bedework

Bedework Calendar System 3.5

Bedework version 3.5

Last modified: October 19, 2009

http://www.bedework.org

Bedework Calendar System

The Bedework Calendar System Manual contains an overview of the system, instructions for customizing and installing a production version of Bedework, administration and user guides, and detailed descriptions of Bedework components.

Table of Contents

Chapter 1 Introducing Bedework 3.5

1.1 Bedework System Overview

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.

Bedework System Architecture

Bedework 3.5 has a central server architecture and is modular and extensible. It consists of the following components:

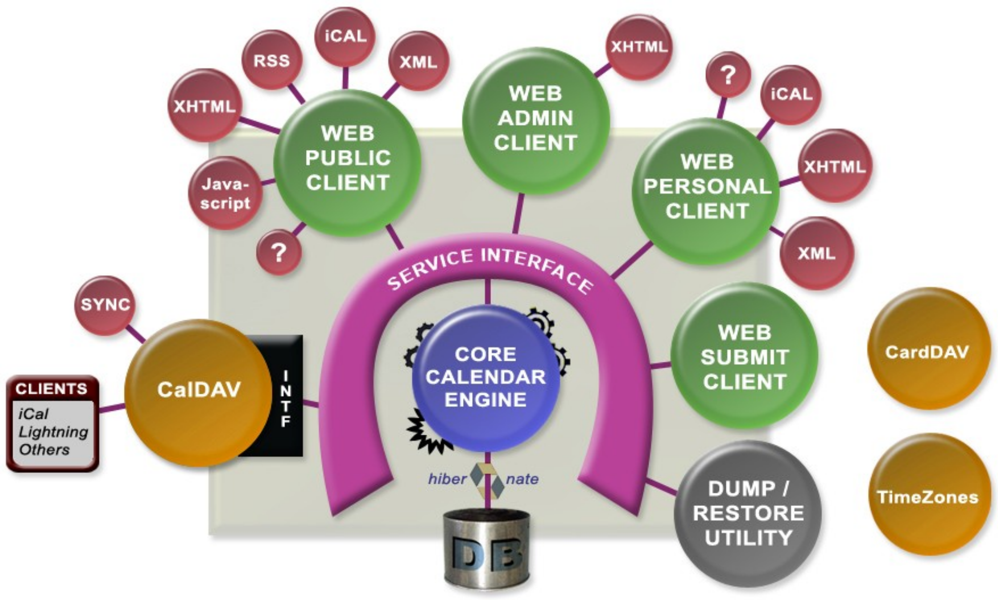
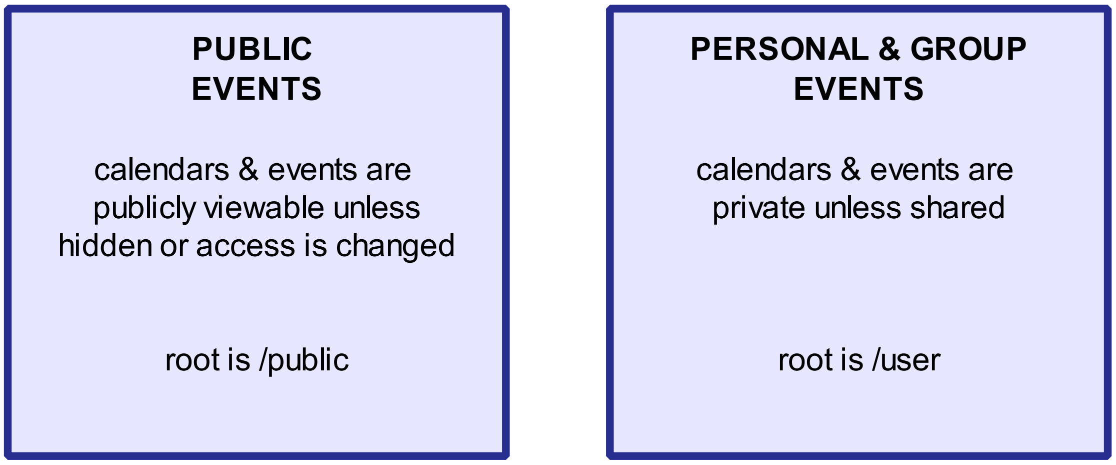


Illustration 1: Bedework System Architecture

- **Bedework core calendar engine**
- **Public events web client**, called a “Calendar Suite”, for display of public events
- **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** supporting contacts for scheduling in the personal client. This server will become more heavily used in future versions of Bedework for contacts, locations, and other resources.
- **TimeZone server** for support of timezone information
- **Dump/Restore command-line utilities** for database backup, initialization, and upgrades.

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 & 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 chapters 4 and 5.

Features of Bedework

- **Java :** Written completely in Java, Bedework is system independent. Currently it will compile and run in Java 1.5 or later.
- **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)
  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.
- **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.
  - **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 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 - XML & XSLT :** The web clients are XML and XSLT based allowing Bedework to be "skinned" for multiple clients and uses. For example, the quickstart comes with skins for producing production RSS, Javascript, and video feeds as well as HTML displays suitable for handheld devices.

- **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 and feeds can be filtered by these.
- **Internationalization :** Internationalization is carried out by creating a new skin. The

skin selected is based upon skin name and locale allowing a significant degree of multilanguage support in the client. Work is currently taking place to strengthen support for internationalization independent of Bedework skins.

