myconfig*/***bedework-eventreg-ds.xml***: connection information for the Bedework event registration system data source
4. bwbuild/*myconfig*/***bedework-synch-ds.xml***: connection information for the Bedework synchronization server data source
5. bwbuild/*myconfig*/***bedework-tz-ds.xml***: connection information for the Bedework timezone server data source

In each case, within the <datasource> stanza:

1. edit <connection-url> to set the database host, port number, and database name.
2. edit <user-name> (the database user) and <password> (the database user's password).

Add JDBC Drivers to JBoss (in most cases, done for you)

Again, if you've based your configuration on one of the samples, you should have an appropriate JDBC driver in bwbuild/*myconfig*/lib/server. During the build, that driver will be copied into <quickstart>/jboss-5.1.0.GA/server/default/lib.   If you'd like to update that driver, replace the one in bwbuild/*myconfig*/lib/server, then rebuild.

Non-standard configurations

The following steps will help you configure a non-standard / unsupported database with Bedework.

Configure Hibernate to work with your database's SQL dialect

Configuring Hibernate to use the appropriate dialect is done in bwbuild/*myconfig/***cal.properties**. You need to set a few properties

org.bedework.global.hibernate.dialect=***yourDialect***

The value ***yourDialect*** is a defined Hibernate SQL dialect such as org.hibernate.dialect.HSQLDialect or org.hibernate.dialect.MySQL5Dialect. The dialect is a class defined on the class path. Hibernate defines a number of 'standard' dialects.

For example, the global property if using MySQL5 would be

```
org.bedework.global.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

A list of dialects understood by Hibernate can be found at:

http://docs.jboss.org/hibernate/core/3.3/reference/en-US/html/session-configuration.html#configuration-optional-dialects

You can also look at the org.hibernate.dialect classes in the Hibernate jar file (for example, to use MySQL 5 which is not currently listed in the on-line Hibernate documentation, use org.hibernate.dialect.MySQL5Dialect).

Set Calendar and Build Preferences

During a normal Bedework install, you'll configure two files prior to building the system:
cal.properties and **cal.options.xml**

Override default properties in cal.properties

Inside the bwbuild/.defaults directory, next to your configuration directory, you'll find a file named bwbuild/.defaults/cal.properties. cal.properties is referenced during the Bedework build and during deployment.   You can override any of those properties by creating a bwbuild/*myconfig/cal.properties in your configuration directory and editing in the overrides.* ***cal.properties*** *is divided into sections with different property prefixes.*

Under Install Options, prune the org.bedework.install property

org.bedework.install defines which applications are built and deployed to jboss. For each name on the list, there should be a corresponding section prefixed with “org.bedework.app.<name>” and also a corresponding section in the options file.

Copy ***org.bedework.install*** to your override file, and remove any applications that you don't need.   For example, if you're using Bedework for public events (and not for personal calendars), you can remove UserCal from the list. You probably want to remove " SoEDept", too; it's an example.   When you remove an application from the list, you can also remove the corresponding stanza further down the file, but it isn't necessary. Those lines won't be referenced.

Review the Global Section of the file

The section prefixed “*org.bedework.global*” defines properties global to the whole deployment process. The values set in this section work for most sites.

Review the Application Section of the file

This is where the per-application properties are set. The values set in these in this section also work for most sites.

Multiple versions of each application type may be deployed, each configured differently. You can see how this works by comparing the stanza for the "Demo departmental public Web Client" to the stanza for the "Public Web Client".

Copy, then set two run time properties in cal.options.xml

It is important to set the system run time properties for a new Bedework installation. These are found in ***cal.options.xml*** within <syspars classname="org.bedework.calfacade.BwSystem">.

1. Copy the entire file bwbuild/.defaults/***cal.options.xml*** into your configuration directory:
   cp bwbuild/.defaults/cal.options.xml bwbuild/*myconfg*/cal.options.xml

2. So that your local changes will get picked up by the build, edit bwbuild/*myconfig*/build.properties and change the following line:
   org.bedework.config.options=${org.bedework.configuration.defaults}/cal.options.xml
   to be: org.bedework.config.options=${env.BEDEWORK_CONFIG}/cal.options.xml

Most options in cal.options.xml can be left with the default values and some are not yet implemented. The two values it is important to set (and their default settings) are:

Set the default timezone:

Look for:

```
<tzid>America/New_York</tzid>
```

The tzid is the default timezone to be used for times and dates. To get a list of timezone ids to choose from, you can call Bedework's timezone server directly, e.g.: http://localhost:8080/tzsvr/?names

Set the system ID:

Look for:

```
<systemid>demobedework@mysite.edu</systemid>
```

systemid is used when generating uids for calendar entities. This name should relate to your organization for ease of identification. If you run multiple Bedework systems, use a different value for each. In addition, the systemid takes part in the creation and interpretation of calendar user addresses which appear in attendees. The part following “@” will probably be the domain to which imip messages are addressed (in some as yet undefined manner).

For example, your systemid might be "bedework@myorganization.com" or "bedework@myuniversity.edu".

Calendar user addresses take the form of a “mailto:” uri so that user “testuser01” on a system configured as above would have a calendar user address of testuser01@cal.mysite.edu

cal.options.xml contains run time properties and is divided into sections much like the cal.properties file. Most of the options are used to set field values in named classes so that the application will load the settings only once with a single call. In the future, most of these options will be set through a web interface on the running system.

Note: if you plan on *building* multiple calendar suites, you'll need to add sections for each suite in both the cal.properties and cal.options.xml files. See the section on Calendar Suites.

Building Bedework

Building and Deploying Bedework

The Bedework release is made of up a number of related parts which are built and deployed through the action of the ***bw*** (bw.bat on Windows) script found in the quickstart directory.

1. ***To run bw, first cd to the quickstart directory.***

The **bw** commands that immediately follow deploy the bedework core: the calendar server (including the caldav server) and the web clients. You'll need to reinvoke the **bw** command with different arguments to build and deploy the timezone server, the carddav server, the addressbook client, etc. Those commands are covered in the sections below.

Invoking the bw script without arguments displays very useful usage information.

Apart from the Bedework core, only the timezone server is critical to getting Bedework running. You'll need to rebuild the timezone server after you change <*timezoneUri*> in bwbuild/*myconfig/cal.options.xml*.

Assuming your configuration is bwbuild/*myconfig* in your home directory, run this command:

```
./bw -bwc myconfig clean.deploy
```

If you created a configuration area somewhere else in your filesystem, then

```
./bw -bwchome configdir -bwc myconfig deploy
```

To build with the jboss-mysql configurations shipped in the quickstart:

```
./bw -quickstart -bwc jboss-mysql deploy
```

These commands build a number of EAR and WAR files in ***<quickstart>/bedework/dist/***, then deploy them to ***<quickstart>/jboss-5.1.0.GA/server/default/bwdeploy***

If you build with "clean", the build will remove all the previously built files before building.

```
./bw -bwc myconfig clean.deploy  <-- builds with clean

./bw -bwc myconfig deploy  <-- builds without clean
```

Build and Deploy the Bedework Timezone Server

As with the above examples, unless your configuration files are under <home>/bwbuild, specify where the configuration files come from

(-quickstart/-bwchome) and unless you are using the "default" configuration, specify the name of your configuration folder (-bwc).

```
./bw <config files args> -tzsvr
(e.g. ./bw -bwc myconfig -tzsvr )
```

This command builds bw-tzsvr.ear and deploys it to ***jboss-5.1.0.GA/server/default/bwdeploy.***

Build and Deploy the Bedework CardDAV Server

Use -quickstart, -bwchome, -bwc as appropriate.

```
./bw <config files args> -carddav
```

This command builds **bw-carddav.ear** and deploys it to ***<qs>/jboss-5.1.0.GA/server/default/bwdeploy.***

Deploy the Bedework Addressbook Client

Use -quickstart, -bwchome, -bwc as appropriate.

```
./bw <config files args> -carddav deploy-addrbook
```

This command copies the files in **<qs>bedework-carddav/clients/javascript/bwAddrbookClient** to ***<qs>/jboss-5.1.0.GA/server/default/deploy/ROOT.war.***

Deploy the Bedework Synch Server

Use -quickstart, -bwchome, -bwc as appropriate.

```
./bw <config files args> -synch
```

This command builds the **bw-synch.ear** and deploys it to ***<qs>/jboss-5.1.0.GA/server/default/bwdeploy.***

Build and Deploy webcache/urlbuilder

See this section.

Hot Deploys

If you build and deploy Bedework while Bedework is running, JBoss <<should>> undeploy what was already running in favor of the newer packages. Often it works as advertises. Sometimes it doesn't. Stopping Bedework, re-deploying, and starting Bedework is safer.   If you succeed in "hot deploying", you might want to schedule a restart for the next convenient time.   Even successful hot deploys are likely to result in a larger memory footprint.

Initializing your databases

Start Bedework

Having rebuilt Bedework with your configuration, start up Bedework in order to get to the JMX console. From the JMX console, you have access to the Bedework utilities (mbeans) that create the Bedework database tables and inject some starter data.

Once Bedework is up and running, visit ***http://yourserver:8080/jmx-console***.

Export the schema, create the database tables, and inject some starter data into the Bedework Database

1. Select “org.bedework” from the “Object Name Filter” menu to the left of the page.
2. Select “service=DumpRestore”

3. You will be presented with a form allowing you to manage the dump/restore process. The “DataIn” attribute should point at the xml datafile you wish to restore. By default, this is the initbedework.xml file.
4. Initialize the schema:
    a. Set the attribute *Export* to **True**
    b. Set the attribute *Create* to **True**
    c. Click **Apply Changes** to set the values.
    d. Click **Invoke** for the *schema* operation.
5. Restore the data (Read the note below)
    a. Navigate back to the DumpRestore service (click **Back to MBean**)
    b. Click **Invoke** for the *restoreData* operation.

The quickstart ships with two initialization files:

- <jboss-dir>/server/default/data/bedework/dumprestore/**initbedework.xml**
- <jboss-dir>/server/default/data/bedework/dumprestore/**initbedework-sparse.xml**

initbedework.xml contains a basic calendar structure, many categories, two calendar suites, and the handful of users & groups that allow the quickstart demonstration to run. It is a good starting point for most deployments.

initbedework-sparse.xml contains no events, no categories, no locations, no contacts, and the barest of calendar structures. Unless you are a Bedework expert, you'll want to use the other one.

The default is ***initbedework.xml.*** You can override it by adding -sparse to the xxxx field before invoking *restoreData*.

Initialize the Bedework Timezone Server Database

1. Select “org.bedework” from the “Object Name Filter” menu to the left of the page.
2. Select "service=Tzsvr”
3. Look for *recreateDb* in the bottom section of the form*. C*lick **Invoke.**
4. Look for *refreshData* in the bottom section of the form*.* Click **Invoke.**

Initialize the Bedework Carddav Server Database

1. Select “org.bedework.carddav” from the “Object Name Filter” menu to the left of the page.
2. Select “Name=user-dirHandler,Type=dirhandler,service=CardDav”
3. Set hibernate dialect. There are two parameters to the SetHibernateProperty operation:
    a. Set the name parameter to hibernate.dialect
    b. Set the value parameter to the db dialect you want.
        i. Hibernate class names for the dialect take the form org.hibernate.dialect.X
           For example:
           for hsql (as shipped) - *org.hibernate.dialect.HSQLDialect*
           for postgresql - *org.hibernate.dialect.PostgreSQLDialect*
           for oracle10g - *org.hibernate.dialect.Oracle10gDialect*
           for mysql - *org.hibernate.dialect.MySQL5InnoDBDialect* (innodb is required)
    c. Click **Invoke** for the operation.
    d. Click the **Back to Mbean** button
    e. Click **saveConfig** to make it permanent
    f. Click the **Back to Mbean** button
    g. Probably safer to do a restart of jboss at this point
4. Initialize the schema:
    a. Set the attribute *Export* to **True**
    b. Click **Apply Changes** to set the values.
    c. Click **Invoke** for the *schema* operation.
    d. Check the console for errors

Initialize the Bedework Sync Server Database

1. Select “org.bedework” from the “Object Name Filter” menu to the left of the page.
2. Select “service=Synch”
3. Initialize the schema:

a. Set the attribute *Export* to **True**
b. Set the attribute *Create* to **True**
c. Click **Apply Changes** to set the values.
d. Click **Invoke** for the *schema* operation.

Setting up authentication

Introduction

The Bedework quickstart authenticates to the packaged ApacheDS directory server. Most sites choose to authenticate their Bedework users against their centralized directory service. For installations that will not be authenticating to a local domain, the Apache DS server shipped with Bedework is acceptable for a production deployment. Add users to ApacheDS using the tools outlined in Getting started Authentication and user accounts, and add superusers to the System tab "Manage system preferences" page in the admin client.

The calendar uses container-based authentication as defined by the Java servlet specification. There is no authentication code within the calendar system.

The authentication method used by JBoss is defined in

```
<JBoss>/server/default/conf/login-config.xml
```

In the quickstart, the connection to ApacheDS is defined in login-config.xml by <application-policy name="bedeworkdemo">. The web applications are configured to use a particular application-policy in ***cal.properties:***

```
org.bedework.app.webapp.security.domain=bedeworkdemo
```

Moving to local LDAP authentication involves copying or modifying the *<application-policy>* block.

Other forms of authentication can be managed by the servlet container in a number of ways which are beyond the scope of this document. The JBoss website provides documentation on configuring JBoss to use other forms of authentication, including active directory, a local file or databases.

An alternative which has been implemented is to use filter based Yale/JA-SIG CAS.

Step 1: Add a superuser

Before you switch from the quickstart's authentication to your site's authentication you need to ensure you have an administrative superuser that exists within your own authentication domain.

Two ways to achieve this are to:

1. Create a Bedework superuser with an account that already exists in your authentication domain. To achieve this, log into the administrative web client (http://localhost:8080/caladmin) as user "admin" (default password "bedework"), select the "System" Tab "Manage system preferences" and add the accounts that exist in your local domain. **Do not remove the "admin" user from the list of superusers**, as a number of services rely on that account (though it does not need to authenticate or exist in your directory).

2. Create a user in your domain, e.g. "admin", that is already set up as a superuser in the Bedework quickstart.

Option 1 is probably the most appropriate, and as the deployer it is probably acceptable to use your own account, at least initially.

Option: Use LDAP for authentication

1. Configure JBoss to point at your local LDAP server:
   a. Edit <quickstart>/jboss-5.1.0.GA /server/default/conf/login-config.xml
   b. Modify the <application-policy name="bedeworkdemo"> section found near line 110 to point to your local LDAP server. This will typically involve modifying three properties:
      - principalDNPrefix, e.g. "uid="
      - principalDNSuffix, e.g. ",ou=accounts, dc=rpi, dc=edu"
      - java.naming.provider.url, e.g "ldap://login.myserver.edu/"

2. Prepare your group properties by modifying */home/<userid>/bwbuild/jboss-mysql/**cal.options.xml***:** Set the <user-ldap-group> options on or around lines 68-87

3. Rebuild the system:
    a. stop JBoss
    b. build Bedework
    c. restart JBoss

4. Test the web clients. Login to the admin or user client to test Ldap auth.

If you choose to change the name “bedeworkdemo” to something more reasonable for your site, you must update the references to "bedeworkdemo" in bwbuild/cal.options (four places) to reflect the new policy name.)

Option: Use CAS for authentication

1. Download the CAS Java Client
2. Unzip the file and copy the CAS client to Bedework:
    a. unzip cas-client-3.1.10-release.zip
    b. cd cas-client-3.1.10/modules
    c. cp cas-client-core-3.1.10.jar <qs>/jboss-5.1.0.GA/common/lib/

3. Add a CAS filter to each application that requires a log in (eventsubmit, ucal, caladmin). ***Do caladmin last; you may want access to it while you are debugging this procedure.*** The files to edit are:

```
<qs>/bwwebapps/websubmit/war/WEB-INF/web.xml
<qs>/bwwebapps/webclient/war/WEB-INF/userweb.xml
<qs>/bwwebapps/webadmin/war/WEB-INF/web.xml
```

   In each file, add the lines in the "CAS Filters" box at the bottom of this page, replacing the URLs with those that work at your site. The **CAS filters must come BEFORE any other filters**. Also, delete any and all of the following elements:
    a. *<security-role-ref>*
    b. *<security-constraint>*
    c. *<security-role>*

4. (optionally) Point the logout buttons on the Bedework clients that have them (Admin Client, Personal Calendar Client, Submissions Client) at your CAS server by editing the corresponding stylesheets. For example, in the Submissions Client (<quickstart>/bedework/deployment/websubmit/webapp/resources/demoskins/default/default/default.xsl), change the logout "span" to look something like this:

```
<span class="logout">
    <a
href="https://yourserver:8443/cas/logout?service=http%3A%2F%2Fyourserver%2Fevents
ubmit%2F"
            id="bwLogoutButton">
         <xsl:copy-of select="$bwStr-Hedr-Logout"/>
   </a>
</span>
```

5. build Bedework

CAS Filters:

```
<filter>
  <filter-name>CAS Authentication Filter</filter-name>
```

```
<filter-class>org.jasig.cas.client.authentication.AuthenticationFilter</filter-class>
  <init-param>
    <param-name>casServerLoginUrl</param-name>
    <param-value>https://myCasServer/login</param-value>
  </init-param>
  <init-param>
    <param-name>service</param-name>
    <param-value>http://myclient/</param-value>
  </init-param>
  <init-param>
    <param-name>serverName</param-name>
    <param-value>http://myBedeworkServer</param-value>
  </init-param>
</filter>

<filter>
  <filter-name>CAS Validation Filter</filter-name>

<filter-class>org.jasig.cas.client.validation.Cas20ProxyReceivingTicketValidationFilte
r</filter-class>
  <init-param>
    <param-name>casServerUrlPrefix</param-name>
    <param-value>https:/myCasServer/cas</param-value>
   </init-param>
   <init-param>
    <param-name>serverName</param-name>
    <param-value>http://myBedeworkServer</param-value>
   </init-param>
</filter>

<filter>
  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>

<filter-class>org.jasig.cas.client.util.HttpServletRequestWrapperFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>CAS Authentication Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CAS Validation Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>CAS HttpServletRequest Wrapper Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<filter>
  <filter-name>CAS Sign-Out Filter</filter-name>

  <filter-class>org.jasig.cas.client.session.SingleSignOutFilter</filter-class>
</filter>

<filter-mapping>
  <filter-name>CAS Sign-Out Filter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

<listener>

<listener-class>org.jasig.cas.client.session.SingleSignOutHttpSessionListener</listene
```