- **RSS, Javascript, iCalendar Feeds :** Feeds can be filtered by category or creator.
- **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.

1.2 Bedework History / 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 the Java Architecture Special Interest Group (JA-SIG).

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". Like Bede is pronounced “bead”, Bedework is pronounced “bead work”.

1.3 Calendaring Standards & Interoperability

Interoperability with other calendar systems and clients is a core design principal of Bedework. Bedework's lead developers participate extensively (as members of Rensselaer Polytechnic Institute) in CalConnect, The Calendaring and Scheduling Consortium. For more information about calendaring and calendaring standards, please see http://www.calconnect.org/

Chapter 2 Getting Started

The Bedework Quickstart

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

The Bedework quickstart comes pre-built, allowing you to get familiar with the calendar quickly and easily, without having to compile and deploy code, and without having to set up a database. It is preloaded with data so you can see how the system looks in production. **It is the foundation from which a production release is built.**

Packaged with the quickstart

1. Bedework (version 3.4)
2. Tomcat 5.5 (apache-tomcat-5.5.17)
3. Hypersonic SQL 1.7 (hsqldb-1.7.3.3)
4. Apache Ant 1.6 (apache-ant-1.7)

Note to Windows users: replace all commands of the form "./[command]" with "[command]" or "[command].bat".

For example, to start Bedework, enter: "**bw -quickstart start**".

System requirements

1. JDK 1.5
2. JAVA_HOME environment variable must be set, for example:.
   Linux: export JAVA_HOME=/usr/java/jdk1.5.0_13
   Windows: set JAVA_HOME=C:\Program Files\Java\jdk1.5.0_13
3. Nothing else should be running on port 8080 and 8887.

Starting Bedework

1. Open a console window and cd to the quickstart directory.
2. Set JAVA_HOME environment variable.
3. Enter the following command: ./**bw -quickstart start**
   This will start Hypersonic, Tomcat, and the Apache DS ldap directory.
4. Access the web applications at http://localhost:8080/bedework
   Note: this link will only work on the system on which the quickstart is running.

5. When finished, you can stop Bedework by entering **CTRL-C** in the window in which the command was started.

Logging

Log messages largely appear in the tomcat log in <tomcat-dir>/logs. This release uses log4j for most of the application logging. The distributed log4j.xml appears in calendar/resources in the quickstart and in common/classes in the tomcat directory. It is configured to maintain a rolling log file, **<tomcat-dir>/logs/server.log**, and also append output to the console.

In addition, a socket appender is defined which allows Apache Chainsaw to be used to watch the log output.

Building the Quickstart release

1. Open a console window and cd to the quickstart directory.
2. Set JAVA_HOME environment variable.
3. Enter one of the following commands:
   ./bw -quickstart deploy
   ./bw -quickstart deploy.debug
   ./bw -quickstart clean.deploy.debug

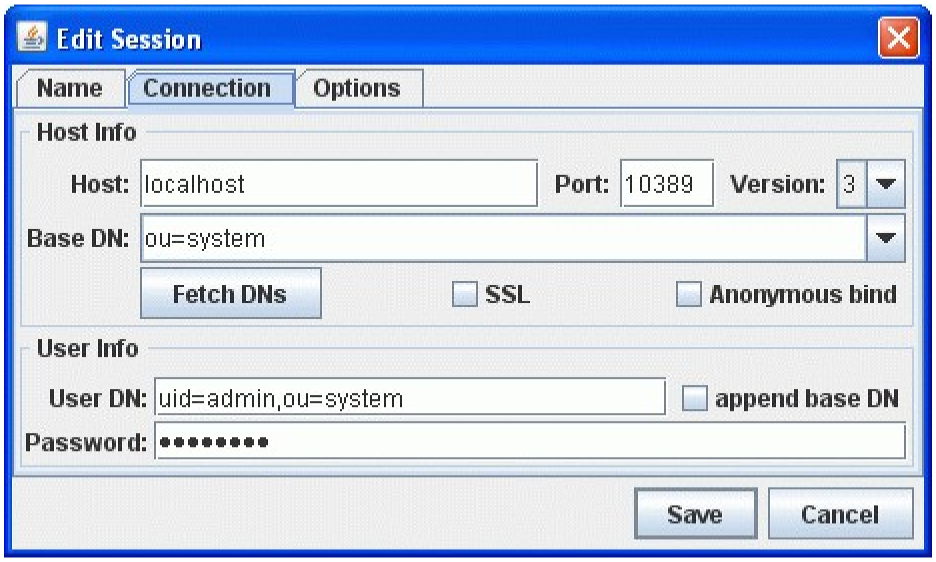
Creating user accounts

To add user accounts to the quickstart ldap directory:

1. Open a console window and cd to the quickstart directory.
2. Set JAVA_HOME environment variable
3. Follow the steps above to build Bedework (this will download the necessary libraries)
4. Enter one of the following commands (Linux or Windows):
   bedework/build/quickstart/linux/adduser userid Firstname Lastname userid@mysite.edu password
   or
   bedework\build\quickstart\windows\adduser userid Firstname Lastname userid@mysite.edu password

Accessing the Apache DS Ldap Server

With Bedework running or with Apache DS running stand-alone, you can connect to the Apache DS server using the following setup (password = "secret"):



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

PASSWORD: secret

Dumping and restoring the database

See also <chapter and section> for detailed information about using Bedework's dump/restore utility.

Prior to deploying a production database, you can work with Bedework's Hypersonic distribution to dump and restore data, back up your work, and get a feel for working with Bedework data. Note that you will use the same dump/restore utility when you have deployed your own production database (e.g. MySQL, Oracle).

To re-initialize Hypersonic (hsql) with default Quickstart data:
The following steps will initialize the db with the default data shipped with the quickstart release. The data is found in bedework/dist/temp/shellscr/dumpres/data/initbedework.xml. This data contains a the basic calendar structure, many categories, two calendar suites, and the handful of users & groups that allow the quickstart demonstration to run. It is a reasonable starting point for most deployments.

1. Open two console windows and cd to the quickstart directory in both.
2. Set JAVA_HOME environment variable.
3. Stop the Bedework system if running.
4. **rm -r hsqldb-1.7.3.3\demo** *(to remove the hsql schema)*
5. Start the system in console window 1:
   ./bw -quickstart start
6. In console window 2:

cd bedework/dist/temp/shellscr/dumpres *(note: if this does not exist, follow the commands to Build the Quickstart release.)*

7. **./bwrun schema-export** *(to create the db tables)*
8. **./bwrun initdb** *(to initialize the db with data/initbedework.xml)*

To dump or restore your own data files:

1. Open two console windows and cd to the quickstart directory in both.
2. Set JAVA_HOME environment variable.
3. Stop the Bedework system if running.
4. Start the system in console window 1:
   ./**bw -quickstart start**
5. In console window 2:
   cd bedework/dist/temp/shellscr/dumpres *(note: if this does not exist, follow the commands to Build the Quickstart release.)*
6. to dump: ./**bwrun dump [filename].xml**
   to restore: ./**bwrun restore [filename].xml [-indexroot <lucene-index-root>]**

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 which can be found at bedework/dist/temp/shellscr/dumpres/data/initbedework-sparse.xml

Chapter 3 Deploying Bedework

3.1 Prerequisites

This section describes setting up your build environment and provides guidance on creating your initial calendars, subscriptions, views, and administrative users.

Install the quickstart and test your environment

Before attempting any customization, please test your environment by running the quickstart release. 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.

Requirements

- JDK 1.5 – specifically, Java 1.5.0_11 or later
- JAVA_HOME environment variable must be set
- Hardware for testing: most current desktop or laptops will be adequate.
- Hardware for production: A server class machine generally with at least 1Gig allocated to the jvm. We recommend 2GB (see settings below).
- An adequate database system. In particular, the database probably has to support Unicode. Many Bedework installations use MySQL or Oracle.
- To implement the Bedework calendaring system, it is useful to understand the following:
    - Java servlets: http://java.sun.com/products/servlet/docs.html
    - Servlet containers (e.g. Tomcat, JBoss)
    - Authentication is local to your site - some Java programming may be necessary to accomplish this.

Supported databases.

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:

- Hsql – the database we provide with the quickstart. It is unclear how safe it is to use as a production database. It appears to be used by various systems to provide a persistence mechanism for messages (e.g. Jboss does this) so it might be usable on a small scale.
- MySql version 5
- Oracle Version 10 and perhaps Version 9 with Version 10 jdbc drivers.

Unsupported databases

Because of our use of 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 most of these 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.
- PostgreSQL: We hope to do more work in the near future supporting postgres.
- MySQL version 4: Has problems in a number of areas. These are unlikely to ever be resolved. Version 4 is now old.

3.2 Prepare a localized version of Bedework

Prepare your build and runtime properties

Bedework uses ant for the build and deploy process and a number of property files are used to control that process. Ant properties have the characteristics that once set they cannot be modified. To override default settings they must be set earlier in the process.

Do not change the original files. Only edit the configurations you copied into your home directory. Keep the distributed versions for reference..

Copy the distributed configurations to your home directory
Bedework uses a configuration directory to store multiple named configurations. The quickstart copy of this directory is at

```
<quickstartDirectory>/bedework/config/bwbuild
```

Copy the entire directory structure into your home directory. Your home directory in unix is usually

```
/home/userid
```

And in windows is typically

```
C:\Documents and Settings\userid
```

So, for example, the bwbuild directory in windows would live here:

```
C:\Documents and Settings\userid\bwbuild
```

Inside ***bwbuild*** you will find a number of example configurations in subdirectories, for example ***default, mysql,*** and ***jboss***. These configurations are used by the ***bw*** script using the ***-bwc*** parameter, for example:

```
./bw -bwc jboss deploy.debug
```

would build the quickstart with the provided jboss configuration, and

```
./bw -bwc mysql start
```

will start the quickstart with the mysql configuration.

If no configuration is named, then the configuration named ***default*** will be used.

In each configuration you will find the following files (the default configuration is used in this example):

```
bwbuild/default/build.properties
bwbuild/default/cal.options.xml
bwbuild/default/cal.properties
bwbuild/default/carddav.options.xml
bwbuild/default/context.xml
```

A ***build.properties*** file must be inside each configuration directory and provides links to the other files and should not need any changes.