```
r-class>
</listener>
```

Securing Bedework

Using HTTPS For Those Clients that Accept Passwords

If another system, such as CAS or Shibboleth, handles your passwords, then you should be all set. However, if you use Bedework's directory server or your organization's LDAP servers, you need to secure your Bedework logins with https. If you are fronting JBoss with Apache , then Apache can handle this.

If your users are accessing JBoss directly:

1. Configure JBoss to use your SSL Certificate by editing <quickstart>/jboss-5.1.0.GA/server/default/deploy/jbossweb.sar/server.xml

2. Edit bedework.properties in your Bedework configuration directory. Look for lines that end in *transport.guarantee=NONE*. There are several, one for each client that requires a login. For any that you are using, change the value NONE to CONFIDENTIAL.

3. Rebuild Bedework.

Securing JBoss's JMX Console

The jmx-console security domain is defined in:<quickstart>/jboss-5.1.0.GA/server/default/conf/login-config.xml.

Credentials are read from:jboss-5.1.0.GA/server/default/conf/props/jmx-console-users.properties. Change the password in this file to something more secure than the default.

If you change the login id (e.g. from *admin* to *someotherid*), change or add *someotherid* to jboss-5.1.0.GA/server/default/conf/props/jmx-console-roles.properties.

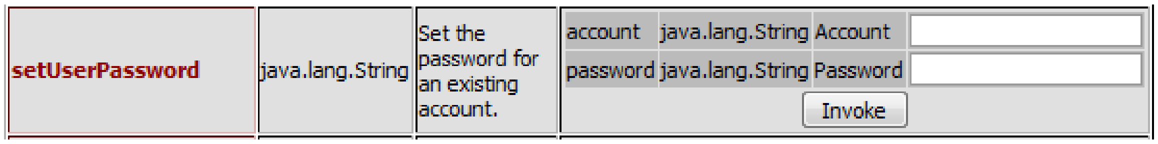
Securing JBoss's Web Console

Securing the web console is the same as securing the JMX Console. At its simplest, modify the password injboss-5.1.0.GA/server/default/conf/props/jbossws-users.properties
(and likewise, if you change the userid, add the role to jbossws-roles.properties).

Securing the Apache DS LDAP Server Shipped with the Quickstart

If you choose to use the LDAP server shipped with the quickstart to maintain user accounts, you should **1)** change the admin password and **2)** re move user accounts you don't need (or change their passwords). *If you intend to use your own enterprise directory server, you can safely ignore this section.*

Change the admin password using the JMX-Console

1. sign into the jmx-console (e.g. http://localhost:8080/jmx-console/)
2. click "org.bedework" in the left-most menu
3. click service=Selfreg
4. you should see the following form in the table of Operations:


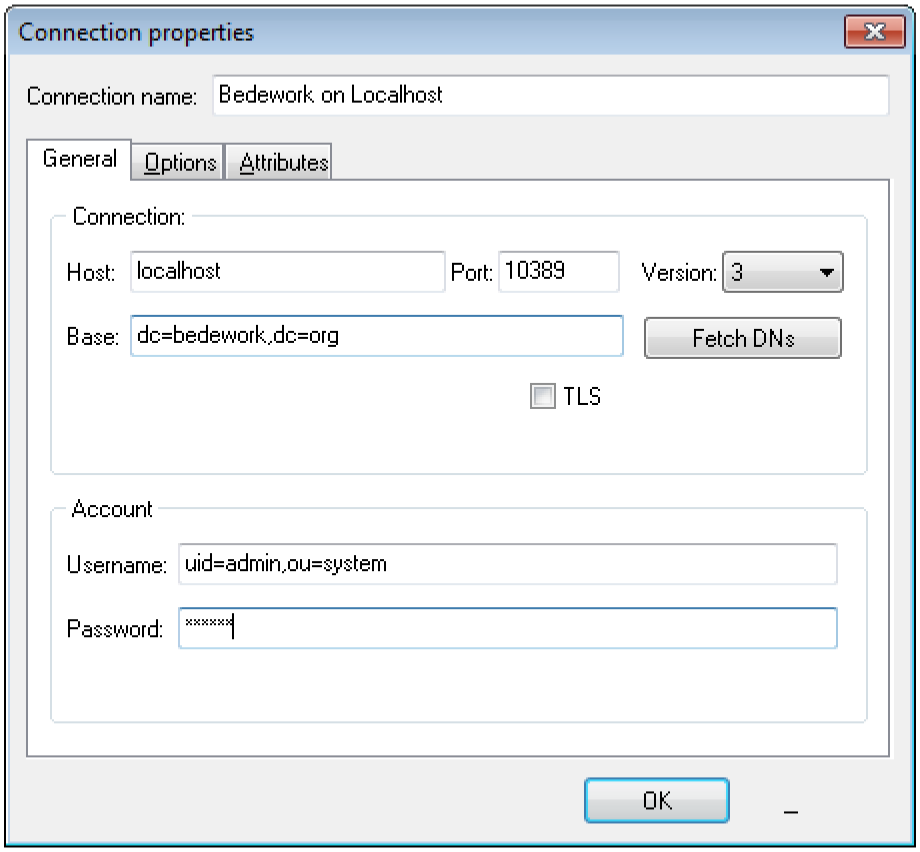
setUserPassword	java.lang.String	Set the password for an existing account.	account	java.lang.String	Account	
			password	java.lang.String	Password	

5. enter the account "admin", select a new password, and click "Invoke"

You can use this approach to change any account password in Apache DS.

Change the admin password using an LDAP client

Use an LDAP client (for example, LDAP Admin) and sign in using the following settings (password = "secret"):


Connection properties

Connection name: Bedework on Localhost

General | Options | Attributes

Connection:

Host: localhost
Port: 10389
Version: 3
Base: dc=bedework,dc=org
Fetch DNs
TLS

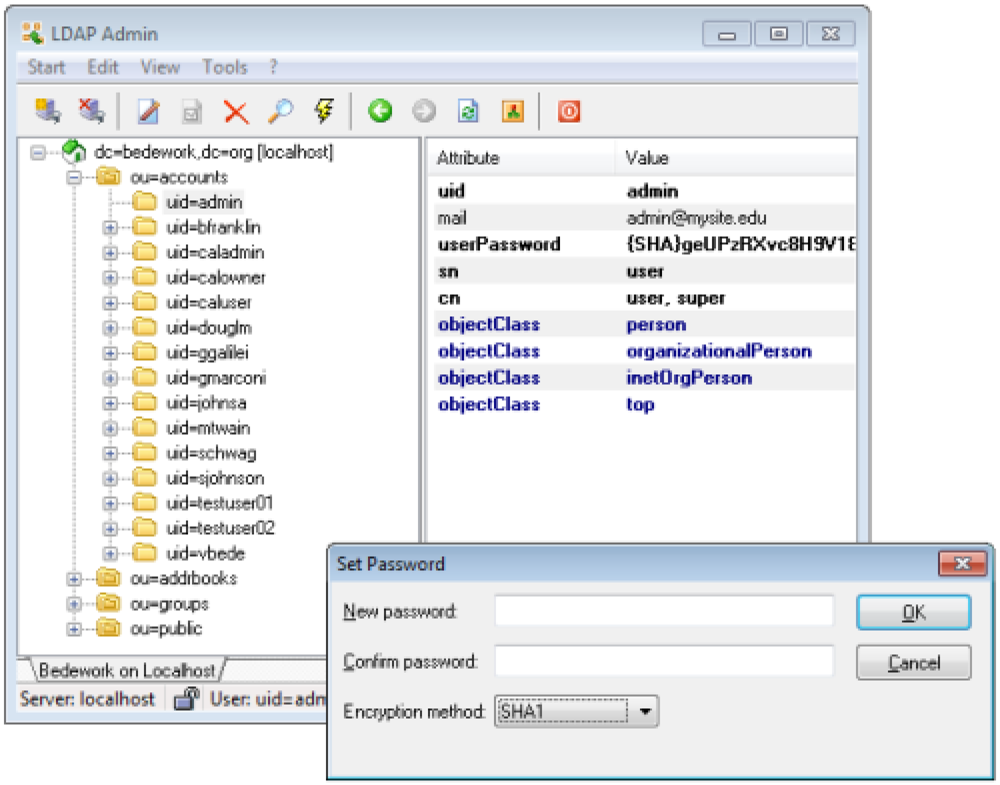
Account

Username: uid=admin,ou=system
Password: ******

OK

PASSWORD = secret

Once connected, select the admin account, and set the password on the account using the client. For example, if you are using the LDAP Admin client, open up ou=accounts, right-click on uid=admin, and select "Change Password". You'll see something like this:



dc=bedework,dc=org [localhost]
- ou=accounts
  - uid=admin
  - uid=bfranklin
  - uid=caladmin
  - uid=calowner
  - uid=caluser
  - uid=douglm
  - uid=ggalilei
  - uid=gmarconi
  - uid=johnsa
  - uid=mtwain
  - uid=schwag
  - uid=sjohnson
  - uid=testuser01
  - uid=testuser02
  - uid=vbede
- ou=addrbooks
- ou=groups
- ou=public

Bedework on Localhost

Server: localhost   User: uid=adm

Attribute	Value
uid	admin
mail	admin@mysite.edu
userPassword	{SHA}geUPzRXvc8H9V1E
sn	user
cn	user, super
objectClass	person
objectClass	organizationalPerson
objectClass	inetOrgPerson
objectClass	top

New password:

Confirm password:

Encryption method: SHA1

Remove user accounts you don't need (or change their passwords)

In the screenshot above, you'll find many default accounts shipped with Bedework: uid=bfranklin, uid=vbede, etc. Prior to production, you should remove these accounts or change their passwords. (For example, in the client above, you can right-click on uid=testuser02 and select "delete"). *NOTE: it is not yet possible to remove accounts using the JMX-Console. If you don't wish to connect using an LDAP client to remove the default accounts, you can use the JMX-Console to simply change their passwords.*

If you plan on removing the admin account, be sure to set another user as superuser using the Bedework Administrative web client before you do (or you'll have to add the admin account back to LDAP). (Look in the Admin Client under System --> Manage System Preferences.) If you're unsure, keep the admin account and change the password, as described above. See also: Setting up authentication

Reviewing JVM parameters

Java servers work best when provided plenty of memory. The "./startjboss" script in the root of the quickstart is configured to start with 600mb of memory by default. This can be decreased for running on a laptop or increased for production (e.g. 4G) using the command line parameters with the ./startjboss script.

From the top of the quickstart,

- Unix: /startjboss [-heap size] [-newsize size] [-permsize size]
- Windows: startjboss.bat [-heap size] [-newsize size] [-permsize size]

The default settings are currently: heap = 600M, newsize = 200M, and permgen = 256M.

You may also choose to edit the default values in the ***startjboss*** script directly in:

- Unix: <quickstart>/bedework/build/quickstart/linux/startjboss
- Windows: <quickstart>\bedework\build\quickstart\windows\startjboss.bat

Call ***startjboss*** with -usage to get a complete list of options.

Themes and Resources

Bedework themes, resources, and other static content

The Bedework web client themes in the quickstart are deployed into and served from JBoss. They can be left there, or they can be placed on a web server more accessible to your web designers. Placing the stylesheets and resources on a separate web server (for example, an Apache server on the same host) may make them more convenient to access and manipulate.

For information about customizing the Bedework web client themes, see Theming Bedework.

Copying themes and resources to a separate web server (optional)

Should you choose to copy the themes to a separate web server, the folders to copy are are found in the Bedework source at

bedework/deployment/**webadmin**/webapp/resources/
bedework/deployment/**webpublic**/webapp/resources/demoskins/
bedework/deployment/**webuser**/webapp/resources/demoskins/
bedework/deployment/**websubmit**/webapp/resources/demoskins

Change the cal.properties file

The destination where you copy these directories is a URL specified in ***cal.properties*** inside your configuration directory by the property **app.*name*.cal.suite** for the public calendar suites and **app.*name*.root** for the other web applications. The ***root*** is where the server discovers and caches the xsl transforms.

The ***root*** should never be served over https.

Given the values in the properties file:

```
org.bedework.app.UserCal.root=
{nolink:http://somewebserver/bedework-version/ucalrsrc}

org.bedework.app.CalAdmin.root=
{nolink:http://somewebserver/bedework-version/caladminrsrc}

org.bedework.app.Events.root=
{nolink:http://somewebserver/bedework-version/calrsrc}

org.bedework.app.Events.cal.suite=MainCampus

...
```

the user client and administrative client stylesheets will be found at the url shown. Bedework will look for the calendar suite stylesheets in a directory that is a concatenation of the root and the calendar suite name. In the example above, the “Events” calendar suite will have its stylesheets located at http://somewebserver/bedework-version/calrsrc.MainCampus ; the “SoeDept” calendar suite at http://somewebserver/bedework-version/calrsrc.SoeDept

Change the cal.options.xml file

Once set in the cal.properties file, you may also need to set the properties ***<appRoot>*** and ***<browserResourceRoot>*** in ***cal.options.xml*** within your configuration directory.

Look for instances of: **<browserResourceRoot>*/name*</browserResourceRoot>**

and instances of: **<appRoot>http://localhost:8080/calrsrc</appRoot>**

The ***appRoot*** property in cal.options.xml is the same as the ***root*** property in cal.properties and should be set to exactly the same value. As noted above, it should not be served over https.

browserResourceRoot, HTTPS, and mixed content messages

The browserResourceRoot is where the browser will retrieve css, images, javascript, and other theme files after a transform is performed. If you keep the structure of the theme directories intact, you should not need to change this value. If you choose to serve the resources from a different host, you should change the the browserResourceRoot to include the full url to the resources. If the resources served from a separate host are served over SSL, change the browserResourcesRoot scheme to https as well to avoid mixed content messages (e.g. /caladmin and /ucal).

Directory browsing and pathname discovery

If directory browsing is disallowed on your web server, be certain to set the <directoryBrowsingDisallowed> option to "true" in the *cal.options.xml* file, which will cause the filters to search for a marker file called xsltdir.properties. This allows pathname discovery to continue working. Pathname discovery allows Bedework to pick appropriate locales and browser types based on browser settings or fall back to default themes.

For more detailed information about Bedework theming and pathname discovery, please see Theming Bedework**.**

Setting up URL builder & Webcache

Introduction

Webcache Update as of Bedework version 3.9.0.15
The Bedework web cache, as originally shipped with Bedework versions 3.7, 3.8, and 3.9, can produce file paths which exceed operating system maximum lengths. As of Bedework version 3.9.0.15 the web cache is updated to correct this problem, provide other enhancements, and ships with the Bedework Quickstart.

- To download the latest webcache (and read installation instructions) please visit http://dev.bedework.org/downloads/bw-webcache/ .
- For more information please see http://www.jasig.org/news/new-bedework-webcache-available

Bedework ships with a **production-ready** page caching application for caching public event data feeds. Webcache is used only for data feeds and is fronted by the URL builder, a feed URL and widget generator. Webcache itself fronts Bedework, storing results pages in memory and, if necessary, a file hierarchy.   Result pages may appear in a variety of formats (xml, json, ics, rss, html) and may contain a single event or a list of events.

Using the Webcache is recommend for two reasons.   Like any caching system, it should improve responsiveness.   Also, Webcache should protect your Bedework server from having to work so hard, which may lead to greater reliability. Webcache can be placed on another host to further protect your Bedework server

To illustrate how Webcache works in the quickstart (where it is installed next to the rest of Bedework). Let's say you'd like a json feed of the next 21 days of all public events.

You'd send this URL: http://calendar.myschool.edu/webcachev/1.0/genFeedList/21/list-json/all/all.   (Don't worry about having to actually construct such a URL. That's what the URL Builder is for).

First Page Cache looks at the file location <topOfPageCache>/v1.0/genFeedList/21/list-json/all/all.json. If it finds a page there, it serves it up. Otherwise, it "translates" the URL into http://calendar.myschool.edu/feeder/listEvents.do?skinName=list-json&days=21, calls it, and then both caches the results and returns them to the user.

Note: Prior to Bedework version 3.9.0.15, cache expiration must be managed using a script (or other mechanism) that you create. For more details, see the Bedework wiki: http://www.bedework.org/trac/bedework/wiki/CachedFeeder/ExpiringEvents . As of Bedework 3.9.0.15, cache entries expire after one hour, and the cache is cleared on a server restart. For details of the webcache update in 3.9.0.15 please read the announcement.

Configuring the URL and Widget Builder

The url builder is a web form that generates feed URLs and widgets for use on other web pages. Users specify whether they'd like a feed or a widget (paste-able code), what kind of output they're interested in, and other options.   Users may filter by categories and/or by a group.

The feed URL's may be XML, RSS, HTML, JSON, or ICS.

The widgets are javascript functions that process a JSON feed.

Point URL Builder at Webcache

1. cd <qs>/cachedfeeder/URLandWidgetBuilder/javascript/bedework
2. edit listEvents.js and update calendarServer (line 25)
3. edit builder.js and update urlPrefix (line 21)

Reploy the urlbuilder:

```
./bw -deployurlbuilder
```

The URL Builder will be deployed to <qs>jboss-5.1.0.GA/server/default/ROOT.war/urlbuilder

If you move the Webcache

The webcache application shipped with the quickstart should work as is, out of the box. You should not need to modify the application. If you move the application to another server, you will need to point the webcache at your Bedework server (because the path will no longer be server-relative).

Point Webcache at Bedework

Easier way: Edit the deployed application

1. cd <qs>jboss-5.1.0.GA/server/default/bwdeploy/webcache.war
2. Explode the archive, if necessary, using the jar command. If you replace the archive with the exploded war you can leave it that way. Make sure the directory is called webcache.war
3. cd WEB-INF/config/environments/development.rb
4. Edit the TARGET server
5. (optionally) Jar up the directory and overwrite the old webcache.war archive.

Hard way: Rebuild webcache

This way is more work because you need to install the jruby environment. You'll need to install jruby, warbler, and jruby-openssl. These commands should do the trick on those flavors of linux that have apt-get. sudo jruby -s gem install warbler sudo jruby -s gem install jruby-openssl

```
sudo apt-get install jruby
export PATH=$PATH:$JRUBY_HOME/bin
sudo jruby -s gem install warbler
sudo jruby -s gem install jruby-openssl
```

Then you can rebuild and redeploy with

```
./bw -buildwebcache
./bw -deploywebcache
```

Not planning to use the shipped JBoss?

Bedework is built to take advantage of some features of the jboss j2ee application server and is also built for jboss deployment. We are unable to provide support for deployment on other application servers. As a number of background services* are required for the proper functioning of the system, moving to another container is a complex proposition and not recommended. The Quickstart release, with the included JBoss, is our recommended foundation for a production release.

*Example background services in Bedework:

- The background services include the caldav implicit scheduling service, indexing, dump/restore and so on.
- The JMS service we use is ApacheMQ and some of its special features are employed to implement this system.

Fronting JBoss with Apache

Introduction

Many people prefer to have Apache forward web requests to their Bedework server.   They may do so because they are more comfortable in Apache or because they enjoy its flexibility.   In this scenario, Apache serves out static content (including, often, static Bedework content), fronts some PHP-based tools, such as phpPgAdmin, AND passes Bedework requests to JBoss at port 8080.

There are at least two Apache2 modules that can pass requests to JBoss: mod_jk and mod_proxy.   Both should work. The instructions below assume mod_jk.   They also assume you are running on a Linux server.

Get Apache Ready

1. Install apache2, if not already present, via yum, apt-get, or download
2. Install libapache2-mod-jk (jk_mod.so) via yum, apt-get, or download
3. cd to your apache configuration area, generally /etc/apache2 or /etc/httpd
4. edit httpd.conf to load jk_mod.so

Set up Workers

Create a mod_jk workers file. Our *workers* will be Bedework web clients running at port 8080.   The workers file can be placed anywhere. In this example, we'll create a new directory called misc and create a file there called jk-workers.properties.   You can start with these contents:

```
# workers.properties
# See a default config at  /etc/libapache2-mod-jk/workers.properties

# The list of workers (can be more than one, but we won’t need more here)
worker.list=bedework

# Define the worker
worker.bedework.port=8009
worker.bedework.host=localhost
worker.bedework.type=ajp13

# Specifies the load balance factor when used with
# a load balancing worker.
# Note:
#  ----> lbfactor must be > 0
#  ----> Low lbfactor means less work done by the worker
worker.bedework.lbfactor=1
```

Create an Apache Configuration file for your Bedework Server

You can get started with the following configuration, changing the path to your workers file, the location of your document root, and the server's URL

This configuration assumes that you are using Apache to serve out your static Bedework content. More on this later. It also assumes you are using https: for any Bedework clients that require a login.