You can set properties in this file which will override the default settings. In particular you can tell the build system the location of the configuration you want to build, which is our next step:

The ***cal.properties*** file is used only for the build and deployment process while the ***cal.options.xml*** file is for runtime properties and is included in the resulting applications.

The ***context.xml*** file is referenced by the properties file and is where you place tomcat settings for the database resource.

The properties file

The ***cal.properties*** file is divided into sections with different property prefixes.

Install

The section prefixed "*org.bedework.install*" defines which applications are to be installed. This consists of a list of application names.

For each name there should be a corresponding section prefixed with "org.bedework.app.<name>" and also a corresponding section in the options file.

The default configuration comes with a number of application configurations. When creating a configuration it is appropriate to simplify by removing unneeded applications. Remove the name from the install property and delete the appropriate section.

Applications in the default configuration are:

- **CalAdmin** – the administration application
- **Events** – public events (unauthenticated access to calendars)
- **SoEDept** – example calendar suite
- **UserCal** – personal events
- **Pubcaldav** – public events caldav server. Provides unauthenticated access to calendars.
- **Usercaldav** - personal caldav server. Provides authenticated access to calendars.
- **dumpres** – the dump restore utility
- **caldavTest** – a test application for caldav
- **test** – a test suite

- **restoreFrom2p3px** – the dump restore utility configured to restore from uwcalendar 2.3.x
- **bwconfig** – (not in the list but a configuration section is present). This is a try at a web based configuration application (not currently in use and likely to be deprecated).

Global
The section prefixed "*org.bedework.global*" defines properties global to the whole deployment process.

Application
The section with properties prefixed "*org.bedework.app.<name>*" are the application deployment properties, one section per named application.

Two properties define the project and type of application. The value of the property "*org.bedework.app.<name>.project*" defines which project the application is a part of. Currently these can be

- "caldav" - a caldav server
- "caldavTest" - a caldav test package
- "webapps" - a web client
- "dumprestore" - a dump/restore application
- "freebusy" - the freebusy aggregator

The value of the property "org.bedework.app.<name>.type" corresponds to the name of a subdirectory in bedework/deployment, e.g. webpublic, webadmin, etc. So to define the administrative client named CalAdmin of type webadmin we have the fragments:

```
org.bedework.install.app.names=...,CalAdmin,...
...
org.bedework.app.CalAdmin.project=webapps
org.bedework.app.CalAdmin.type=webadmin
```

Multiple versions of each application type may be deployed, each configured differently. This is of importance for calendar suites (departmental calendars).

The options file.

This ***cal.options.xml*** file contains run time properties and is divided into sections much like the 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.

It is important to set the “system” properties for a new Bedework installation. These are found in the “syspars” section of the .options.xml file. A number of options can be left with the default values and some are not yet implemented. The two values it is particularly important to set (and their default settings) are:

```
<tzid>America/New_York</tzid>
<systemid>demobedework@cal.mysite.edu</systemid>
```

These are described as follows:

The tzid property
The tzid is the default timezone to be used for times and dates.

The systemid property
The systemid is used when generating uids for calendar entities. This name should be related to your organization for ease of identification. If you run multiple Bedework systems, thid value should be different for each. In addition, it 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).

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

```
mailto:testuser01@cal.mysite.edu
```

Other properties
There are also a number of names used when creating default calendars. These can be set to some appropriate localized value, **but note:** the xsl stylesheets of Bedework 3.5 and earlier versions make reference to one or two of these properties by name, in particular the string “user” for the name of the user calendar root. If you change this value, you must update the xslt files that reference it. This will be corrected in version 3.6.

The size settings are mostly unused at the moment.

The property

```
<userauthClass>org.bedework.calcore.hibernate.UserAuthUWDbImpl
</userauthClass>
```

defines which class handles administrative groups for the administrative client. The group class setting are explained in the “Access rights and groups” section below.

3.3 Set up a production database

The quickstart is set up to use hsqldb and this provides a good way to try out the system. To move to another database system you will need to configure the build to affect the following:

- Configure tomcat or each context to use another database
- Configure Hibernate to use the appropriate dialect
- Rebuild
- Configure the dump/restore

Note: if you'd like to set up an initialized database in hsqldb for testing, you can skip to “Build Schema and Initialize” on page 10 and just run “./bwrun schema-export” followed by “.bwrun initdb” against an empty hsql database. To create this, stop hypersonic (if running) and rename (or throw away) the directory named <hsqldb-dir>/demo. Restart hsql and the directory will be recreated in an empty state. See Build Schema and Initialise for more information about the schema-export and initdb commands.

Language settings.

Your database needs to support at least unicode. For most European and American systems this may not be obvious at first but eventually problems will occur with certain special characters. For most non-European languages unicode or some other multibyte support is an absolute necessity.

Each database has its own settings for language support. Many have built in traps for the unwary (most of us). For example, in Postgres UNICODE apparently means UTF8. This is incorrect for Chinese for example.

In addition, 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 obviously needed. Those components are at least:

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

- Jdbc settings
- The database

Configure your applications context.xml

Tomcat 5.5.x applications are configured by setting up context.xml files for each application.

The ***context.xml*** file located in the configuration look something like this:

```
<Context path="@CONTEXT-ROOT@" reloadable="false">
 <Resource name="jdbc/calDB" auth="Container"
           type="javax.sql.DataSource"
           driverClassName="org.hsqldb.jdbcDriver"
           url="jdbc:hsqldb:hsql://localhost:8887"
           username="sa"
           password=""
           maxActive="8"
           maxIdle="4"
           maxWait="-1"
           defaultAutoCommit="false" />

  <!-- Disables restart persistence of sessions -->
  <Manager pathname=""/>
</Context>
```

At the very least, set the URL to your database, and set the username and password with which you will connect.

The “@CONTEXT-ROOT@“ is an example of a token which will be replaced during deployment by a value from the properties file. If you have the above token in your file it will be replaced with the correct context root.

The reference to this context.xml file is in the cal.properties file and looks something like:

```
...
org.bedework.app.Events.tomcat.context.xml=${env.BEDEWORK_CONFIG}/context.xml
```

This can be changed if you need more than one context.xml file or wish to locate it outside of the configuration directory:

```
...
org.bedework.app.Events.tomcat.context.xml=<home>/mycontexts/context.xml
```

```
...
```

Configure Tomcat 5.5.x

With the above configuration you do not need to configure a database in tomcat's server.xml. However some changes are needed. These are:

- Allow no roles
- Add jdbc drivers
- Allow directory browsing
- Jvm parameters

Allow no roles

Tomcat's handling of security constraints was at some point changed to be absolutely compliant with the servlet specification. Unfortunately, in this regard at least, the specification is somewhat impractical. Within the web.xml security-constraint element is a role-name element which is set to "*" as in this example:

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>CalendarAdmin</web-resource-name>
    <description>Public events Administration</description>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint> ...
```

The entry

```
<role-name>*</role-name>
```

used to mean ANY role. It now means by default any role defined in the web.xml. There is no way defined in the servlet specification to have security constraints without roles.

However, it appears that the behavior is configurable in tomcat. In the server.xml there is the definition of the Realm used for authentication. Add a setting for the allRolesMode attribute, in the quickstart it will look like

```
<Realm className="org.apache.catalina.realm.UserDatabaseRealm"
             resourceName="UserDatabase"
             allRolesMode="authOnly" />
```

and you should be able to login without any role assigned.

Add Jdbc drivers

1. Copy the appropriate jdbc driver .jar file for the database you are deploying to the <tomcat>/common/lib directory. For example, the quickstart comes packaged with the Hypersonic SQL driver: <quickstart>/apache-tomcat-5.5.17/common/lib/hsqldb-1.7.3.3.jar .

2. ~~Copy the same driver jar into <userhome>/bwbuild/<config>/lib/jdbc/ so that it can be built into the dump/restore utility you will use to initialize the database. (If the lib/jdbc directory doesn't exist, create it.)~~ **Please note: this step does not currently work. Instead:** drop the jar file into <quickstart>/dumpres/lib/ after it is built and unzipped. *(See "Build the schema and initialize using the Dump / Restore Utility" on page 12 below).*

You can find the drivers associated with the database version you are running on the database's web site. MySQL and Oracle drivers, for example, can be found at http://www.mysql.com/products/connector/ and http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html respectively.

Allow directory browsing

This change is only required of you are using tomcat to host your stylesheets. Our recommendation is to host them on your public web server. If you do wish to host them on tomcat you must decide whether you want directory browsing enabled or disabled. Enabled allows Bedework to discover the stylesheets by checking for the existence of directories. Otherwise marker files must be created. The tomcat default is to disallow directory browsing. If you wish to allow browsing open the file conf/web.xml and find the setting for the listing property:

```
<init-param>
    <param-name>listings</param-name>
    <param-value>false</param-value>
</init-param>
```

Change the value "false" to "true" to allow browsing of directories.

Uri encoding

By default tomcat uses ISO-8859-1. If you want to use Tomcat with UTF-8, you must add the following parameter in your server.xml file: **URIEncoding="UTF-8"**.

```
<Connector port="8080" maxHttpHeaderSize="8192"
           maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
           enableLookups="false" redirectPort="8443"
           acceptCount="100"
           connectionTimeout="20000" disableUploadTimeout="true"
           URIEncoding="UTF-8" />
```

JVM parameters

The quickstart jvm parameters may be adequate for small scale testing or development. For production use you will certainly need to modify the JVM parameters. Below is a sample set of options (from a jboss configuration).

```
JAVA_OPTS="$JAVA_OPTS -server -Xms1792m -Xmx1792m"
JAVA_OPTS="$JAVA_OPTS -XX:NewSize=512m -XX:MaxNewSize=512m”
JAVA_OPTS="$JAVA_OPTS -XX:SurvivorRatio=2"
JAVA_OPTS="$JAVA_OPTS -XX:PermSize=128m -XX:MaxPermSize=128m"
JAVA_OPTS="$JAVA_OPTS -XX:+UseParallelGC -XX:+UseAdaptiveSizePolicy”
JAVA_OPTS="$JAVA_OPTS -XX:+AggressiveHeap"
```

Configure Hibernate

Setting your SQL dialect

Configuring Hibernate to use the appropriate dialect is done in **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.MySQL5Dialec
t
```

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

http://www.hibernate.org/hib_docs/v3/reference/en/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).