```
JkWorkersFile /etc/httpd/misc/jk-workers.properties
#JkLogFile    /var/log/httpd/bedework/jk-mod.log
#JkLogLevel info
#JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"
JkOptions +ForwardURIProxy
#JkOptions +ForwardKeySize
JkOptions -ForwardDirectories
#JkRequestLogFormat "%w %V %T"
#JkShmFile /var/log/httpd/jk.shm

NameVirtualHost *:80
NameVirtualHost *:443

<Virtualhost *:80>
    DocumentRoot /opt/bedework/wwwDocRoot
    ServerName calendar.example.edu
    ErrorLog /var/log/httpd/calendars-error.log
    CustomLog /var/log/httpd/calendars-access.log combined
    Options Indexes

# mount the bedework jboss apps

    JkMount /* bedework

# unmount everything that is being served by Apache directly
# for directories, unmount both the directory itself (dirname) and then all under it
(dirname/*)
    JkUnMount /bwAppRoot bedework
    JkUnMount /bwAppRoot/* bedework
    JkUnMount /bedework-common bedework
    JkUnMount /bedework-common/* bedework
    JkUnMount /tzdata.zip bedework
    JkUnMount /favicon.ico bedework
    JkUnMount /index.html bedework
    JkUnMount /phpPgAdmin bedework
    JkUnMount /phpPgAdmin/* bedework
    JkUnMount /urlbuilder bedework
    JkUnMount /urlbuilder/* bedework

# redirect secure web apps to https

    RedirectMatch  ^/caladmin(.*)$ https://calendar.example.edu/caladmin/
    RedirectMatch  ^/eventsubmit(.*)$ https://calendar.example.edu/eventsubmit/
    RedirectMatch  ^/ucal(.*)$ https://calendar.example.edu/ucal/

 </Virtualhost>

<Virtualhost *:443>

    DocumentRoot /opt/bedework/wwwDocRoot
    ServerName calendar.example.edu
    ErrorLog /var/log/httpd/calendars-error.log
    CustomLog /var/log/httpd/calendars-access.log combined
    Options Indexes

    SSLEngine on

    SSLCertificateFile /etc/httpd/ssl/certs/bedeworkcert.pem
   SSLCertificateKeyFile /etc/httpd/ssl/certs/bedeworkkey.pem

   # mount the bedework jboss apps

   JkMount /* bedework

   # umount everything that is being served by apache directly

    JkUnMount /bwResourcesRoot bedework
    JkUnMount /bwResourcesRoot/* bedework
    JkUnMount /bedework-common bedework
    JkUnMount /bedework-common/* bedework
    JkUnMount /tzdata.zip bedework
    JkUnMount /favicon.ico bedework
    JkUnMount /phpPgAdmin bedework
    JkUnMount /phpPgAdmin/* bedework
    JkUnMount /urlbuilder bedework
    JkUnMount /urlbuilder/* bedework
</Virtualhost>
```

Restart apache2 to test that the configuration is valid

In most cases, /etc/init.d/apache2 restart or /etc/init.d/httpd restart

Copy Static Content to Apache

If you've chosen to have Apache serve your Bedework static content, copy that content from JBoss to the Apache DocumentRoot you defined above.

1. Create a directory called bedework-common in your wwwDocRoot and fill it with the contents of <quickstart>/jboss-5.1.0.GA/server/default/bwdeploy/bwcal.ear/bedework-common.war.
2. Copy these files or directories from <quickstart>/jboss-5.1.0.GA/server/default/deploy/ROOT.war:
   a. urlbuilder
   b. tzdata.zip
   c. favicon.ico (unless you have your own ready to go)
3. Create a directory called bwResourcesRoot in your wwwDocRoot and copy as many of these directories from <quickstart>/jboss-5.1.0.GA/server/default/deploy/ROOT.war as you need:
   a. caladminrsrc (admin client)
   b. ucalrsrc (personal client)
   c. eventsubmitrsrc (submissions client)
   d. calrsrc.MainCampus (public client)

Try it

You should be able to reach your Bedework clients on port 80.

A note on jmx-console

If you'd like to make your JMX Console available, you'll need to edit <quickstart>/jboss-5.1.0.GA/server/default/deploy/jmx-console.war/WEB-inf/jboss-web.xml.   Add a virtual-host ***below*** security-domain. The resulting file should look like this:

```
<!DOCTYPE jboss-web PUBLIC
  "-//JBoss//DTD Web Application 5.0//EN"
  "http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd">

<jboss-web>
    <!-- Uncomment the security-domain to enable security. You will
         need to edit the htmladaptor login configuration to setup the
         login modules used to authentication users.   -->
    <security-domain>java:/jaas/jmx-console</security-domain>
    <virtual-host>calendar.mycollege.edu</virtual-host>
</jboss-web>
```

Public Events Calendaring

Documentation In Progress
This chapter of the Bedework 3.9 Documentation is in the process of being updated.

Introduction

Bedework's public events system allows different units within an organization to publish events to a central pool where they can be disseminated to the largest appropriate audience. Public events can be delivered via

- web interfaces,
- subscriptions in the personal client,
- direct downloads,
- filtered feeds of events (e.g. ical feeds, rss, caldav, json), and
- embeddable event widgets that rely on filtered feeds of events.

External calendar subscriptions can be aggregated with the central event pool.

Bedework is highly configurable. This section provides guidance on setting up your public events framework in Bedework.

Exposing Public Events

There are several ways users can access public events:

Calendar suites

Each calendar suite in your Bedework implementation can produce a custom web site with its own context, and you will need at least one calendar suite to display public events or to produce data feeds.

Downloads

Events and whole calendars can be downloaded in iCalendar format from Bedework's web interfaces.   The downloads are suitable for any desktop calendar client such as MS Outlook and Apple's iCal. They can be imported into other calendar systems as well.

To download an event, look for the download icon on the event details page or in a list view. To download a calendar, visit the “All Topical Areas” section of a public calendar suite and select the icon to download the calendar. Calendars can be downloaded from today into the future, for past dates, or for a date range.

Subscriptions

Subscriptions to Bedework's public events can be pulled for use in Bedework's user client, where there is built in integration for subscribing to public calendars, and from any ical capable calendar client. RSS and javascript feeds are also available (see “Data feeds” below). The public CalDAV server allows access to any public Bedework calendar from CalDAV capable clients.

Data feeds - ical, raw xml, rss, json, js, etc.

Bedework 3.6 ships with a specialized application for serving data feeds: the “feeder” web app found at http://localhost:8080/feeder. The feeder app comes with a number of skins, the default being clean XML.

Bedework has a special action for generating a discreet list of events most useful to data feeds: the listEvents action. If you want to produce feeds from other Bedework actions, see the stylesheet examples in the feeder app's “default” directory.

Data feeds rely on XSL stylesheets to transform Bedework XML into a proper output format, except for the case of iCalendar feeds which are converted by the system directly.

Be aware: data feeds can potentially request large amounts of data which can be further compounded by a high request rate. The core Bedework system should be protected by a front end caching system, and Bedework 3.6 ships with an example cache and URL builder packaged in Tomcat (see: http://localhost:9090/urlbuilder). For more information about the URL builder and web cache, see the section 6.9 in this chapter, “Cached Feed and URL Builder”.

Several caching systems have been recently developed for use with Bedework. Please visit the Bedework webite, wiki, and mailing lists for information about caching if you anticipate using the feed API directly against your system.

Action: listEvents

"listEvents.do" generates a listing of discrete events optionally filtered by categories or creator in a range of time. This listing is useful for RSS, javascript, and ical feeds and for producing the traditional event list such as a printed Academic Calendar or agenda.

It is the view established in the "List" tab of the public and personal clients.

The request returns XML output which can be transformed into RSS, Javascript, HTML, etc. using Bedework's XSLT filter.   The quickstart comes with a number of XSL templates that handle conversion to various outputs. For example, rss-list.xsl and json-list-src.xsl are default transforms for RSS and JSON feeds. Look for these files in the application root for the public client (<quickstart>/bedework/deployment/webpublic/webapp/resources/). Further instructions can be found at the top of the files.

The listEvents action can be used in two ways:

1. Default: return a list of events from today through a specified number of days (e.g. RSS or agenda view); if no duration is supplied, return the next seven days.

2. Return a list of events within a specified date range; if no start date is supplied, begin today. The maximum number of days returned is limited to 31. Requesting a range larger than this limit will return an error.

The action's simplest form looks like:
http://localhost:8080/cal/listEvents.do
and returns the discrete events for the next seven days.

Parameters:

Time parameters:

• days=n (number of days to display forward from "today"; default is seven days.)

or

• start=yyyy-mm-dd (start date)

• end=yyyy-mm-dd (end date - exclusive. *Optional* - if unspecified, the default duration of seven days from the start date will be displayed.)

Calendar path:

• calPath=/public/cals/MainCal

• calPath=/public/cals/SomeOtherCal

• calPath=/public/cals/ (should return events from both MainCal and SomeOtherCal)

The calPath parameter constrains the query. If you are building a link to a data feed, best practice is to **always** send it.

Filters:

• catuid=*categoryid* (filter by a category)

• catuid=*categoryid*&catuid=*othercategoryid* (filter by more than one category)

• cat=*catname* (filter by a category name (less exact))

• cat=*catname*&cat=*othercatname* (filter by more than one category name)

• creator=*creatorid* (filter by creator, e.g. */principals/users/agrp_Somegroup*)

Format:

• format=text/calendar (forces the output into iCalendar format)

If no time parameter is included, the list will display "today" plus seven days. Time and filter parameters can be combined. An arbitrary number of category filters may be added to the query string.

A number of Bedework skins provide a means for arbitrary filtering, and are easily extended. For example, the list-json.xsl skin found in the feeder app allows for the a key/value pair to be sent with a filter name as an application variable. By extending this file you could, for example, send the parameter: "setappvar=filter(location:Jefferson Hall)" and then use this to filter events in the stylesheet.

Examples:

(note: these exclude the calPath parameter. In most cases, you should append the calPath to the query string shown. e.g. "&calPath=/public/cals/MainCal").

/listEvents.do?days=4
returns the next four days' events

/listEvents.do?start=2007-01-01
returns all events from Jan 1, 2007 forward (seven days)

/listEvents.do?start=2007-09-01&end=2007-10-01
returns discreet events from Sept 1, 2007 through Sept 31, 2007 (end date is exclusive)

/listEvents.do?catuid=ff808181-1fd7389e-011f-d7389eff-00000004
returns the next seven days worth of events filtered by the category "Exhibits" (as found in the quickstart). Note: this is the preferred method of selecting by exact category.

/listEvents.do?cat=Ballroom%20Dance
returns the next seven days worth of events filtered by the category "Ballroom Dance"

/listEvents.do?creator=/principals/users/agrp_SomeGroupId
returns the next seven days worth of events created by the group "agrp_SomeGroupId", where the group id is the "event owner" for the group.

/listEvents.do?days=3&catuid=ff808181-1fd7389e-011f-d7389eff-00000004
returns the next three days worth of events filtered by the category "Exhibits" (as found in the quickstart).

http://localhost:8080/cal/main/listEvents.do?start=2009-06-07&format=text/calendar
(http://localhost:8080/cal/main/listEvents.do?start=2009-06-07&format=text/calendar*)
returns events in icalendar format for seven days starting June 7, 2009

Output:

Events are represented as follows:

```
:

:

<page>eventList</page>
<events>
  <event>

  :

  :

  </event>
</events>

:

:
```

To output the data in RSS, append the query string with “&skinName=rss-list”, e.g.

/listEvents.do?days=4&skinName=rss-list

To output the data in json, append the query string with “&skinName=json-list-src”, e.g.

/listEvents.do?days=4&skinName=json-list-src

Starting with the rss-list.xsl or json-list-src.xsl files it is easy to build your own output types and data feeds. For more information on these topics, see the chapter 6 “Bedework Theming” and the Bedework wiki: “Using json Feeds in Static Web Pages”.

http://www.bedework.org/trac/bedework/wiki/BedeworkManual/v3.6/DesignGuide/jsonFeeds

Cached Feed and URL/Widget Builder

The Cached Feed and URL Builder are packaged in the Tomcat server found in the Bedework quickstart and provide a way for users to produce cached data feeds and widgets for public events.

URL/Widget Builder

From the user's perspective it all starts here. The user fills out a web form where she dictates which events she wants (lectures and seminars hosted by the School of Architecture, for example), over what timeframe (the next 21 days, the current month, etc.) and in what form (rss feed, iframe widget, etc.). For those looking for a feed, the form generates a URL that when clicked initiates a download (icalendar) or perhaps an invocation of their feed aggregators (rss). For those looking for a widget, the form writes some paste-able code into a text box.

CachedFeeder

CachedFeeder is written in Ruby and uses Ruby's page cache. Results that Bedework generates are mapped to a unique URL; each URL corresponds to a physical location in the cache's filestore. The filestore is hierarchical and follows the path components in the URL. As you'd expect, when a URL is requested, CachedFeeder first looks for the corresponding cache file. If the file exists, then it returns it. If it doesn't exist, the caching app calls Bedework's feeder application and caches the resulting file in the cache's file system store.

CachedFeeder uses jruby, an implementation of Ruby that generates JVM code, to compile the program into a war file for deployment. The cache appears by default in <tomcat>/webapps/webcache/v1.0. Currently, cache pages aren't expired by the system. See http://www.bedework.org/trac/bedework/wiki/CachedFeeder/ExpiringEvents for some advice on expiring pages.

Feeder Application

The feeder application is a simplified version of the webclient application. It doesn't need nearly as much functionality, since the app only needs to worry about public event feeds.

Further documentation for the Cached Feed and URL Builder is maintained in the Bedework Wiki at http://www.bedework.org/trac/bedework/wiki/CachedFeeder

Adding custom properties (X-Properties) to events

X-Properties are a means of extending ical to define custom fields. They are useful for site-specific data and for extending Bedework for site specific functionality, but be aware that while other calendar clients will preserve them, they will not display them. For example, if a user downloads an event into outlook, your locally defined x-properties will not be seen.

All Bedework X-Properties take the following form:

X-BEDEWORK-*PROPERTYNAME;param1;param2;param3*:value

Bedework X-Property names are declared at the top of bedeworkXProperties.js. If you would like an X-Property to be considered for inclusion in Bedework, we suggest beginning the name with "X-BEDEWORK-".

Local properties can be named appropriately, e.g. X-MYUNIVERSITY-PROPNAME.

Bedework X-Properties

The following table lists two of the x-properties used by Bedework. For more information about Bedework's x-properties, see the Bedework wiki.

X-Property Name	Property Value	Parameters	Description
X-BEDEWORK-SUBMITTEDBY	Identity of event submitter	none	Assembles the userId of the event submitter, the group name, and the group userId
X-BEDEWORK-IMAGE	URL of image resource	X-BEDEWORK-PARAM-WIDTH X-BEDEWORK-PARAM-HEIGHT X-BEDEWORK-PARAM-DESCRIPTION	URL of image to be included with event description in web views

Adding Custom X-Properties

Add custom X-Property handling to the setBedeworkXProperties() function of bedeworkEventForm.js. This function takes the event form object as a parameter; you can convert a custom form field to an x-property using the update() method of the x-property javascript object (declared in bedeworkXProperties.js in admin, submission, and user clients):

```
bwXProps.update(name,params,value,isUnique);
```

The update method takes the following parameters:

- name - name of the x-property, e.g. X-MYUNIVERSITY-GPSCOORDS
- params - series of key value pairs expressed in a 2-D array
- value - value of the x-property
- isUnique - true or false; if true, Bedework will only allow one x-property with this name.

Example:

```
function setBedeworkXProperties(formObj,submitter) {

  // set up specific Bedework X-Properties on event form submission
  // Depends on bedeworkXProperties.js
  // Set application local x-properties here.
  // X-BEDEWORK-IMAGE and its parameters:
  if (formObj["xBwImageHolder"] && formObj["xBwImageHolder"].value != ''){
    bwXProps.update(bwXPropertyImage,
                  [[bwXParamDescription,'An Orca!'],
                   [bwXParamWidth,'400'],
                   [bwXParamHeight,'300']],
                   formObj["xBwImageHolder"].value,true);
  }
  // X-BEDEWORK-SUBMITTEDBY
  bwXProps.update(bwXPropertySubmittedBy,[],submitter,true);

  // commit all xproperties back to the form
  bwXProps.generate(formObj);
}
```

Note: in this example, the image parameters are hard coded. The value of formObj["xBwImageHolder"] will be a url reference to an image.

X-Property Output

X-Properties are output within the event XML like so:

```
<event>

:

:

  <xproperties>
     <X-BEDEWORK-SUBMITTEDBY>
        <values>
           <text>caladmin for Communications (agrp_admGrp2)</text>
        </values>
     </X-BEDEWORK-SUBMITTEDBY>
     <X-BEDEWORK-IMAGE>
        <parameters>
          <X-BEDEWORK-PARAM-DESCRIPTION>An Orca!</X-BEDEWORK-PARAM-DESCRIPTION>
          <X-BEDEWORK-PARAM-WIDTH>400</X-BEDEWORK-PARAM-WIDTH>
          <X-BEDEWORK-PARAM-HEIGHT>300</X-BEDEWORK-PARAM-HEIGHT>
        </parameters>
        <values>
          <text>http://photography.naturestocklibrary.com/orca-stock-photo.jpg</text>
        </values>
     </X-BEDEWORK-IMAGE>
  </xproperties>


:

:

</event>
```

Note: if no parameters are present, they will not be output

Entering Public Events

Introduction

There are four ways that events may be added to your public events calendar(s):

1. They may be added directly by calendar administrators via the administration web client.
2. They may be imported by superusers via the administration client.
3. They may arrive from an external source by way of a subscription. Subscriptions may be set up by a calendar administrator in the administration client.
4. They may be suggested by authenticated members of your user community via the submissions web client, then published by calendar administrators via the administration client.

Log into the Administrative Web Client

The administrative web client is the control center for managing public events. When logged in you are presented with the main menu tab view that will look like this:

<graphic>

for a typical event administrator, and like this:

<graphic>

for a super user.

Add a non-recurring event

1. Select the “Add Event” link under the “main menu” tab.
2. Type in a title for the event.
3. If you are working with a single public calendar for publishing (the default) you will not be presented with a calendar to select. However, if you are working with more than a single publishing calendar or are logged in as a superuser, you must select a calendar. Typically this would be the default ***cals/mainCal***.
4. Select ***all day*** if this is an all day event.
5. (Optionally) select ***floating*** if you don’t want to be restricted by time zone or
6. If you haven't selected all day:
    a. select a start date
       <graphic>
    b. select a start time
    c. select an end date for the event
    d. select an end time or duration, or select ***this event has no duration or end date.***
7. Under recurrence leave at ***this event does not recur***. See the next section for recurring events.
8. Leave the event status as ***confirmed***, or if the event is likely to be updated, select ***tentative***. If an event has been canceled, select ***canc eled***; this will display a cancellation notice in the event title in calendar suites and data feeds.
9. Enter more details about the event in the description. *You must add something here*.
10. (Optionally) enter a cost for the event.
11. (Optionally) enter a URL for the event, if you have a web page with more information about it.
12. (Optionally) enter a URL for an image (jpeg, for example) to be displayed with the event details.
13. Select a location for the event. For a more comprehensive list of locations, click on ***all***. ***preferred*** lists locations you've already used.
14. Select a contact for the event. For a more comprehensive list of contacts, click on ***all***. ***preferred*** lists contacts you've already used.
15. Select the topical area(s) with which you want to tag the events
    <graphic>
16. Select the “add event” button

Add a recurring event

1. Follow steps 1 through 6 above
2. Select ***event recurs***
3. Select how often the event recurs under frequency. For example if the event recurs every month, select ***monthly***.
4. Leave at ***forever*** -or-
   Select ***times*** and type in a number to specify how many times you want the event to recur -or-
   Select ***until*** and add an end date. *Note: the end date you specify here is different from the end date you specified earlier for the event. The end date here is the date you want the recurrence to stop and not the end date for the event.*
5. For more advanced options, click ***show advanced rules***. You might need to re-select ***monthly***(or frequency of your choice) to see the advanced rules.
    a. If you want the event recurring every other month type in 2 under the ***interval*** label or any interval of your choice.
    b. You can specify how you would like the event to recur. For example, if you want the event to recur on the first Monday and Tuesday of the month and also on the last Monday and Tuesday,
       <graphic>
       select as shown in the diagram.
    c. You can also create exceptions for your recurring events. For example, if there is a public holiday on a Monday you have designated your event to recur. Create an exception by deleting the recurrence instance of the event after it is created.
    d. You can also create one-off recurrences for the event. Select a date under the “recurrence and exception dates” field and select the “add recurrence” button.
6. Follow steps 8 through 16 above to add the event.

To Edit/delete an event

1. Select ***manage events*** under main menu. Alternatively you can search for the event by typing the event name under ***event search*** in the main menu.
2. You can show all events or active events or you can filter events by selecting a category.
3. To edit an event, select the event under the ***Title*** column or, if the event recurs, click on ***master*** under the ***Description*** column. Changes you make to the master affect all instances of the event. To edit or delete a single instance of an event, click on ***instance*** under ***Descrip tion***.
4. When you are done with your changes click on ***update event***.

Importing events

Superusers can import events in the adminitration client by clicking on the ***System*** tab, then clicking ***Upload ical file***. The currently active group will own the events. You may only import events into true calendars (not calendar aliases), e.g. public/cals/mainCal. Because of this, it is a good idea to set categories on the event in the ical file prior to import. We will allow import of events with topical area tagging in upcoming releases.

Subscribing to an external source of events

You may also wish to consider subscribing to the ical file instead of importing its events.

Suggesting events using the Submissions Client

The submissions web client allows authenticated members of your community to suggest public events. Using the quickstart, you can log into the client from the Bedework quickstart jump page: http://localhost:8080/bedework/

Submitted events do not automatically go into the public calendar. They are placed in a "moderation" queue, visible from the administration client under the ***Pending Events*** tab. Event administrators can publish, delete, or update the event. If the event is published, it is displayed in the public calendar for the administrator's calendar suite.

Adding an event through the submissions is very similar to adding an event in the administration client. Most of the fields are the same. The main differences are that the fill-in form is broken into a few screens -- with ***next*** and ***previous*** links added, the submitter's email address is requested, and a text box is provided for any additional instructions. The calendar administrator who is reviewing the suggested event might email the submitter for clarification.   Another difference is that there is no place to add recurrence rules. The submitter can cover those in the instructions text box.

Processing submitted events in the Administration Client

To administer a submitted event

1. Log into the administration client
2. Select the ***Pending Events*** tab
3. Select the event title link under the ***title*** column

Under ***Event Information*** you can see details about the event. If the user who submitted the event has suggested a location for the event, you can create the new location under ***Add Location*** in the ***Main Menu*** tab. You can hide the event comments by clicking ***show/hide***. You can also edit the other fields on the form. For examle, you might click on a topical area.

- To claim the event, click on ***Claim Event***. A claimed event becomes the property of the currently active group. Only members of that group (and superusers) will be able to update, delete, or publish that event.
- To publish the event, click on ***Publish Event***. If your set up includes more than one calendar or your are logged in as a super-user, you will need to select a calendar to publish the event. If you follow the single calendar model, this would typically be the “cals/MainCal”. Once an event is published, it will appear on the public calendar.
- You may also ***Update*** the contents of the event while leaving it in the queue or ***Delete*** it.

Configuring Your Public Events System

Overview

This section leads you through the steps of setting up your public events system.

Plan your public events system

Start with a sketch

A good way to start is to sketch out a hierarchy of calendars that work in your organization. Start modestly, with a sparse tree of calendars that you'll likely end up using. It's much easier to add than it is to move calendars (and their attendant objects -- categories, topical areas, and subscriptions) around later.   By the time you've instantiated your modest tree, you'll have the know-how needed to configure Bedework in the shape of your organization.

If you loaded the recommended data set when you initialized Bedework, you already have a hierarchy that can serve as a starting point or as an example.   Click on ***View All Calendars*** in the public client (*yourserver*:8080:/cal) to take a look at the shipped hierarchy.

There are many kinds of calendars (event collections) that you might want to have. It is likely that at least some departments or offices in your organization will want their own calendars.   It is also possible that you'll want to have calendars that group like events together. For example, you might want holidays collected on a calendar. You also might want to have social events on a separate calendar.   You'll also want to consider grouping calenders into folders.

There are two primary considerations in deciding how to group calendars together.

1. Since in the usual configuration, everything in a folder will be tagged with a common tag, would anybody be interested in the whole collection? For example at a university, a folder called ***School of Science*** with members that include ***Biology*** and ***Chemistry*** probably works better than a folder called ***Departments*** with members that include ***Anthropology***, ***Engineering***, ***Biology*** and ***Medicine***; while users might be interested in looking at which events occur within the School of Science, they are unlikely to be interested in events that take place in *any* department because it isn't a particularly useful way to filter events.

2. This will be become clearer later, but it is often useful to tie a ***View*** to a folder. You can see the views along the left-hand side in the public client. The considerations are much the same.

In general, there are two common ways to group calendars: By organizational structure (example: *School of Engineering* folder with *Civil Engineering* and *Mechanical Engineering* as members) and by function (example: *Lectures, Seminars, and Conferences* folder with *Lectures, Seminars,* and *Conferences* as members). Many organizations use both kinds of groupings. The advantage to the "both ways" approach is that events can be routinely tagged twice: one tag to indicate *who* the event is associated with and one to indicate *what kind* of event it is. Users (and web sites than consume events) can then easily filter events by *who* or by *what kind*.

Bedework Terms:

- **Calendar suite**: a web application with its own context, theme, subscriptions, and views. Calendar suites are attached to Bedework's group hierarchy, and event administrators work within the context of the calendar suite under which their group is placed.   Calendar suites control how events are tagged and filtered.
- **Calendar collection (Calendar)**: a container for events (or other calendaring items such as tasks and journal entries). When we use the term "Calendar" we mean calendar collection.
- **Folder**: a container for calendar collections (only)
- **Category**: an iCalendar (RFC-2445) property used to tag events. Categories are maintained by superusers and calendar suite owners
- **Subscription**: an alias to a calendar collection. Subscriptions may point to calendar collections, to other subscriptions in Bedework, or to external icalendar feeds. Subscriptions provide structure to Bedework's global calendar tree and to calendar suites. Likewise, they provide a structure and hierarchy for tagging events by category.
- **Topical Area**: a topical area is a subscription that will appear in a calendar suite's add/edit event form and provides fine-grained control over how events are tagged.
- **Filter**: an expression used to match events from the global pool. Bedework makes particular use of category filtering in public events calendaring
- **Event administrator**: a user who is a member of a public events administrative group. Event administrators can add and edit events using the administrative web client.
- **Calendar suite owner**: a user who is a member of public events administrative group to which a calendar suite is attached. Calendar suite owners can modify calendar suite preferences, subscriptions, and views. Calendar suite owners, like superusers, can also add and edit categories.

Create a category for each tree node

Introduction

Because categories play such an important role in public event retrieval, only superusers and calendar suite owners are allowed to manage them. (In future releases, we expect to allow admin groups to create their own categories with which they can filter events – but this is not available in Bedework 3.9).

The URL Builder takes advantage of a naming convention for categories: categories used to represent organizations should be prefixed by “org/”; categories used to represent special system categories (and can be hidden from the URL Builder) should be prefixed by “sys/”. See the “sys/ongoing” category and the “org/Engineering” categories in the Bedework quickstart initial data for examples.

Create Categories

There are many ways to approach the task of setting up all the Bedework structures and definitions you'll need, but if you have a calendar plan on paper, then creating categories may be best place to start.

Create a category for each member of your tree, folders and calendars (aliases) alike.

Create your calendar hierarchy

Introduction

Under the hood, Bedework uses a single calendar collection to store all public events. This model provides flexibility and power for Bedework deployers while simplifying the user interface for event administrators.

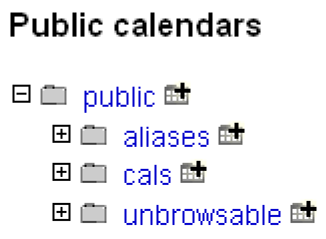
Filtered output: In the single calendar model, categories are the primary means of organizing events. Calendar suites can subscribe to the single calendar collection with a category filter or to predefined calendar aliases in the global calendar tree with filters already defined. This approach provides fine-grained control over what events are returned to the views. As chains of subscriptions get created, events are filtered by the union of all category filters on output.

Tagged input: Subscriptions in a calendar suite may be marked as "Topical Areas" and assigned categories. Topical Areas appear in the add event form for the calendar suite and provide a means for event administrators to tag events by one or many categories. A subscription that is not marked as a topical area won't appear in the add event form for tagging and is, in essence, read-only; a subscription to an external holiday feed is a good example.

Event administrators add events within the context of a calendar suite and tag events with one or more Topical Areas defined for that suite. Each topical area may apply one or many categories to an event. This approach simplifies the event administrator's task substantially: he or she doesn't need to know what categories to select to make an event appear in a particular view or data feed; nor does a user need to know anything about the underlying calendar structure.

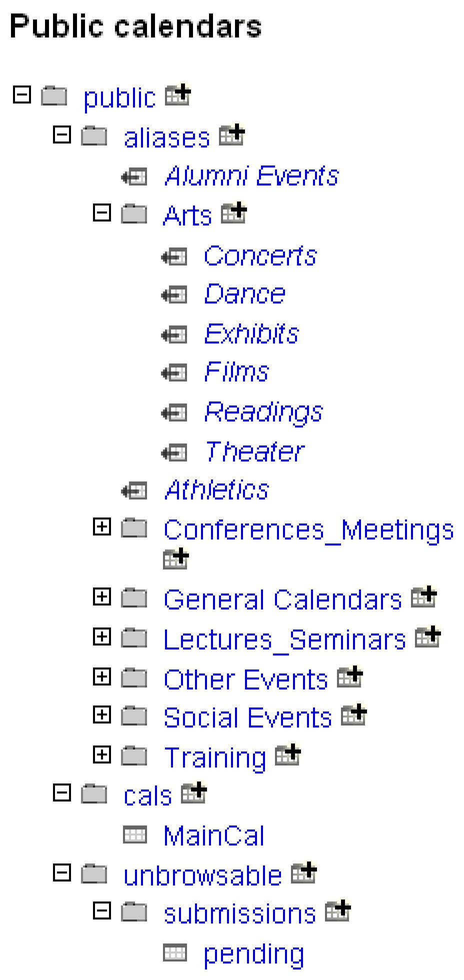
To summarize: subscriptions marked as topical areas are used to tag events on input, and subscriptions filter events on output based on the category filters applied to them. When a subscription points to an alias in the public tree, events are filtered by the union of all filters in the subscription and the public tree alias. Likewise, events are tagged by a union of all categories in the subscription and public tree alias. It is possible to create tagging and filtering chains by creating a chain of subscriptions.

The top-level structure of the /public calendar tree, as shipped with the Bedework quickstart, is divided into calendars, aliases, and an unbrowsable branch. In /public/cals we have only one calendar collection:



All events added to Bedework using the public events administration client are placed within the MainCal calendar collection.

Opening up the tree, we find calendar aliases that can be used by calendar suites for building subscriptions. The calendar aliases are centrally curated, pre-filtered subscriptions. The /public/aliases branch is also used to generate the listing of "suggested topical areas" for the public submissions web client.   Event administrators refine these when they pick up the event for publication. The branch is likewise exposed to personal calendar users as a subscription source within the personal client.



Public calendars

- public
  - aliases
    - *Alumni Events*
    - Arts
      - *Concerts*
      - *Dance*
      - *Exhibits*
      - *Films*
      - *Readings*
      - *Theater*
    - *Athletics*
    - Conferences_Meetings
    - General Calendars
    - Lectures_Seminars
    - Other Events
    - Social Events
    - Training
  - cals
    - MainCal
  - unbrowsable
    - submissions
      - pending

Calendar aliases / subscriptions appear in CalDAV clients as if they were calendars.

The unbrowsable branch of the tree contains the calendar collection used to hold pending events from the submissions web client and can be used for other special purpose calendar collections.

Add folders and alias subscriptions

1. Log into the Administrative Client with a superuser account.
2. Click on the ***System*** tab, then ***Manage calendars & folders***.
3. Click on ***Folders***
4. Open the hierarchy to reveal *aliases* under public. Anything created there will be available to all calendar suites for subscription.
5. Clicking on the calendar-plus icon, create a folder or a subscription (an alias).
6. If creating a folder, you'll typically choose the corresponding category under ***Categories*** (auto-tagging on input) and leave the Filter alone and the ***Current Access*** as is.
7. If creating an alias, you'll typically choose the corresponding category under both ***Categories*** and ***Filter*** (auto-tagging on both input and output). For the ***URL to calendar***, copy and paste **bwcal:///public/cals/MainCal** from the Note just above the Add button. Leave ***ID*** and ***Password*** blank.

Add CalSuite Subscriptions and Views

Introduction

Up until now, we've paid little attention to ***Calendar suites***, an important abstraction in Bedework. Calendar suites are full-blown Bedework public events web sites *that share a single database of events* (and administrative and submissions clients). Each suite has an independent look and feel (skin) and an independent set of managers. Each suite also filters events in its own way.   For example, you might have a calendar suite

Bedework ships with a default calendar suite called **MainCampus** (and a second suite called SoEDepartmental which serves as a model for additional suites)**.** Some sites will never need to add others. Others choose to develop many, perhaps dozens. More on that later. For now, we'll assume you have one: MainCampus.

Earlier you created categories and used them to set up your hierarchy of folders and calendars (aliases which filter on categories). While the hierarchy you created is visible in the public calendar in certain places (View All Calendars and Advanced Search), *much of the work you've done up to now isn't available to your calendar users or to your calendar managers*. Why? The public client operates in the context of the MainCampus calendar suite and the MainCampus calendar suite isn't aware of the additions. In other words, *you must explicitly select those parts of the hierarchy you want to reveal in the public client* by adding them to MainCampus. You do this in the administrative client by working in the context of MainCampus (which is the default) and creating ***calendar suite subscriptions*** and ***views*** under the Calendar Suites tab.

Add Calendar Suite Subscriptions

Subscriptions define what events are pulled into the suite to be viewed. When assigned as a topical area, subscriptions are used to tag events with one or many categories.

To get you started, **create a subscription for any new top-level alias or folder you created in your hierarchy** by:

1. In the admin client, log in with a superuser account. Make sure that your calendar suite is set to MainCampus (look towards the upper left).
2. Click on the **Calendar Suite** tab
3. Click on **Manage Subscriptions**
4. Open up MainCampus, by clicking on the plus on the far left, next to the folder icon. You'll see all the aliases that are already defined and that they correspond to the top level of the calendar hierarchy that shipped in the quickstart.
5. Click on the add icon to the right of MainCampus
6. In Add Subscription form, fill in the **System Name** and the **Display Name**. Set them both to whatever they will reference, unless the name you've chosen contains certain special characters. For example, if you'd like to display "Arts & Entertainment", set the System Name to "Arts and Entertainment" and the Display Name to something like "Arts and Entertainment".

   > These characters aren't permitted in System Names: & / \ ' " (ampersand, backward slash, forward slash, single quote, double quote).

7. Optionally add a **Description**.
8. Select **Public Alias** as the type.
9. Click on the **Select a public calendar or folder** option. This will reveal the hierarchy you set up.
10. Choose the corresponding top-level folder or alias.

Add Calendar Suite Views and Update the All View

Subscriptions are aggregated into views for display in the public web client. In the Bedework theme, the views are presented along the left hand side of the page. By default, users start with the **All** view selected. **All** generally includes most if not all of the calendar suite subscriptions. The other views usually contain one or two subscriptions, depending on how you choose to aggregate your calendar suite subscriptions.

To get you started, **add a view for each subscription you created in the previous step and add that subscription to the *All* view**:

1. In the admin client, log in with a superuser account. Make sure that your calendar suite is set to MainCampus (look towards the upper left).
2. Click on the **Calendar Suite** tab
3. Click on **Manage Views**
4. In **Add a new view**, type in the name that corresponds to the subscription and click on the **add view** button.
5. In the **Update View** form, Find your subscription in the list of *Available subscriptions* and click on the arrow to slide it over to the *Active subscriptions* column.
6. Click on the **Return to Views Listing** button.
7. Click on the **All** view and repeat step 5.

Set Calendar Suite Preferences

In the Administrative Client, under the Calendar Suite tab, you'll find Manage Preferences. Click there and examine the options. They are:

1. **Preferred view**. You can change if from **All** to one of your other views.
2. **Preferred view period**. Busy sites often change this to day to keep the list of events from growing too long.
3. **Default Categories**. You can hide unused categories here.
4. **Preferred time type**. AM/PM or 24 hour time.
5. **Preferred end date/time type**: This controls the default behavior when creating events. After the start time is entered, either an event duration field or an time/date end time field is presented to the user. The user can also switch to the non-default behavior at event

creation time.

6. **Default timezone**: Set your time zone here.

Add admin groups and users

Introduction

Administrative groups are hierarchical and inherit access rules down the group tree. In the public events system, all admin groups are children of a default root group named ***campusAdminGroups*** and inheritance of access control rights starts from there.

Calendar suites are attached to groups. An event administrator will add and edit events in the context of the calendar suite that is found first when traversing back up the tree.

Public events are administered centrally through the administrative web client. *Every administrator is a member of a group, and events are owned not by administrative users directly, but by a special “events owner” associated with the group*. Each group also has a group owner that is a specific administrative user; this role does not currently provide any functionality, but allows you to specify the lead event admin for a group.

The *events owner* is a Bedework system user, not found in the organization's user space, that owns all the events for a group. Having such an owner allows all group members to see and edit events within the group. Also, the event owner is the “user” whose preferences define the behavior of a calendar suite. The system ensures that the *events owners* are distinct from real users by prefixing them with a string, by default ***agr p_***.

Within the administrative client, events are filtered by the current *events owner* so that administrators can only see and edit events for their current group. A super user can switch to any group. By searching for events in the admin client, event administrators can also tag events created by other groups.

If the system is configured to do so in the configuration properties files, locations, contacts and categories can be filtered to make them editable only by the group. They are however, always readable. Creating contacts and locations that cannot be edited by typical admins can be achieved by creating them in a special group.

Group structure and access control

Access control is inherited from the top down the group tree. Therefore, it is best to create a single, top-level group for access to /public and then add all other administrative groups to it. By default, Bedework comes with a top-level group named “campusAdminGroups”. All other groups should be made members of this group to inherit write-content access on /public. Groups should represent the departments within your organization who will be responsible for adding and maintaining public events, e.g. “Arts”, “SOE” (school of engineering), or “Athletics”.

The first children of campusAdminGroups should be groups associated with calendar suites. By convention, we name these groups calsuite-*Som eName* to distinguish them from typical admin groups.