Before building the system, if you are building for a database other than the quickstart hsqldb, you will need to make the appropriate jdbc drivers available. Place the driver jar file in the directory bedework/lib/jdbc and it will appear on the class path for the schema and the dump and restore applications.

You will also need to place this jar file in the tomcat common/lib directory so that tomcat can access the database.

Build the newly configured system

With all that in place it is now possible to build yourself a system using your configuration. Assuming you chose to update the default configuration then the build command would be

```
./bw clean.deploy.debug
```

If you used the mysql configuration then the command would be

```
./bw -bwc mysql clean.deploy.debug
```

If you created a new configuration named, for example, “prod” then the command would be

```
./bw -bwc prod clean.deploy.debug
```

This will create a number of WAR files in **<bedwork>/dist/** including for example:

```
cal.war, caladmin.war, and ucal.war
```

If you are using the Tomcat directory within the quickstart distribution, your application war files are now deployed. Otherwise, collect the war files and drop them in your container.

Note: ".debug" will provide debugging output in the server log. You can remove it and rebuild when you are convinced you are production ready.)

Build the schema and initialize using the Dump / Restore Utility

Now create a schema and initialize the database. The deploy process created a zip file called bedework/dist/dumpres.zip which can be unzipped to run the schema build. This is Bedework's dump/restore utility. The unzipped utility contains a Unix shell script named

bwrun.sh (in Windows, ***bwrun.bat***) that you will use to dump and restore the database. This file is useful for creating backups and essential for performing upgrades between Bedework versions.

Unzip that file to a convenient location and in linux ensure the file called ***bwrun.sh*** is executable (**chmod +x bwrun.sh**). This manual generally assumes you have unzipped the folder to the root of the quickstart directory, <quickstart>/dumpres/.

Drop a copy of the database's jdbc driver jar file into <quickstart>/dumpres/lib/

The Hibernate schema tool can create all the tables and constraints directly in the target database. To do this, make sure Bedework is running, then enter:

```
./bwrun schema-export
```

The tables will be created in your database (a file named "schema.sql" is also created). This process is non-destructive, so if you wish to start with a clean database, you will need to manually drop your tables. A word of warning here; while the schema-export option is useful there is little or no error checking taking place. Check back through the output for errors. A logfile is written and may be easier to check through.

If you wish only to create a schema file for your system, enter:

```
./bwrun schema
```

This produces a file named "schema.sql".

To initialize an empty database use initdb, e.g.

```
./bwrun initdb -ndebug -indexroot path-for-lucene
```

This uses an initial data file wrapped up with the dump restore application (data/initbedework.xml). *See also the "restore" target below to restore a different file such as <quickstart>/bedework/projects/dumprestore/resources/inibedework-sparse.xml which contains only the minimum data to run the system.*

The optional parameter -ndebug will turn off debugging output. This makes things faster for large restores.

The indexroot parameter is used to tell the restore where your lucene indexes should be built. In the quickstart this is a relative path because the quickstart might be restored anywhere, but in general this should be an absolute path.

For testing, a directory in your home directory might be appropriate, for production use perhaps a /data directory, e.g. /data/calendar-3.3.1/lucene/indexes

This directory must be destroyed and recreated before restoring the data.

Note: initbedework.xml creates a single super-user "admin". However, there is a potential problem here if you are trying to bring yourself up in your local user address space; you need an administrator that can log into the admin client. To simplify this, we will probably add a property to set the administrative user you wish to use in future releases.

To dump the database data use

```
./bwrun dump <filename>
```

The application can be used to produce a regular nightly xml dump of the data. In future releases the dump/restore will be reformatted to facilitate restoration of a single users data.

To restore the data use

```
./bwrun restore <filename> -ndebug -indexroot path-for-lucene
```

where <filename> is the name of a dump file produced by the dump process. The tables must be empty to restore the data.

The indexroot parameter is needed if you want to change the one set in the data. Remember to empty (or destroy and recreate) the directory.

3.4 Authentication

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

Authentication can be managed by the servlet container in a number of ways which are currently beyond the scope of this document. The authentication method used by tomcat is defined in <tomcat>/conf/server.xml and in the quickstart is configured to use the packaged Apache Directory Server. The tomcat website provides details on configuring tomcat to use other forms of authentication, including active directory, a local file or databases. (see also Tomcat SSL HowTo).

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

3.5 Build and deploy

Build the calendar with the command:

```
./bw clean.deploy.debug
for the default config, or
./bw -bwc configname clean.deploy.debug
```

(".debug" will provide debugging output in the server log. You can remove it and rebuild when you are convinced you are production ready.)

```
./bw clean.deploy
or
../bw -bwc configname clean.deploy
```

3.6 Add a super user

At some point you will switch from the quickstart authentication to your sites authentication. Before this happens you need to ensure you have an administrative super user that exists within your own authentication domain.

Two ways to achieve this are to:

1. Create a user in your domain, e.g. caladmin, that is already set up as a superuser in the Bedework quickstart.
2. Create a Bedework superuser with an account that already exists in your authentication domain.

Option 2 is probably the most appropriate and as the deployer it is probably appropriate to use your own account, at least initially.