While it is possible to close branches of the /public calendar tree from access to all administrators by creating a different group hierarchy, we discourage doing this. If public calendars are kept topical, an administrator from any group can add events to any calendar in the tree. However, an administrator can only see events created by his or her group.

It is important to remember that group calendaring (e.g. scheduling meetings within a department) should be kept away from the /public tree – which is only intended for public events. Events that must not be seen by any but a subset of your community are not public (such events are to be distinguished from those intended for a subset of your community that are still okay for others to see). These should be kept in the personal and group space, or if your need dictates it, in a top-level /dept branch of the tree which you will need to create. We believe in actual practice, most group calendaring will best be managed within the personal and group space.

Administrative groups are stored in the calendar database. Administrative groups are intended to be separate from user groups to allow different access rights to be defined for administrative users. (See “Access rights and groups” below)

Add administrative groups

1. In the Administrative Client, choose the ***Users*** tab, then click on ***Manage admin groups***.
2. Click on the ***Add a new group*** button
3. Fill in ***Name, Description, Group owne***r (typically */principals/users/admin*), and set the ***Events Owner*** to */principals/users/agrp_groupN ame* where *groupName* is the same string you used in the ***Name***.

> The prefix ***agrp_*** should be left on event owners; the prefix is defined in your configuration, in the cal.properties file. Look for org.bedework.app.CalAdmin.admingroupsidprefix.

Add calendar administrators to administrative groups

To add an event administrator to the system, simply add the user to a group. When a user is removed from all groups, the user will be removed from the system. Because the public event space is distinct from the personal calendaring space, administrative users are managed in Bedework's database by default (though they need not be)

1. In the Administrative Client, choose the ***Users*** tab, then click on ***Manage admin groups***.
2. On the line that begins ***calsuite-MainCampus,*** click on ***membership***.
3. Add superusers, one at a time, in the ***Add member:*** box, making sure that user is selected to the right, then click ***Add***.

Add superusers

1. In the Administrative Client, choose the ***System*** Tab, then click on ***Manage system preferences***
2. Add users to the ***Super Users*** field. The field accepts a list of comma separated user ids.
3. In most cases, you'll want to add your superuser to the group from which your calendar suite is hung, in this case ***calsuite-MainCampus***. Choose the ***Users*** tab, then click on ***Manage admin groups***.
4. On the line that begins ***calsuite-MainCampus,*** click on ***membership***.
5. Add superusers, one at a time, in the ***Add member:*** box, making sure that user is selected to the right, then click ***Add***.

Theming the Public Events Web Clients

Theming the Public Events Web Clients

Each calendar suite in Bedework's public events system can have its own look and feel. Changes to a public theme can be as simple as altering a settings file and changing some CSS or as complex as creating a completely new layout for events. Please see Theming Bedework for instructions on how to theme Bedework.

Writing custom outputs for data feeds, such as RSS, is also accomplished in Bedework's theming system.

Add and Configure Additional Calendar Suites

Introduction

The quickstart includes the Calendar Suite *MainCal*. You can think of a Calendar Suite as a window through which you manage and view your public events.   For many sites, the MainCal Calendar Suite is sufficient. It's a full-sized window through which all public events are managed and viewed.   There are three likely motivations for adding another Calendar Suite:

1. You'd like to provide a smaller window for managing certain events.   This is usually done to restrict access to certain event tags (topical areas) to specific administrators. As you'll see below, you can set up a Calendar Suite for this purpose by using the administrative client.
2. You'd like to provide a smaller window for managing and viewing certain events, and/or you'd like to create a second web site with an independent look and feel. This is usually done to create an independent departmental instance of a fully-functioning Bedework calendar site. As you'll see below, creating a fully-independent departmental site involves configuration work in the administrative client *and* makin g configuration changes to the Bedework system itself. The Bedework system configuration changes involve defining a new web client to be built and deployed.

Add a New Calendar Suite

A Calendar Suite that is to have its own web context (e.g. http://localhost:8080/mydept) must be built, and the resulting .war will be packaged up and deployed alongside the other suites.   If you wish to use calendar suites solely for limiting access to topical areas, you do not need to build it and may skip directly to **Configure a Bedework Calendar Suite**.

Build a Bedework Calendar Suite

1. Follow the instructions for setting up your build environment.
2. Add calendar suites to ***cal.properties*** and ***cal.options.xml***, using the quickstart's example of *SoEDept* (the demo departmental public web client). Also add the suite to the list of applications to be built in the cal.properties file for the property: ***org.bedework.install.app.na mes***.
3. Copy a template directory for use with the new calendar suite from an existing one, and name it with the name given the calendar suite. E.g. copy bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus as ....demoskins/MyDept
4. Build Bedework

Configure a Bedework Calendar Suite

You must now define the Calendar Suite in the admin client and associate it with an administrative group.

1. Create an admin group from which to hang the Calendar Suite:

    a. Log into the Bedework Admin Client as a superuser.
    b. Select the *Users* tab.
    c. Select *Manage admin groups*
    d. Click the button *Add a new group*
    e. Set the group's properties:
        i. Name: calsuite-MyDept *(note: it is recommended practice to name Calendar Suite groups in this way, prepending them with "calsuite-" or a similar signifier so that they are easy to distinguish from other groups. Furthermore, you should only add users to this topmost group who should be granted Calendar Suite Owner access. Such users can manipulate the calendar suite itself.)*
        ii. Description: provide a reasonable description for the group, e.g. "calendar suite for MyDept"
        iii. Group owner: a userid, probably of the main contact point for the group; this field is only informational
        iv. Events owner: agrp_CalSuite-MyDept *(note: it is recommended practice to make the events owner the same as the Name prepended with "agrp*" or a similar signifier to associate the owner with the group. The "events owner" is a system user: the "agrp_" prefix distinguishes this user from a real user - one associated with an actual person. The calendar suite's preferences will be attached to this user.)

2. Add the new group to “campusAdminGroups” so that proper access rights are inherited down the group hierarchy.
    a. Select the *Users* tab.
    b. Select *Manage admin groups*
    c. Select *membership* for ***campusAdminGroups*** and add the new group to it.
3. Add the calendar suite:
    a. Select the *System* tab.
    b. Select *Manage Calendar Suites*
    c. Click the button *Add Calendar Suite*
4. Set the Calendar Suite's properties:
    a. Name: this is the calendar suite name you defined in myconfig.properties, e.g. if you defined org.bedework.app.MyDept.cal.suite=MyDept, use "MyDept".
    b. Group: this is the name of the group you defined in step 1, e.g. "CalSuite-MyDept"
    c. Root calendar: this will almost always be "/public", the root of the public calendar tree

Enable image uploads

Bedework has long allowed calendar managers to "attach" an image to an event by supplying an image URL.   Now events administrators may optionally upload an image to "attach" it to an event. During image upload, thumbnail images are automatically generated. Thumbnails are used in the "Upcoming" events listing. The full image is used in the detailed event view.

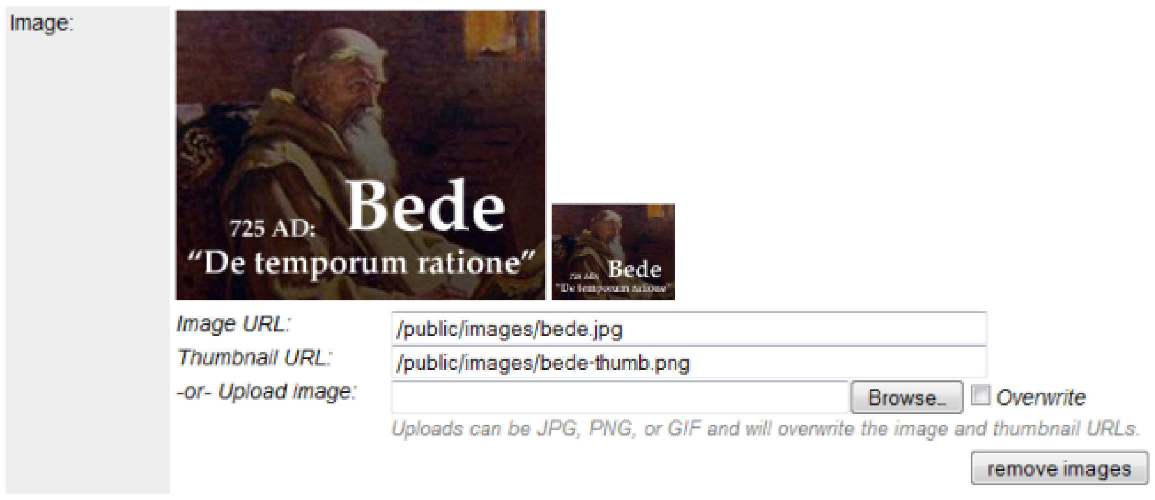


Image URL: /public/images/bede.jpg
Thumbnail URL: /public/images/bede-thumb.png
-or- Upload image:
Uploads can be JPG, PNG, or GIF and will overwrite the image and thumbnail URLs.

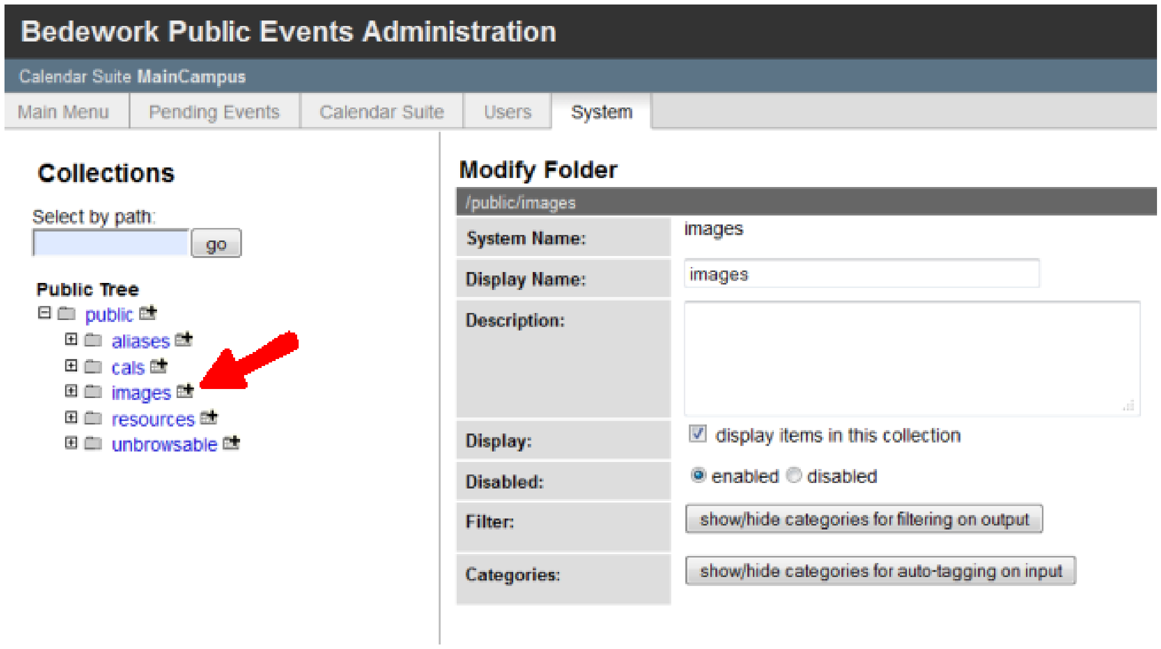
Enabling Image Uploads

Image uploads are enabled by default in Bedework 3.9. If you wish to enable this feature in Bedework 3.8 or you need to enable this feature after upgrading to Bedework 3.9 from a previous version, please follow these steps:

1. Log into the public events administrative webclient (e.g. http://localhost:8080/caladmin)

2. Create an image repository (folder) inside Bedework
    a. Open the admin client.
    b. Visit the **Systems** tab.

c. Click on **Manage Calendars and Folders.**
d. Create a *folder* under /public. Let's call it *images*.



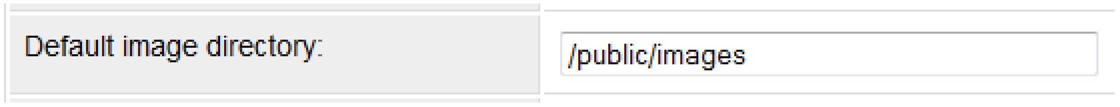
Modify Folder

/public/images

System Name:	images
Display Name:	images
Description:	
Display:	✓ display items in this collection
Disabled:	[✓ enabled] disabled
Filter:	show/hide categories for filtering on output
Categories:	show/hide categories for auto-tagging on input

Note that this directory can be named anything you like - and you can use a different directory for different calendar suites, should you choose to do so.

3. In the calendar suite preferences, add the path to that folder
   a. Visit the **Calendar Suite** tab.
   b. Click on **Manage preferences.**
   c. Fill in **Default image directory** with, in our example, */public/images.*
   d. Save your changes



Default image directory:	/public/images

If you'd like to change the maximum size of an uploaded image, you can change the system-wide setting under Systems Preferences: Look for **M aximum size of a user entity**. It's in bytes. You can also change it on a per calendar suite basis. Look for the Calendar Suite preference **Maxim um size for file uploads.**
A value of zero indicates no limit.

Disabling Image Uploads

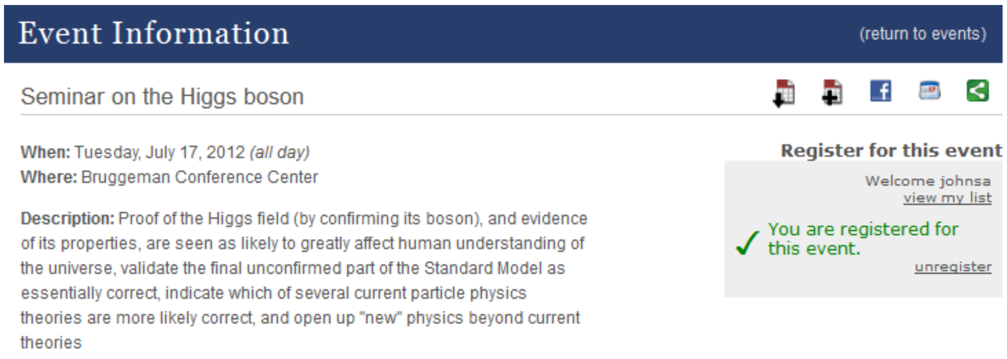
If you do not want to allow image uploads within a calendar suite, remove the "Default image directory" value in the calendar suite preferences:

1. Log into the public events administrative webclient (e.g. http://localhost:8080/caladmin)
2. Visit the **Calendar Suite** tab.
3. Click on **Manage preferences.**
4. Remove the **Default image directory** value
5. Save your changes

Public Events Registration

Bedework Public Events Registration System

Bedework 3.9 supports a basic public events registration system that allows authenticated users to register for events. Users may view and modify their registrations, such as unregistering or changing the number of tickets they've requested. When registration is full, users may choose to be placed on a waiting list. Users on waiting lists will automatically be moved up in the queue if space becomes available.


Event Information (return to events)

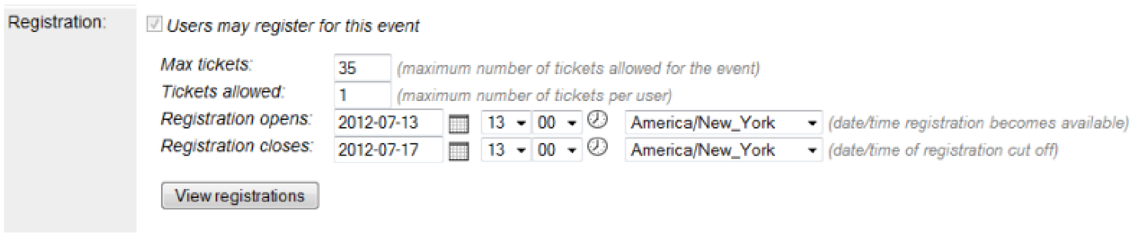
Seminar on the Higgs boson

When: Tuesday, July 17, 2012 *(all day)*
Where: Bruggeman Conference Center

Description: Proof of the Higgs field (by confirming its boson), and evidence of its properties, are seen as likely to greatly affect human understanding of the universe, validate the final unconfirmed part of the Standard Model as essentially correct, indicate which of several current particle physics theories are more likely correct, and open up "new" physics beyond current theories

Register for this event
Welcome johnsa
view my list
✓ You are registered for this event.
unregister

Administrators can specify how many users may register, how many tickets each registrant may request, and set the opening and closing dates of registration. Administrators can view and modify a registration list and download CSV files of their registrations on-demand.


Registration: ✓ *Users may register for this event*

Max tickets:	35	*(maximum number of tickets allowed for the event)*
Tickets allowed:	1	*(maximum number of tickets per user)*
Registration opens:	2012-07-13 13 00 America/New_York	*(date/time registration becomes available)*
Registration closes:	2012-07-17 13 00 America/New_York	*(date/time of registration cut off)*

View registrations

Enabling and Disabling Public Events Registration

Enabling Public Events Registration in Bedework 3.9

The public events registration system **is enabled by default** if you have installed Bedework 3.9 with the data available in the quickstart.

Disabling Public Events Registration

If you wish to disable the public events registration system remove the "Eventreg admin token" from the System Preferences in the administrative web client:

- Log into the Bedework public events administrative client, e.g. http://localhost:8080/caladmin
- Go to the System tab and select "Manage System Preferences"
- Remove the value of the "Eventreg admin token" field (set it to an empty value)
- Click "Update" button to save your changes.

Enabling Public Events Registration after upgrade to 3.9 from a previous release

If you have moved to Bedework 3.9 using data from a previous release, you are not likely to have the data required for event registration in your system yet. Follow these steps to turn on the event registration system: (Please note: the process outlined below is only set up for the default quickstart and postgresql configs at the moment.)

- Start up jboss and apacheDS

- Log into the JMX-Console shipped with Bedework's JBoss, e.g. http://localhost:8080/jmx-console
  - Click "org.bedework" in the left menu, then "service=Eventreg" in the right menu
    - If no database exists:
      - Set “create” and “export” attributes to "True"
      - Click "Apply changes" button (at the bottom of the list of attribute values)
      - Find the "schema" operation in the lower list, and click the “Invoke” button to export schema and create database
      - You should see a successful result; click "Back to MBean" button to return to "service=Eventreg"
    - Point at needed systems:
      - Set the WsdlUri attribute value to: http://localhost:8080/wsdls/pubcalws-soap/wssvc.wsdl
      - Set the TzsUri attribute value to: http://localhost:8080/tzsvr

        - Click "Apply changes" button
    - Admin token:
        - If no admin token exists, click “generateAdminToken”
        - You should see a successful result; click "Back to MBean" button to return to "service=Eventreg"
        - You should see a string such as "c0e93685-93cd-4bee-bed2-9996b89be28c" in the EventRegAdminToken attribute value.
        - Click "Apply changes" button
        - Copy the EventRegAdminToken value (for use in the next step)

- Log into the Bedework public events administrative client, e.g. http://localhost:8080/caladmin
    - Go to the System tab and select "Manage System Preferences"
    - Paste the EventRegAdminToken value into the "Eventreg admin token" field
    - Click "Update" button to save your changes.

- Test:
    - Add a new public event. You should be able to select the checkbox "Users may register for this event", and make the event registerable.
    - Visit the event in public client -- you should be able to register for it.

Enabling Public Events Registration in Bedework 3.8

The public events registration features have been backported into Bedework 3.8. If you would like to install the system, perform the following steps: (Please note: the process outlined below is only set up for the default quickstart and postgresql configs at the moment.)

- Update the Bedework source, build the event registration system, and clean/build Bedework:
    - ./bw -updateall
    - ./bw -quickstart -eventreg
    - ./bw -quickstart clean
    - ./bw -quickstart deploy
      (These commands should be performed from the root of the quickstart directory. See also: building the Quickstart release.)

- Fix up quickstart
    - mkdir quickstart-3.9/jboss-5.1.0.GA/server/default/data/bedework/eventreg/
    - unzip pubcalws-soap.zip into quickstart-3.9/jboss-5.1.0.GA.server/default/deploy/ROOT.war/wsdls/

- Start up jboss and apacheDS

- Log into the JMX-Console shipped with Bedework's JBoss, e.g. http://localhost:8080/jmx-console
    - Click "org.bedework" in the left menu, then "service=Eventreg" in the right menu
        - If no database exists:
            - Set “create” and “export” attributes to "True"
            - Click "Apply changes" button (at the bottom of the list of attribute values)
            - Find the "schema" operation in the lower list, and click the “Invoke” button to export schema and create database
            - You should see a successful result; click "Back to MBean" button to return to "service=Eventreg"
        - Point at needed systems:
            - Set the WsdlUri attribute value to: http://localhost:8080/wsdls/pubcalws-soap/wssvc.wsdl
            - Set the TzsUri attribute value to: http://localhost:8080/tzsvr
            - Click "Apply changes" button
        - Admin token:
            - If no admin token exists, click “generateAdminToken”
            - You should see a successful result; click "Back to MBean" button to return to "service=Eventreg"
            - You should see a string such as "c0e93685-93cd-4bee-bed2-9996b89be28c" in the EventRegAdminToken attribute value.
            - Click "Apply changes" button
            - Copy the EventRegAdminToken value (for use in the next step)

- Log into the Bedework public events administrative client, e.g. http://localhost:8080/caladmin

    - Go to the System tab and select "Manage System Preferences"
    - Paste the EventRegAdminToken value into the "Eventreg admin token" field
    - Click "Update" button to save your changes.

- Test:
    - Add a new public event. You should be able to select the checkbox "Users may register for this event", and make the event registerable.
    - Visit the event in public client -- you should be able to register for it.

Personal & Group Calendaring

Documentation In Progress
This chapter of the Bedework 3.8 Documentation is in the process of being updated.

Overview

Bedework includes a fully realized, standards-based personal calendar system. Personal calendars allow users to carry out all standard calendaring functions and provide a customized view of public events through subscription to public calendars. Users can access their personal calendars through a web client and, via calDAV, on their personal devices using Mac iCal, iPhone calendar app, Mozilla Lightening, the EM Client, and any other CalDAV capable calendar.

Default calendars

A new user will have a set of default calendars created when they first log in. One of these is the default calendar for events named “calendar” (the name can be configured during system configuration or in the “Manage System Preferences/Parameters” of the admin client). In addition a set of special calendars are created, “Inbox” and “Outbox”. The inbox and outbox are used for scheduling meetings and supporting Bedework’s implementation of itip.

Subscriptions and views

The default state for a personal calendar user is to have one view with a name determined by the “defaultUserViewName” syspars setting (default “All”). This view contains one subscription to the user root collection at “/user/<account>” with the user account as the name. Only the default calendar with the name given by the “userDefaultCalendar” syspar is created.

Other special calendars, such as Inbox etc are only created as needed.

The initial default setting is normally created at the first login for that user. Because all calendars in a subscription are visible, if a user creates a new calendar it will automatically be visible in the default view. Thus the initial default state is relatively simple for users to manage and will probably be sufficient for most users.

Meeting Scheduling

The scheduling features of Bedework are almost complete; there is enough support to carry out simple scheduling. The flow of meeting requests and responses is defined by the relevant RFCs (2446, 2447) and the CalDAV scheduling extensions.

Calendar users and addresses

The potential attendees for a meeting may be internal to the system, that is they are Bedework users, or external. This is determined by their **cale ndar user address (CUA)** which looks like an email address. The Bedework system is identified by one or more email domains, for example **cal. mysite.edu** and an address in those domains is considered internal otherwise it is external.

For example, with the above domain, jim@cal.mysite.edu is internal, jim@thatsite.edu is external.

In addition, Bedework supports **principals** which look something like “/principals/resources/vcc311”. These are generally used to handle resources and locations but user principals are also mapped on to the email form of the calendar user address.

Bedework also allows the configuration option of preserving the domain part of a **CUA.** If we are not preserving the domain then a Bedework CUA of jim@cal.mysite.edujim@cal.mysite.eduwould map on to a Bedework user **jim**. For single domain systems this is more convenient.

Special Calendars: Inbox and Outbox

Each user has an inbox and an outbox. These are calendars with some special characteristics. Incoming scheduling requests always go to the inbox. They may arrive there via CalDAV, through uploading meeting requests or via an email interface. Scheduling responses to **external** users will go to the outbox. The may be immediately processed and at some point removed from the outbox.

To initiate a meeting request, use the add meeting link, add the attendees then continue on to set the details. Meeting requests must have one or more attendees and one originator (usually the current user) who will be added as an attendee.

The inbox and outbox will be created automatically when required. The actual names are configured during the build process so may be localized.

Incoming meeting requests will be placed in the default scheduling calendar: this can be configured in the user preferences of the personal web client. To respond to a meeting request, click on the event in the calendar.

Automatic processing

There are user preferences which indicate meeting requests can be automatically processed. If time is available for the incoming request it will be accepted, otherwise it will be declined. It is also possible to indicate that acceptances will be automatically processed; a meeting will have the attendee status updated automatically when the incoming response is an acceptance.

Scheduling resources

Bedework supports simple scheduling of resources. This is enabled with a degree of automatic processing of meeting requests to special resources. For example, if a room has the principal ***/principals/resources/vcc311*** then that principal can be added as an attendee to a meeting which is intended to be in that room. The aggregated freebusy for the attendees will be displayed which includes the free time in that room.

The meeting request will be processed and added to the room's calendar, effectively booking the room for that period.

Special Calendars: Deleted

This is here to allow users to subscribe to a calendar to which the have only read access but still be able to 'delete' events they do not want to see. For example, a user may subscribe to a 'films' calendar and delete those they are not interested in. On deletion of such an event we add an entry to the deleted calendar which acts as a mask on retrieval.

Note that there is a fix in the stylesheets which hide the delete action if the subscription is marked unremovable. The assumption is that such subscriptions require that the events also be unremoveable. These subscriptions can be used for class lists etc.

Calendar Sharing

Sharing calendars in Bedework is a two step process:

1. User A must grant access on the calendar or folder to be shared
2. User B must subscribe to the shared calendar or folder

Granting access to a calendar or folder

1. Select the "manage" link next to Calendars in the left menu
2. Select the calendar or folder to be edited
3. Grant rights on the calendar or folder to the appropriate user or group

If you are granting "write" or "all" access on a calendar or folder, and you want to see events published by another user, you must grant yourself explicit access on the calendar or folder as well. Why is this? Because "owner" access (which is by default "All") applies to the events within the calendar collection, not the calendar collection itself. So, when User B posts an event to User A's calendar, User A will not see it by default because she *doesn't own the event.* User A must explicitly grant herself access to the calendar or folder (e.g. "All" access to user id "userA") to see these events. ***

***This is standard CalDAV access control which is based on WebDAV access control. Bedework will simplify this process in future releases using a calendar sharing wizard, hiding the complexities of access control from regular users.

Subscribing to another user's calendar or folder

1. Select the "manage" link next to Subscriptions in the left menu
2. Select the link "Subscribe to another user's calendar"
3. Enter the following fields:
    - Name: an arbitrary name for the subscription; this will be displayed in the left menu.
    - UserID: the user id of the owner, e.g. caluser1

- Calendar Path: the path of the calendar *relative* to the calendar owner's root folder, e.g. "calendar". Enter no path to subscribe to another user's root folder.
- You can also specify the color for the subscription (style) and if the subscription should affect your freebusy.

Using Bedework CalDAV with Desktop & Mobile Clients

Using Mozilla Lightning with Bedework

Mozilla Lightning is a Thunderbird plug-in and CalDAV client that integrates well with Bedework. Bedework calendars are accessed from Bedework's CalDAV server. The following instructions were compiled using Lightning version 0.7 and Bedework 3.4.1.

1. Right-click within the calendar listing panel in Lightning (under "Calendar Name") and select "New Calendar".
2. At the prompt for creating a new calendar, select "On the Network".
3. At "Create New Calendar", select CalDAV and enter the path to the desired calendar via Bedework's user or public CalDAV server, e.g:

From the Bedework quickstart:

```
http://localhost:8080/ucaldav/user/vbede/calendar
http://localhost:8080/pubcaldav/public/Athletics
```

From a production Bedework: you must know the domain name and contexts provided for your production instance. By default the contexts are "/ucaldav" and "/pubcaldav":https://yourdomainname/ucaldav/user/someuser/calendarname

```
https://yourdomainname/ucaldav/user/someuser/calendarname
http://yourdomainname/pubcaldav/public/calendarname
```

1. Give the subscription a display name and a color

Using Apple's iCal with Bedework

Apple iCal is a CalDAV aware desktop client that that integrates with Bedework. Bedework calendars are accessed from Bedework's CalDAV server.

1. In iCal's menu, select iCal -> Preferences.
2. Select the Accounts tab, and click the "+" in the bottom left to create a new CalDav account. Enter a descriptive name and your Bedework user name and password. The "Account URL" field should look similar to that shown below. Only descend down the Bedework calendar tree to the user name, and note that the trailing slash is mandatory. Some URL examples:http://localhost:8080/ucaldav/principals/users/vbede/

   ```
   http://localhost:8080/ucaldav/principals/users/vbede/
   https://calendars.someuniversity.edu/ucaldav/principals/users/username/
   ```

3. If successful, your account should be created as shown:
4. You should now see your Bedework items and be able to write to the Bedework server.

Using the iPhone with Bedework

Apple's iPhone supports CalDAV in its native calendar client. To access a Bededwork calendar on the iPhone:

1. Go to “Settings” on the home screen
2. Select “Mail, Contacts, Calendars”
3. Select “Add Account
4. Select “Other”
5. Under calendars, select “Add CalDAV Account”
6. Add the server (e.g. “calendars.myserver.edu”), your userid, and password.
7. Select “Next” – the client will attempt to verify the account information and it will fail – but you now can access the “Advanced” menu.
8. Select “Advanced”

9. Select “Account URL”
10. Append the account URL to the root of your user calendar tree, e.g. https://calendars.someuniversity.edu/ucaldav/principals/users/username/ (You can also try /ucaldav/user/username/). Only descend down the Bedework calendar tree to the user name, and note that the trailing slash is mandatory when you enter the string.

Using CardDAV

Bedework's CardDAV Implementation

Bedework's CardDav server is a standards-compliant CardDav server that stores* and serves out v4.0 vCards. See http://tools.ietf.org/html/rfc6350. Bedework's CardDav server can be configured to use a variety of vCard data stores.   Out of the box, it looks for private address books in an HSQLDB database that is packaged in the Bedework distribution and looks for public address books (one for people and one for locations) from the directory server (ApacheDS) that is packaged with Bedework. By default the server is available at http://mybedeworkserver.edu/ucarddav.

As a standards-compliant server, it should work well with many of the address book clients available on PC's and phones, however because many of them are based on V3.0 vCards, it may not.

*depending on whether the source is r/w or r/o.

Working with the Quickstart

The Quickstart uses Bedework's directory server (buildt on Apache DS) to serve public contacts -- both people and locations.   Private address books are stored in an HSQLDB database. HSQLDB is the java-based database engine included in the quickstart.   Instructions for adding more public information are included below.
Three locations and three public people are preloaded into the quickstart and served out by Apache DS.   That data resides in <qs>/apacheds-1.5.3-fixed/bedework.org. Three of the preloaded Bedework users -- vbede, bfranklin, and mtwain -- have contacts stored in their address books.

Getting Bedework ready:

1. Kill bedework, if you've been running it.
2. cd <qs>
3. svn update bedework
4. svn update bedework-carddav
5. make a copy of your apacheds directory in case your directory gets hosed later:

   ```
   cp -rp apacheds-1.5.3-fixed apacheds-1.5.3-fixed.REFERENCE (or whatever)
   ```

6. review carddav settings in cal.options.xml
   - For example, <cardPathPrefixes>/public/people,loc_:/public/locations</cardPathPrefixes>. See note on "loc_" in the * Public Locations* section.
7. build bedework:

   ```
   ./bw -quickstart deploy
   ```

8. build carddav:

   ```
   ./bw -quickstart -carddav
   ```

   #(if needed) delete your carddav database to make sure that you're starting clean. (Bedework will recreate it when it starts):
   - cd <qs>/jboss-5.1.0.GA/server/default/data/hypersonic
   - delete or move to the side CardDb3p7*
9. In one window, start apache directory services
   - cd <qs>
   - /bw -quickstart dirstart
10. In other window, start bedework

- cd <qs>

  ```
  • ./startjboss
  ```

11. (if you deleted you database) Import the schema:
    - visit localhost:8080/jmx-console
    - Log in with username "admin" and password "bw" (if the login doesn't work, check the file <qs>/jboss-5.1.0.GA/default/server/conf/props/jmx-console-users.properties).
    - scroll all the way down and choose service=CardDumpres very near the bottom
    - set export to True
    - set create to True
    - (temporarily, pending fix) remove "carddumprestore" path component from SchemaOutFile
    - Apply Changes#* Move toward the bottom and clink on the Schema Invoke button.
    - Check the jboss window for "schema export complete".

Public Locations

We follow a convention of prefixing location calendar addresses with "loc_" to distinguish them from people. For example, here's the vcard for "Cafe de la Butte", one of the locations shipped with Bedework (http://localhost:8080/ucarddav/public/locations/CafeDeLaButte.vcf). Note the EMAIL attribute:

```
BEGIN:VCARD
VERSION:4.0
REV:20101012T010409Z
SOURCE:/ucarddav/public/locations//CafeDeLaButte.vcf
UID:/ucarddav/public/locations//CafeDeLaButte.vcf
KIND:location
NOTE:Cafe De La Butte\, 71 Rue Caulaincourt\, Paris\, France
EMAIL:loc_cafedelabutte@mysite.edu
CALADRURI:mailto:loc_cafedelabutte@mysite.edu
FN:CafeDeLaButte
END:VCARD
```

That email prefix can be changed in cal.options.xml inside the configuration directory.

Adding Public Locations

1. Because public locations are served by the built-in LDAP server, prepare them for import by preparing an LDIF file (let's say locations.ldif), with entries that look like this:

   ```
   dn: cn=CafeDeLaButte,ou=locations,ou=public,dc=bedework,dc=org
   objectclass: top
   objectclass: calendarresource
   objectclass: schedapprovalinfo
   objectclass: calEntry
   objectclass: room
   kind: location
   uid: bw000001@mysite.edu
   calCalAdrURI: mailto:loc_cafedelabutte@mysite.edu
   mail: loc_cafedelabutte@mysite.edu
   cn: CafeDeLaButte
   ou: locations
   description: Cafe De La Butte
   autoschedule: TRUE
   ```

2. (Download, Install and) Start up Apache Directory Studio and point it at the Bedework Directory Server

        - Add ldap connection
            - server: localhost:10389
            - user dn: uid=admin,ou=system password="secret"
3. Locate ou=locations under ou=public, under dc=bedework, dc=org## Import locations.ldif by right-clicking on ou=locations and choosing import/ldif import and browsing to its location.   If there's a chance of duplicates, click **Update existing entries**.
4. Test with http://localhost:8080/ucarddav/public/locations.

Adding Public People

Like with locations. public people need to be imported via an LDIF. So, if you're starting with a VCARD, this entry

```
BEGIN:VCARD
VERSION:4.0
EMAIL:tedison@mysite.edu
N:Thomas Alva Edison
CALADRURI:mailto:tedison@mysite.edu
FN:TEdison
WORK.TEL;TYPE=voice:222-222-2222
NICKNAME:Sparky
END:VCARD
```

needs to be turned into this entry:

```
dn: cn=TEdison,ou=people,ou=public,dc=bedework,dc=org
objectClass: organizationalPerson
objectClass: person
objectClass: inetOrgPerson
objectClass: calEntry
objectClass: top
cn: TEdison
sn: Thomas Alva Edison
calCalAdrURI: [mailto:tedison@mysite.edu]
displayName: Sparky
mail: tedison@mysite.edu
ou: people
telephoneNumber: 222-222-2222
uid: bwpp00002@mysite.edu
```

Then:

1. (if it's not running) Start up Apache Directory Studio
2. Locate ou=people under ou=public, under dc=bedework, dc=org
3. Import locations.ldif by right-clicking on ou=locations and choosing import/ldif import and browsing to its location.   If there's a chance of duplicates, click **Update existing entries**.
4. Test with http://localhost:8080/ucarddav/public/people

Importing vcard data into Bedework using the Bedework Address Book web client (for a single user's address book)

The Bedework address book client allows personal calendar users to import vcard data directly into their personal address books.

1. Log into the Bedework personal calendar.
2. Select "Contacts" in the left menu to open the Bedework address book client.
3. In the address book client menus, select Tools Import
4. In a text editor, open the vcard file you wish to import.
5. Copy the text of the vcard data and paste it into the text box in the Bedework address book client.
6. Click "Import".

The Bedework Addressbook Client

Bedework's Address Book client is a stand-alone JavaScript client. It can be directly invoked (by default at http://mybedeworkserver.edu/bwAddrbookClient), but is more typically invoked from within the personal client by clicking on **Contacts**.

Configuring the Bedework Addressbook Client

The Bedework Addressbook client may be connected to multiple vcard data sources. As shipped, it makes three connections to Bedework's CardDAV server: to the individual's personal addressbook, to a "public people" addressbook , and to a "public locations" addressbook.

To configure the client, visit
<quickstart>/bedework-cardav/clients/javascript/bwAddbookClient/config/config.js (source)
or
<quickstart>/jboss-5.1.0.GA/server/default/deploy/ROOT.war/bwAddrbookClient/config/config.js (deployed application)

Theming Bedework

Documentation In Progress
This chapter of the Bedework 3.8 Documentation is in the process of being updated.

Overview

This section is a resource for web designers who would like to customize the look and layout of the Bedework calendar web clients. No programming experience is assumed, though a basic understanding of how web applications work is helpful.

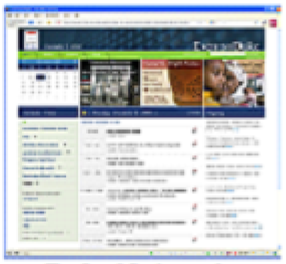
Themes

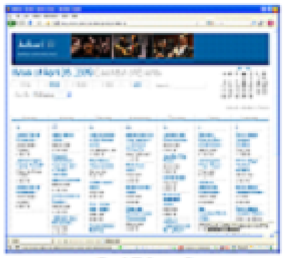
A generic theme is provided with each web application in the quickstart release. The public client's MainCampus calendar suite comes with a number of themes to get you started. Fonts, colors, and most layout specifics are defined with CSS.

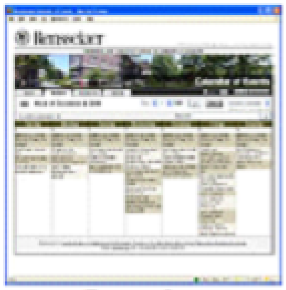
MainCampus skin

Examples of how members of the Bedework community have implemented themes can be viewed by selecting from our listing on the Bedework website, “Who's using Bedework?”: http://www.bedework.org/bedework/update.do?artcenterkey=35

Examples:

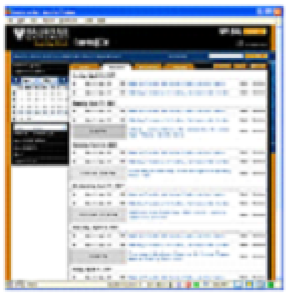

Duke University


Juilliard


Rensselaer

Bennington College

Universidad Pública de Navarra


Dalhousie University

Theming Prerequisites

Changing headers, footers, colors, and fonts can be accomplished with an understanding of XHTML and CSS. Changing global layout or presentation behavior requires an understanding of XML and XSL (xslt and xpath, in particular).

Because the calendar front-end uses XSLT to transform XML, you must use valid XML markup for all templates. Skins that produce HTML must be written using XHTML (regardless of the final output). Skins without valid markup will fail to transform.

Definitions

- **Theme:** a collection of XSL, CSS, image, javascript, and other resources used to deliver the look, feel, and functionality of a Bedework xml-based web client.
- **Skin:** a single XSL file, or a group of XSL files included into a master XSL file.

Location of the Theme Directories

In the quickstart directory, the theme files for the xml-based web applications are found in the following directories:

```
bedework/deployment/webpublic/webapp/resources/
 bedework/deployment/webuser/webapp/resources/
 bedework/deployment/webadmin/webapp/resources/
 bedework/deployment/websubmit/webapp/resources/
 bedework/deployment/feeder/webapp/resources/
```

If you rebuild the application (you will want to do this to create a production release), the skins are copied into:

```
jboss-5.1.0.GA/server/default/deploy/ROOT.war/calrsrc.MainCampus
  [jboss-5.1.0.GA/server/default/deploy/ROOT.war/calrsrc.OtherCalSuite]
jboss-5.1.0.GA/server/default/deploy/ROOT.war/ucalrsrc and
jboss-5.1.0.GA/server/default/deploy/ROOT.war/caladminrsrc and
jboss-5.1.0.GA/server/default/deploy/ROOT.war/eventsubmitrsrc and
jboss-5.1.0.GA/server/default/deploy/ROOT.war/calfeedrsrc.MainCampus
```

The *quickstart* distribution comes with the themes already in place and ready to use.

Editing, Building, and Avoiding SSL Mixed Content

Once deployed (as they are in the quickstart), the skins can be manipulated directly inside the JBoss ROOT.war directory for quickest development; be aware however that on a rebuild the source files will overwrite the JBoss files, so it is safest to edit in the source folder and copy files into the JBoss directories as you go. You may also choose to place your theme files on a separate web server. (Note: the destination folder into which the theme files are copied during a build is defined by the ...resources.dir property in the cal.properties config file.)

The locations of the theme directories are defined in the file

```
 bedework/config/bwbuild/<target>/cal.options.xml
```

which should be copied to your user home directory or to another location as described in Configuring Bedework.

For each web client, the **<appRoot>** is where the server will find the xsl files that transform Bedework's XML, and the **<browserResourceRoot>** i s where the browser will request css, images, javascript, and other resources. In the quickstart, these values are the same for each client and point to http://localhost:8080/_resourcesdir_.&nbsp;

In production, however, the <browserResourceRoot> must be served over https for secure web sites to avoid mixed content errors. To build a production release, change the values of <appRoot> and < browserResourceRoot> to the name of the server hosting the resources, and add https: to the <browserResourceRoot> for those web clients you will provide over SSL (e.g. personal, submissions, and admin web clients).

Please note: the <appRoot> should **not** be served over https

.
The <appRoot> takes the following value:

```
http://a-web-server-path/to/your/template/directory/
```

which for the quickstart is

```
http://localhost:8080/calrsrc/ and
http://localhost:8080/ucalrsrc/ and
http://localhost:8080/caladminrsrc/ and
http://localhost:8080/eventsubmitrsrc/ and
http://localhost:8080/calfeedrsrc/
```

The *appRoot* is modified at run time for portals and calendar suites. If we are running under a portal the approot has *.portalPlatform* appended. For example, if the portal platform is *uPortal2* the user client appRoot defined above becomes

```
appRoot = http://localhost:8080/ucalrsrc.uPortal2
```

In addition, we append the calendar suite name for public clients (all calendar suites and the feeder application). So if the calendar suite is

MainCampus we have

```
appRoot = http://localhost:8080/ucalrsrc.MainCampus
```

or for the portlet

```
appRoot = http://localhost:8080/ucalrsrc.uPortal2.MainCampus
```

Structure of the Theme Directories

The public web client appRoot directories have the following structure:

```
appRoot/
          default/ (default locale)
              strings.xsl (language text for default locale)
              default/ (default browser type)
                  default.xsl (references globals, strings, and theme index)
                  globals.xsl (global variables)
                  other skins (can be called dynamically)
              PDA/ (mobile browser type)
                  default.xsl (references iphone theme)
                  globals.xsl (global variables)
          themes/
              nameTheme/ (resources for a specific theme)
                  *.xsl (xsl template files)
                  css/ (css files)
                  images/ (css files)
                  javascript/ (css files)
```

The personal, admin, and submissions web clients use a slightly different structure for resource files and language strings. All templates in these clients are found in the default.xsl file for a particular skin. These will be refactored in upcoming releases to match the current structure of the public clients. Behaviors are otherwise the same among all the xsl clients.

Stylesheet discovery

When a Bedework web application is requested, the server goes through a process of stylesheet discovery in three steps: selecting the locale, selecting the browser type, and finally selecting the skin.

Locale selection

The top-level directories of the *appRoot* define locales. The default locale is *default*. You may add directories here using a locale name such as *en_US* or *fr_CA*. Browsers provide Bedework with a locale; if a directory is found with a name that matches the locale provided by the browser, Bedework will use the skins found there. Otherwise, the default locale directory will be used.

For a locale to be supported by Bedework it must be included in the list of supported locales in the Admin Client.   For example, if you wish to support *fr_CA*, visit http://localhost:8080/caladmin System tab Manage system preferences, and add *fr_CA* to the list of supported locales.

Browser type selection

The next level down defines the browser type. Bedework looks first for a folder whose name is associated with the user-agent of the visiting browser. If found, Bedework will use that folder to deliver the skin. If not found, Bedework will use the default directory seen here. Currently, the acceptable folder names are:

- *default*
- *Netscape4*
- *MSIE, Opera*

- *Mozilla*
- *PDA.*

Bedework can be forced to use a specific browser type and skin and can have multiple skins per browser type. See Actions & Parameters for more information about this.

This code is somewhat archaic, and PDA user-agent sniffing is out-of-date; to use the PDA directory found in the quickstart release, you can explicitly request it by adding the request parameter *browserTypeSticky=PDA.* *browserTypeSticky=default* will reset this to the default path.

Skin selection

Once the locale and browser type paths have been selected, the server will select the xsl **skin** used for transforming Bedework's XML. If no skin is explicitly requested, the server will fall back to default.xsl. A skin can be directly requested using the request parameter *skinName=someskin* or *skinNameSticky=someskin*. See Actions & Parameters for more information about this.

In the public client, the default.xsl skin includes the global.xsl file for global variables, the strings.xsl file for internationalized language strings, and the root xsl file of the theme that will be used by default. Here is the example from default.xsl of the MainCampus calendar suite:

```
<!- DEFINE INCLUDES ->
  <xsl:include href="./globals.xsl" />
  <xsl:include href="../strings.xsl" />

  <!- DEFAULT THEME NAME ->
  <!- to change the default theme, change this include ->
  <xsl:include href="../../themes/bedeworkTheme/bedework.xsl" />
```

In the personal, submissions, and admin client, *default.xsl* includes *strings.xsl*. The rest of the xsl logic is self-contained.

Skins are *only* loaded at first reference and are then cached. The skins can be explicitly flushed using the ***refreshXslt=yes*** query parameter (see Actions and Parameters below). Most themes shipped with Bedework have a ***Refresh XSLT*** link in the footer of the web client. (You will probably want to disable this link prior to production.)

The discovered 'real' path is cached with the 'idealized' path as the key so that subsequent lookups for the same path will proceed without the discovery phase.

Internationalization

As a first step towards internationalizing Bedework, nearly all English strings have been pulled from the xsl templates that produce the personal, public, admin, and submission web clients.

The public client locale directories contain two files, *strings.xsl* and *localeSettings.xsl*. These two files contain the replacement text and settings for the language associated with the current locale. The themes reference the current locale's strings and localeSettings files. The strings should be translated to the default language you wish to use for your site. If you wish to support other languages in the same client, create a locale directory and place a translated strings and localeSettings files into it.

For example:

```
appRoot/
          default/ (default locale)
              strings.xsl (language text for default locale)
              localeSettings.xsl(internationalization settings)
              default/
                  default.xsl
                  globals.xsl
          fr_CA/ (locale for French, Canadian)
              strings.xsl (language text for fr_CA locale)
              localeSettings.xsl(internationalization settings)
              default/
                  default.xsl
          themes/
```

Each *default.xsl* includes the strings file from its locale directory. Each otherwise includes *default/default/globals.xsl* file and the theme of its choice. Here is an example of what the fr_CA default.xsl would look like for the MainCampus calendar suite:

```
<!-- DEFINE INCLUDES -->
<xsl:include href="../../default/default/globals.xsl" />
<xsl:include href="../localeSettings.xsl" />
<xsl:include href="../strings.xsl" />


<!-- DEFAULT THEME NAME -->
<!-- to change the default theme, change this include -->
<xsl:include href="../../themes/bedeworkTheme/bedework.xsl" />
```

Caveats

In the Administration and Personal web clients, there are a handful of English strings that have yet to be pulled from the xsl template. Also, there are a few English strings that are written out by Javascript routines that reside in bedework-common/javascript. Those routines have not been prepared for translation yet.

Making Stylistic Changes – the Public Client Themes

The public themes consist of a collection of xsl skins that contain a series of xsl *templates*. The theme directories are found in

```
bedework/deployment/webpublic/webapp/resources/demoskins/CalSuiteName/themes
```

In the MainCampus/themes directory, you'll find Bedework default theme *bedeworkTheme*. The root xsl file is called *bedework.xsl*. Likewise, in the iphoneTheme, the root xsl file is called *iphone.xsl*.

In each theme directory, you will also find a *themeSettings.xsl* file in which the *resourcesRoot* variable is defined. *resourcesRoot* is used throughout the xsl documents to reference css, images, and javascript.

If you have a good grasp of XHTML and CSS, you can modify the graphics, colors, and general feel of a skin by editing one of the examples and copying the changes into its associated resources directory in JBoss. Skins are cached in memory, so to update a skin you must refresh the stylesheet by appending your query string with *refreshXslt=somestring*. For example:

```
http://localhost:8080/cal/setup.do?refreshXslt=yes    -or-
http://localhost:8080/cal/event/eventView.do?eventId=2&refreshXslt=yes
```

Working with the bedeworkTheme

Bedework's default theme is based on work at Duke and Yale Universities. The theme includes a number of features that can be configured in *th emeSettings.xsl*. These include:

- **Ongoing events:** you can specify a category to mark events as ongoing, and display these events in a separate listing. By default, the category "Ongoing" (as shipped with the quickstart's default data) is used to mark ongoing events.
- **Featured events:** the three graphics at the top of the default theme are "featured events". The images and links used are defined in *bed eworkTheme/featured/FeaturedEvent.xml*. If you choose to use featured events, you must (for now) maintain this xml file outside the Bedework system (either manually or using an outside process to write the xml file).
- **Configurable event icons:** the icons used to download or share events can be enabled and disabled by editing the *themeSettings.xsl*.
- **Configurable header, footer, left-sidebar links, and favicon:** all can be defined in the themeSettings.xsl file. By editing this file and manipulating the CSS found in *css/bwTheme.css,* it is possible to create a highly customized instance of Bedework without having to dig deeper into the xsl code.

Setting event colors in the Public Web Client

Events can be colored by category in the public web client if you choose. This is currently accomplished by customizing a template in the stylesheet.

For example, to color events in the calendar grid of the bwclassicTheme, search for the template that looks like this (approx. line 170 of themes/bwclassicTheme/eventGrids.xsl):

```
<xsl:template match="categories" mode="customEventColor">
  <!-- Set custom color schemes here. -->
  <xsl:choose>
    <!--
    <xsl:when test="category/value = 'Athletics'">bwltpurple</xsl:when>
    <xsl:when test="category/value = 'Arts'">bwltsalmon</xsl:when>
    -->
    <xsl:otherwise></xsl:otherwise> <!-- do nothing -->
  </xsl:choose>
</xsl:template>
```

Uncomment the category tests and add your own logic for setting the event colors. The values such as *bwltpurple* are css classes that will be applied to each event in the grid or list, and are found in *bedework/deployment/resources/xsl/default/default/subColors.css*

This file gets deployed to the bedework-common directory and is referenced by the public web client.

You can use this approach to color events in other ways or use your own custom classes.

Making Stylistic Changes – the Personal, Admin, and Submissions Clients

In future versions of Bedework, we will refactor the personal, admin, and submissions clients to use the same structure as the public themes. For now, theming for these clients involves modifying the appRoot/default/default/default.xsl files and default.css files.

Actions & Parameters

What are Actions & Parameters?

Typical of web applications, Bedework receives HTTP requests and returns responses based on the page or action requested and the parameters sent in the query string. Look at the following URL:

```
http://localhost:8080/cal/main/setViewPeriod.do?b=de&viewType=weekView&date=20091121
```

Bedework is built in the Apache Struts MVC framework, and as such does not reference "web pages" through URLs directly. *setViewPeriod.do* in the URL above calls a Java "action" that returns a response based on the query string "date=20101121". This URL tells the personal calendar to set the current view to November 21, 2010.

Parameters can be strung together with ?'s and &'s on a query string like so:

```
http://hostname:port/context/action?param1=value&param2=value
```

Normal Actions & Render Actions

Request processing in Bedework is divided into two parts: a normal action that may change the state of the application, and a render action that returns the resulting state for display. This is required, among other things, for Bedework to run as a portlet.

For each normal action that is called, Bedework will automatically redirect to the appropriate render action. Normal actions take a *.do* extension. Render actions have an *.rdo* extension.

As the developer of a skin, you will be primarily concerned with normal actions, and it is these that will be presented in the XSL stylesheets for users to click. All actions and parameters described below are normal actions.

Bedework Actions & Parameters in Detail

All actions used by the Bedework web clients are defined in the client's *struts-config.xml* file. For example, the public and personal web clients are described in

```
bedework/projects/webapps/webclient/war/WEB-INF/struts-config.xml
```

and all actions used by the Bedework admin web client are described in

```
bedework/projects/webapps/webadmin/war/WEB-INF/struts-config.xml
```

For detailed information about all actions and parameters used in Bedework, please see the API reference found from the Bedework Wiki.

Bedework Skin Parameters

The following parameters provide control over stylesheets. They can be added to any Bedework URL and combined with any other parameter.

setappvar

setappvar=*key(value)* - Pass a key/value pair into the XML output.

- Use for passing a parameter between pages, much like in other frameworks,
- Pass as many appvars as you need
- Change the value of an appvar by sending the same key with a different value.
- ***Note:*** appvars, once set, persist through a user session

skinName, skinNameSticky

skinName=*name* - Set the skin to *name* during for the request

skinNameSticky=*name* - Set the skin to *name* for the rest of the session (or until another skin is called).

where name is the file name of an XSLT document at the bottom of the *locale/browserType* path, leaving off the .xsl extension.For example:

```
appRoot/
    default/   (locale)
   default/    (browserType)
           default.xsl
       shiny.xsl
       cleanXml.xsl
    themes/
```

If *skinName* is unspecified, Bedework uses the default skin. The URL to select "shiny.xsl" might look like this: http://hostname/cal/setup.do?skinNameSticky=shiny

```
http://hostname/cal/setup.do?skinNameSticky=shiny
```

browserType, browserTypeSticky

browserType=*default, MSIE, Netscape, Netscape4, Mozilla, PDA, other* - Set browser folder during request

browserTypeSticky=*default, MSIE, ...* - Set browser folder for the remainder of the session (or until another *browserType* is called).

For example:

```
appRoot
    default/
        default/
            default.xsl
        Mozilla/
            default.xsl
        PDA/
            default.xsl
    themes/
```

If unspecified, Bedework uses the folder that most closely matches the user-agent of the requesting browser. If not found, the "default" folder will be used. The example above would use the Mozilla folder for an incoming Mozilla browser. You may also create your own folder and call it explicitly, though Bedework cannot automatically associate it with a user-agent.

Note: the PDA user-agent list is not currently up-to-date; to use the PDA browser path, call it explicitly.

refreshXslt=*anystring*

XSL skins are cached once per user session to improve performance. If you make a change to your skin, you need to pass this parameter to reload it. *anystring* can be any value; we typically set it to *yes*. Example:

```
update.do?catcenterkey=12&skinName=shiny&refreshXslt=yes
```

noxslt=*anystring*

This parameter turns off the XSLT filter and returns raw xml in the response. You can look at the xml by selecting "view source" from your browser. This feature is very important when designing skins because it allows you to reference the exact XML you are trying to transform. *anystring* can be any value; we typically set it to *yes*. Example:

```
update.do?catcenterkey=12&noxslt=yes
```

XML

Bedework XML Structure

In Bedework, you can effectively look at the XML in two ways:

First way: add noxslt=yes parameter

In a user client (guest or personal) append noxslt=yes to the query string and view the page source. For example:

```
http://localhost:8080/cal/setup.do?noxslt=yes
```

There is a link in the footer of each web client that reads “show XML” which adds this parameter to the query string for the current page. When manipulating the XSLT, this is the easiest way to understand the underlying XML that is being transformed.

Second way: examine the source (JSP)

Look at the JSP pages which produce the XML output. These can be found in:

```
bedework/projects/webapps/client/war/docs
```

where *client* is **webclient**, **webadmin**, **websubmit**, or **feeder**.

Each “page” of an XML response takes the following form:

```
<bedework>

  [header data/variables and urlPrefixes for building links]

  <page>pageName</page>
  [page specific XML – changes from page to page]


  [footer data/variables (if any)]

</bedework>
```

In most cases, the skins look first at the *pageName* found in the incoming XML and branch to an appropriate XSL template based on that. Look to the default template (the first in the stylesheets, matching on “/”) to see a listing of all incoming “pages” and their associated XSL templates. For example of this branching, look in

```
/bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/themes/bedeworkTh
eme/bedework.xsl
```

Bedework XSLT

Bedework uses Apache Xalan as its XSLT processor, and processes XSLT version 1.0 and Xpath version 1.0. Bedework uses the standard Xalan libraries without extensions.   The XSLT language is reasonably simple, and if you are familiar with basic programming or scripting, you will have little difficulty reading it.

The key to understanding how the transformations take place, however, is to look at the underlying XML (by switching off the transform in Bedework using the “noxslt=yes” parameter and viewing the page source) and recognizing how XPath expressions are used to locate the xml nodes for transformation.

As noted above, a good place to begin understanding how the XML is transformed is

```
/bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/themes/bedeworkTh
eme/bedework.xsl
```

XSLT References

XSLT & XPath Quick Reference (PDF): http://www.mulberrytech.com/quickref/XSLT_1quickref-v2.pdf

XPath Specification: http://www.w3c.org/TR/xpath

XSLT Specification: http://www.w3c.org/Style/XSL/

Adding "Share This"

Adding the "Share This" Service to the Bedework Theme

To add the Share Thisservice to the new Bedework Public theme, you must configure it explicitly for your site.

1. Visit http://sharethis.com/developers/api and follow the steps.
2. You will need to edit head.xsl, eventList.xsl, and event.xsl in the Bedework Theme and add your javascript code. (The Bedework Theme is found in bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/themes/bedeworkTheme.)

Internationalization

Introduction

These are some guidelines to help you with Bedework internationalization. Internationalization of a calendaring system is not only a question of translations of the names and literals, there are also some other differences from one country or culture to another on how time is named or handled. For example, for some the weeks start on Sunday, for others on Monday.

Nevertheless, Bedework is designed to easily manage language and calendaring internationalization issues.

In order to handle your locale needs correctly, all the members in the chain from browser to the backend DB must be configured accordingly. This section, as an example, will show how Spanish localization is accomplished. The same procedure can be used for any language.

Overview for the impatient

The advice found in the remainder of this chapter can be summarized by the following checklist:

1. Check that the Operating System (OS) has your locale configured

> *(For example, setting LC_ALL=es_ES.UTF-8 and LANG=es_ES.UTF-8 , check control panels in windows).*

2. Check that your JVM is running with the right locale configuration.

> *(For example, look at your JAVA_OPTS for -Duser.timezone=Europe/Madrid -Duser.country=ES -Duser.language=es)*

3. Check your DB can handle multilingual data.

> *(Details depends on your DBMS (Oracle, MySql, etc.), but usually DB must be configured UTF-8)*

4. Check you have included your locale pages (xsl, css, etc) in the correct application directory.

   > *(Bedework has english, spanish and german translations included. These translations handle the translation of literals (‘save’, ‘cancel’, etc.) for your language. If you need other language than these, take a look at a folder like jboss-5.1.0.GA\server\default\deploy\ROOT.war\calrsrc.MainCampus\es_ES\ and there you can find the files used for spanish translation as an example. You should add a new folder ./ROOT.war/calrsrc.MainCampus/YOUR_LANG with your translation)*

5. Ensure that the Bedework administrative client includes the languages you intend to deploy on the "Manage System Preferences" page

   > *(In the admin client, visit System --> Manage System Preferences, and enter the locales (e.g. en_US,es_ES) in the "Supported Locales" field.)*

6. If client does not get the proper locale pages, check he has correctly configured the language options of his navigator.

   > *(Bedework, tries to show the calendar using the preferred language set in the browser; check your browser's language settings.)*

Summarizing, for example, you can make a custom script to start Bedework on your language as this:

```
# Custom script for launching quickstart in spanish.

JAVA_HOME=/home/bedework/jdk160_27
PATH=$PATH:$JAVA_HOME/bin

export JAVA_HOME PATH

LC_ALL=es_ES.utf8
LANG=es_ES.utf8
export LC_ALL LANG

JAVA_OPTS="$JAVA_OPTS -Duser.timezone=Europe/Madrid -Duser.country=ES
-Duser.language=es -Dfile.encoding=UTF-8 -Duser.variant=ES "
export JAVA_OPTS

cd /home/bedework/quickstart-3.7

echo "Starting Apache DS in the background..."

./bw -quickstart dirstart >/home/bedework/last_bedework_startup.log &

echo "Starting Bedework in the background..."

./startjboss -heap 512M >>/home/bedework/last_bedework_startup.log &
```

JVM and Operating System configuration

JVM and Operating System (Checks #1, #2)

Some of the string values and date formats that appear in client’s browser are taken by Bedework from the locale management of the operating system on the server. The JVM on which Bedework runs takes some of the string values and date formats from the underlying OS using java locale, date and time classes. So it is quite important that both, JVM and OS, are configured correctly.

Operating System checks

To check the current locale configuration of the OS you can use the command “locale” in Unix boxes or checking the language options of Control

Panel in Windows machines

In Unix the locale command returns something like this:

```
$ locale
LANG=en_US.UTF-8
LC_CTYPE="en_US.UTF-8"
LC_NUMERIC="en_US.UTF-8"
LC_TIME="en_US.UTF-8"
LC_COLLATE="en_US.UTF-8"
LC_MONETARY="en_US.UTF-8"
LC_MESSAGES="en_US.UTF-8"
LC_PAPER="en_US.UTF-8"
LC_NAME="en_US.UTF-8"
LC_ADDRESS="en_US.UTF-8"
LC_TELEPHONE="en_US.UTF-8"
LC_MEASUREMENT="en_US.UTF-8"
LC_IDENTIFICATION="en_US.UTF-8"
LC_ALL=
```

In this case “en_US" (English – United States) is the default language configuration. To change it to "es_ES" (Spanish – Spain) use the following commands:

```
$ export LC_ALL=es_ES.UTF-8
$ export LANG=es_ES.UTF-8
```

And now the command locale should show

```
$ locale
LANG=es_ES.UTF-8
LC_CTYPE="es_ES.UTF-8"
LC_NUMERIC="es_ES.UTF-8"
LC_TIME="es_ES.UTF-8"
LC_COLLATE="es_ES.UTF-8"
LC_MONETARY="es_ES.UTF-8"
LC_MESSAGES="es_ES.UTF-8"
LC_PAPER="es_ES.UTF-8"
LC_NAME="es_ES.UTF-8"
LC_ADDRESS="es_ES.UTF-8"
LC_TELEPHONE="es_ES.UTF-8"
LC_MEASUREMENT="es_ES.UTF-8"
LC_IDENTIFICATION="es_ES.UTF-8"
LC_ALL=es_ES.UTF-8
```

For Unix boxes: Your server may not have all possible locale configurations installed. Some Unix(es) gives you some information about it with non-priviledged commands. In Linux (Fedora, CentOS, etc) the command “set” gives you some info.

```
…..
SUPPORTED=eu_ES.UTF-8:eu_ES:eu:en_US.UTF-8:en_US:en:fr_FR.UTF-8:fr_FR:fr:de_DE.UTF-8:d
e_DE:de:es_ES.UTF-8:es_ES:es
…..
```

⚠ **Make sure the language you want to support is installed on your operating system**
Be aware that you can configure and issue the "$export LC_ALL=<locale>" even when that locale is not really supported on your installation. Even more, after that unsupported configuration, the "locale" command shows supposed current locale even when the OS does not really manage it if it is not installed.

JVM Checks

The JVM takes some environment values from the operating system by default. To check the default locale (among others) your JVM is using you can use this class:

```
(See http://www.dickbaldwin.com/java/Java052.htm):

import java.util.*;
import java.lang.*;

class Prop01{
  public static void main(String[] args){

    //Instantiate and display a Properties object
    // containing the system properties

   Properties obj = new Properties(System.getProperties() );
   obj.list(System.out);

  }//end main()

}
```

Running this class in command line, " java Prop01 ", returns a list like this:

```
java.runtime.name=Java(TM) 2 Runtime Environment, Stand...
sun.boot.library.path=/apps/jdk/jdk142_04/jre/lib/sparc
java.vm.version=1.4.2_04-b05
java.vm.vendor=Sun Microsystems Inc.
java.vendor.url=http://java.sun.com/
path.separator=:
java.vm.name=Java HotSpot(TM) Client VM
file.encoding.pkg=sun.io
user.country=ES
sun.os.patch.level=unknown
java.vm.specification.name=Java Virtual Machine Specification
user.dir=/export/home/unavarra
java.runtime.version=1.4.2_04-b05
java.awt.graphicsenv=sun.awt.X11GraphicsEnvironment
java.endorsed.dirs=/apps/jdk/jdk142_04/jre/lib/endorsed
os.arch=sparc
java.io.tmpdir=/var/tmp/
line.separator=

java.vm.specification.vendor=Sun Microsystems Inc.
os.name=SunOS
sun.java2d.fontpath=
java.library.path=/apps/jdk/jdk142_04/jre/lib/sparc/cli...
java.specification.name=Java Platform API Specification
java.class.version=48.0
java.util.prefs.PreferencesFactory=java.util.prefs.FileSystemPreferences...
os.version=5.8
user.home=/export/home/unavarra
user.timezone=
java.awt.printerjob=sun.print.PSPrinterJob
file.encoding=ISO8859-15
java.specification.version=1.4
user.name=unavarra
java.class.path=.:/apps/jdk/jdk142_04/lib/tools.jar:/...
java.vm.specification.version=1.0
sun.arch.data.model=32
java.home=/apps/jdk/jdk142_04/jre
java.specification.vendor=Sun Microsystems Inc.
user.language=es
java.vm.info=mixed mode
java.version=1.4.2_04
java.ext.dirs=/apps/jdk/jdk142_04/jre/lib/ext
sun.boot.class.path=/apps/jdk/jdk142_04/jre/lib/rt.jar:/a...
java.vendor=Sun Microsystems Inc.
file.separator=/
java.vendor.url.bug=http://java.sun.com/cgi-bin/bugreport...
sun.cpu.endian=big
sun.io.unicode.encoding=UnicodeBig
sun.cpu.isalist=sparcv9+vis2 sparcv9+vis sparcv9 spar...
```

From this list, the relevant values in our case are:

```
user.country=ES
user.timezone=
user.language=es
```

Notice that the value “user.timezone=” is empty. To run the JVM with the correct ones, the Java command line options can be used, as:

```
$java -Duser.timezone=Europe/Madrid -Duser.country=ES -Duser.language=es Prop01

java.runtime.name=Java(TM) 2 Runtime Environment, Stand...
sun.boot.library.path=/apps/jdk/jdk150_02/jre/lib/i386
java.vm.version=1.5.0_02-b09
…..
user.country=ES
user.timezone=Europe/Madrid
user.language=es
…..
```

You should change those environment variables before starting your Bedework installation like this:

```
(Unix)

JAVA_OPTS="$JAVA_OPTS -Duser.timezone=Europe/Madrid  -Duser.country=ES
-Duser.language=es -Dfile.encoding=UTF-8 -Duser.variant=ES"
```

```
(windows)

SET JAVA_OPTS=%JAVA_OPTS%  -Duser.timezone=Europe/Madrid  -Duser.country=ES
-Duser.language=es -Dfile.encoding=UTF-8 -Duser.variant=ES
```

The startjboss or startjboss.bat inherits the values of JAVA_OPTS and adds them to the ones it sets itself.

Ensure your database can handle multilingual data

Multilingual database configuration

If you intend to manage multilingual data like special chars and formats, obviously your DBMS and DB must be configured to handle them. Your DB could have been created with a simple locale configuration and could not permit special chars like diacritics, UTF-8 representations and so. In many DBMSes the locale options for DBs are defined at the creation of the DB. Consult your DBA if you find problems at this point.

For example, in Oracle you can check if your DB is created with multilingual support with this command, and the values that are correct are shown below:

```
select * from nls_database_parameters where parameter like '%CHARACTERSET'

NLS_CHARACTERSET    AL32UTF8
NLS_NCHAR_CHARACTERSET    UTF8
```

In many others, like MySql, **UTF-8** should be the default character set defined at creation time of the DB.

Custom translations

Adding a new translation

Bedework has english, spanish and german translations included by default. These translations handle the translation of literals (‘save’, ‘cancel’, etc.) for your language. So, if your navigator has one of those languages as preferred language, no other steps are required to see Bedework in your language.

If you need a language other than these, perform the following steps. These steps outline the process used for the MainCampus calendar suite of the public web client, but the process would be the same for a different public calendar suite or for the the personal web client.

1. Copy an existing translation, e.g. "default", "es_ES", or "de_DE" found in the Bedework source in <quickstart>/bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/

    Example:

    ```
    cp
    <quickstart>/bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/
    es_ES
    <quickstart>/bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/
    fr_CA
    ```

2. Update the MainCampus/YOUR_LANG/**strings.xsl** and MainCampus/YOUR_LANG/**localeSettings.xsl**files inside your new locale directory. Most changes will be to the strings.xsl file, translating the many strings used by the web client.
    - The strings.xsl file contains a long list of strings used in the web client user interface.
    - The localeSettings.xsl file contains layout templates for locale-specific date handling. At the moment, this includes only the settings for the javascript calendar widget used in the public web client.

3. IF you wish to use your new locale directory as the default language,
    a. rename MainCampus/**default** to MainCampus/**en_US**
    b. rename your new locale directory to MainCampus/**default**

When Bedework is rebuilt, your locale directory will be deployed to:

```
<quickstart>/jboss-5.1.0.GA/server/default/deploy/ROOT.war/calrsrc.MainCampus/YOUR_LANG
```

If you don't wish to rebuild, you can copy your directory in manually.

Using Apache to front Bedework?
If you're using Apache in front of Bedework, and you've moved your theme directories to be served out of Apache, make sure you manually copy your new locale directory to that location.

Add your new locale to the list of supported locales in the administrative web client

In the admin client, visit System --> Manage System Preferences, and enter the locales in the "Supported Locales" field. (It is the last field on the page.) This field accepts a comma separated list of locales (e.g. "en_US,es_ES").

Language preferred by the browser

Showing what the client wants

Bedework tries to show calendars in the preferred language set in the client's web browser.

If a user reports that he can not see the pages in the locale he wants, it is important to check if the client really has properly configured his default or current language in the browser.

System Updates, Upgrades, and Tuning

Overview

These sections address how to apply patches, tune, and upgrade your Bedework environment.

Applying Fixes and Updates to Bedework

Use the bw command to update everything

The simplest way to update Bedework is to use the ***bw -updateall*** command:

```
cd <quickstart>
./bw -updateall
```

This command issues a series of svn update commands against those projects essential to Bedework.

Remember to rebuild and restart!
Updates to the source code won't take effect until you rebuild Bedework and (in many cases) restart.

Updating specific files or projects

Bedework code repository

The Bedework project files are kept in several subversion repositories located at https://www.bedework.org/svn/<projectName>/...   For example, the Bedework core trunk is stored at https://www.bedework.org/svn/bedework/trunk, the Bedework core 3.7 branch at https://www.bedework.org/svn/bedework/releases/bedework-3.7, and the Bedework CardDav project trunk at https://www.bedework.org/svn/carddav/trunk.

As long as you know the project name, you can figure out the rest, so here's a guide:

Project	Repository URL	Notes
bedework	https://www.bedework.org/svn/bedework	Many projects, not listed, are linked to Bedework. When you update Bedework, the linked projects are updated, too.
carddav	https://www.bedework.org/svn/carddav	Carddav server and Addressbook client
bwxml	https://www.bedework.org/svn/bwxml	XML definitions for Bedework
bwtzsvr	https://www.bedework.org/svn/bwtzsvr	Timezone Server
cachedfeeder	https://www.bedework.org/svn/cachedfeeder	Page Caching application and web form front-end
etc		

The quickstart ships with subversion information for some of the directories contained within the quickstart. This means, assuming you have subversion installed, you can use the ***svn update*** command to apply specific fixes or to bring the distribution up to date.

Call svn update to update a single file or directory

To update the bedework folder under the quickstart:

```
cd <quickstart>
svn update bedework
```

To update only the public client's Bedework theme

```
cd <quickstart>
cd
bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/themes/bedeworkThe
me
svn update .
```

Reindexing Bedework

Though unusual, it is possible for the Lucene indexes used for text searches to get out of step (for example, after a system crash).

If you find you need to reindex Bedework for searching,

1. Log in to the JMX-Console: https://yoursite/jmx-console

2. Select "org.bedework" from the left-most "Object Name Filter" menu. "org.bedework" is usually the bottommost link.

3. Select "service=Indexer"

4. Click the button "Invoke" for the operation "rebuildIndex" (this is the bottommost operation in the list).
   Note: this process may take a while.

When completed successfully, you should see a message such as the following in your JMX console (your numbers will be larger, depending on the size and age of your installation):

```
[About to build directories at
[yourPathHere]\jboss-5.1.0.GA\server\default\data\bedework\/lucene/indexroot/new
, Statistics for User
,              users: 26
,        collections: 128
,           entities: 0
, unreachableEntities: 0
, Statistics for Public
,              users: 0
,        collections: 71
,           entities: 8
, unreachableEntities: 0
, About to rename directories from
[yourPathHere]\jboss-5.1.0.GA\server\default\data\bedework\/lucene/indexroot/current
to [yourPathHere]\jboss-5.1.0.GA\server\default\data\bedework\/lucene/indexroot/old
, Unable to rename current index at
[yourPathHere]\jboss-5.1.0.GA\server\default\data\bedework\/lucene/indexroot/current
to [yourPathHere]\jboss-5.1.0.GA\server\default\data\bedework\/lucene/indexroot/old
]
```

Tuning Bedework

Database Caching and Hibernate

The performance of your system is likely to be very dependent upon the caching strategies used. For small to medium deployments the default settings may be perfectly adequate.

For large systems you may need to spend some time reading the documentation available at the Hibernate site and the sites of the various cache implementors.

Bedework allows different cache regions for each application type. This allows the deployer to adopt aggressive caching strategies fro the public events clients for example, while having less caching or much shorter flushing intervals for personal clients where immediate visibility of changes is necessary.

There are two parameters in the bwbuild/*myconfig*/cal.options.xml file which affect caching. The default settings are

```
<cachingOn>true</cachingOn>

<cachePrefix>bwpubevents</cachePrefix>
```

These settings turn caching on and set the cache prefix to “bwpubevents”. This is used in the ehcache settings file to distinguish cache settings.

The default cache for Bedework is currently Ehcache. The latest versions of this cache do allow clustering. Other caching schemes are available such as the Jboss cache. The Hibernate site offers details on the alternatives.

Setting JVM Parameters

See "Reviewing JVM parameters"

Upgrading

Bedework has been running at your site for some time and new versions have appeared and now it's time to upgrade. How do you migrate all that data from one version to the next?

Upgrading from version 3.5 or later involves dumping your Bedework database using the dump/restore utility and then restoring the file into the new version.

Upgrading from an earlier version of Bedework older than 3.5 uses the same process, but for public calendaring requires some post-processing of the data to migrate from a multiple calendar model to a single calendar model. A rudimentary script is available in bedework/projects/bwtools to help with this process.