3.7 Add administrative groups and users

In the quickstart 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)

1. Within the /caladmin application, select "Admin Groups: Add"
2. Give the group a name, a description, provide a userId for the owner, and set the "Events Owner" to agrp_*groupName*, or something meaningful that will will easily identify which group the event belongs to. (Note: you should leave the prefix (e.g.

"agrp_") on event owners; the prefix is defined in the properties file as the property org.bedework.app.CalAdmin.admingroupsidprefix)

3. Once the group is added, add members by providing user ids. (These should map to those used to authenticate to the administrative client.)

3.8 Create initial public calendar aliases

- Login to the /caladmin application as super user.
- From the "Sytem" tab, select "Manage calendars & folders". The default public calendar tree is divided into three sections: aliases, cals, and unbrowsable.
- Add folders and aliases in the /public/aliases branch to provide a default set of curated event groupings. The aliases are available to all calendar suites for subscription.
- For more information about architecting your public tree, see chapter 6, "Public Events Calendaring".

3.9 Add Calendar Suites

Bedework's web views of *public event information* are called "Calendar Suites". You cannot run a public client unless an associated calendar suite has been defined in the admin client.

The quickstart release comes with a default calendar suite called MainCampus which is appropriate for use as the first public calendar suite in a production environment.

For more information see chapter 6.5 "Managing Calendar Suites."

3.10 Access rights and groups

Access to resources in Bedework is controlled by an access control system based on WebDAV and CalDAV. A user's access is based on their identity and group membership. The system allows a different group structure for administrative and user access. This ensures that a given user, when logged in to the user client, does not have any special administrative rights which may lead to unfortunate consequences, such as deleting a public calendar by mistake, or adding private events to public calendars.

Two system parameters determine this behavior, the initial values are set in the xml options file, democal.options.xml in the quickstart. The relevant elements in the syspars section are:

```
    <admingroupsClass>org.bedework.calcore.hibernate.AdminGroupsDbImpl
```

```
      </admingroupsClass>
      <!--
      <usergroupsClass>org.bedework.calcore.ldap.UserGroupsLdapImpl
      </usergroupsClass>
       -->
       <usergroupsClass>org.bedework.calcore.hibernate.GroupsDbImpl
       </usergroupsClass>
```

Note the second of the three is commented out. These settings define which class will be used to manage groups for the administrative and user clients. The first class is an implementation which uses a the Bedework database to store the administrative groups.

The last setting is a dummy class which does nothing but which could use the database to allow testing or perhaps even for sites which don't want to use a site-wide directory.

The commented out setting is a class which uses ldap to determine group membership. In the options file is a section labeled "user-ldap-group" which configures this class. This class should also be usable with Active Directory.

Using ldap for groups and account validation

Authentication is performed outside of Bedework, either by the container (e.g. tomcat) or by a filter mechanism or some other approach.

However, Bedework still requires access to directory services of some kind to determine group membership and to check whether accounts are valid. As indicated above, a class is available to perform group lookup and account validation using an ldap directory.

The current implementation of this class assumes the directory is readable. In production this is unlikely to be adequately secure so we will be working on changes to improve the security. One possibility is to define a custom resource in the container to provide access to the naming context. That will allow some of the authentication parameters to be moved out into the container configuration.

In addition the class named above may not carry out all the checks appropriate for your organization. In paticualr you may want to implement a class that overrides the validUser method. The default carries out no checks with your system directory. A deployed version should probably check the directory to ensure the user does exist.

3.11 Setting up the Bedework Themes

Copy the Bedework themes (templates)

You should now make a copy of the template directories for theming. While they can be left in Tomcat, we recommend placing them on a web server accessible to your web designers. Placing the stylesheets and resources on a separate web server (such as your primary web server) will make them significantly more convenient to access and manipulate.

The directories to copy 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
```

The destination where you copy these directories is a URL specified in the cal.properties file by the property: “app.<name>.cal.suite” for the public calendar suites and “app.<name>.root” for the other web applications.

Given the values in the properties file:

```
org.bedework.app.UserCal.root=http://somewebserver/bedework-3.5/ucalrsrc
org.bedework.app.CalAdmin.root=http://somewebserver/bedework-
3.5/caladminrsrc
org.bedework.app.Events.root=http://somewebserver/bedework-3.5/calrsrc
org.bedework.app.Events.cal.suite=MainCampus
org.bedework.app.SoeDept.root=http://somewebserver/bedework-3.5/calrsrc
org.bedework.app.SoeDept.cal.suite=SoeDept
```

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-3.5/calrsrc.MainCampus ; the “SoeDept” calendar suite at http://somewebserver/bedework-3.5/calrsrc.SoeDept

Once you have set the above property values, copy the values into the <approot> elements in the options.xml file.

HTTPS and mixed content messages

If you choose to serve a client over https, you will need to specify the same protocol for the approot to avoid mixed content messages.

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 .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 Chapter 4.

Rebuild / Redeploy Bedework
To pull in the theme configuration changes, rebuild Bedework:

```
./bw clean.deploy.debug
for the default config, or
./bw -bwc configname clean.deploy.debug
```

3.12 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?

The dump/restore utility is the tool to use. The intent is that any given version of the restore process will be able to read dump files produced by earlier versions.

Currently we expect to be able to read data dumped from UWCalendar 2.3.2 (with slight massaging) and any version of Bedework. Later on we intend dropping UWCalendar support but it will still be possible to migrate through a two step process.

3.13 Performance and tuning

Hibernate and caching
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.

Since release 3.4.1 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 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.

3.14 Access Control

An overview

Bedework has an access control system based in large part on the WebDAV ACL specification [RFC3744] with CalDAV extensions. Access rights are inherited down the tree of folders, calendars and calendar entities and can be given to or denied from users and groups and other principals.

Setting access rights is complex and difficult, even for apparently simple cases. For that reason later versions of Bedework will incorporate 'wizard' style tools to help users set access based on their intentions rather than requiring them to determine the appropriate set of ACEs.

Definitions

Principals

A principal represents an entity which might need to be denied or given access or with which we might want to interact. For example, user principals usually represent real people and group principals represent groups. Other principals might be host principals, ticket principals or resource principals.

A resource principal might be, for example, a room allowing us to invite a room to a meeting, thus booking that room.

ACE
An ACE is an Access Control Entry. This defines grant or deny access for a single principal. It consists of a principal type, the principal itself – perhaps in a modified form and the denied or granted access rights. Special principal types include *unauthenticated*, *authenticated* and *others*.

ACL
An Access Control List. This is a list of ACEs which together define the access for a given entity. Internally only the ACEs set explicitly for an entity are stored with that entity. The full ACL consists of all inherited ACEs together with any explicit ACEs.

Scheduling and freebusy access

Access control for scheduling is based on CalDAV calendar access and CalDAV scheduling.

CalDAV introduced the **read-freebusy** privilege which allows principals to read freebusy information for the resource with which it is associated.

CalDAV scheduling introduces further privileges which are associated with a principal, not with any specific resource, and are actually attached to the users inbox. The privileges are **scheduling** which includes

- **schedule request**: allow (named principals) to request meetings
- **schedule reply**: allow (named principals) to reply to meeting requests
- **schedule freebusy** - allow (named principals) to send freebusy requests

If user *douglm* sets the schedule privilege on the inbox for say user *testuser01* then user *testuser01* can send and reply to meeting requests and see *douglm*'s freebusy through the web ui and the caldav server.

NOTE – this is important, in addition to those privileges, *douglm* must set **read-freebusy** on those calendars that he wants *testuser01* to access for freebusy info.

The reasoning is that we may want to block or allow scheduling operations as a whole, but if we allow them we may want to provide a different view of our freebusy to different users.

So if I as *douglm* have calendars "*work*" and "*athome*" I might not allow freebusy access to my "*athome*" calendar to colleagues because I don't want them to see how empty my home-life is.

Alternatively I might disallow access to "*work*" to my wife for the opposite reason.

Simplifying:
It is possible to simply access for most users by setting access at the **/user** level, as this will be inherited by all subfolders and calendars.

What you set depends upon what you want as an organization.

The simplest approach is to add the following access to **/user**

```
authenticated: schedule, read-freebusy
```

which should allow any authenticated user to schedule and reply to meeting requests and see anybody elses freebusy.

The administrative interface currently doesn't display **/user** in the manage calendars page but you can use a url like:

```
http://localhost:8080/caladmin/calendar/fetchForUpdate.do?calPath=/user
```

A little more open is to allow that access for others INSTEAD of authenticated - that would allow scheduling via CalDAV form anywhere.

3.15 TUTORIAL: setting up Bedework to work with MySQL & LDAP

The following tutorial illustrates a specific example of setting up Bedework to use a MySQL database and local LDAP authentication. This walkthrough assumes a Linux operating system and uses the Tomcat packaged with the quickstart.

To get the most out of this concrete example, please read through Chapter 2 "Getting Started" and all of Chapter 3 to this point.

Note to Windows users: replace all commands of the form "./[command]" with "[command]" or "[command].bat".

For example, to start Bedework, enter: "**bw -quickstart start**".

1. Download the quickstart release from http://www.bedework.org/bedework/update.do?artcenterkey=2

2. Log into a console window and set JAVA_HOME.
   E.g. "export JAVA_HOME=/usr/java/jdk1.5.0_13" (in windows: "set JAVA_HOME=C:\Program Files\Java\jdk1.5.0_13")

3. Unzip the quickstart, and run it to demonstrate that it is working as expected:

```
cd <quickstartDirectory>
./bw -quickstart start
```

4. Copy the configurations directory to your home directory:

```
cp <quickstartDirectory>/bedework/config/bwbuild /home/<userid>/
```

5. Verify that you have an instance of MySQL running on your system, e.g. version 5.1.
   a) Login to MySQL as root and create a database:

```
create database bedework3p5
```

   a) Grant privileges on the table to a user of your choice:

```
grant all on bedework3p5 to bedework identified by 'somepassword';
```

6. Modify the example mysql config provided with the quickstart:
   a) Modify /home/<userid>/bwbuild/mysql/**cal.properties**:
   - Remove from line 19 any applications you don't want to build org.bedework.install.app.names=<apps>
   - On lines 34-36, set the URL, username, and password for the database created in step 5 (you will do this again in the next step). E.g. the URL would be: org.bedework.global.jdbcurl=jdbc:mysql://127.0.0.1:3306/bedework3p5 (assuming MySQL is running on the same server – otherwise adjust the URL appropriately).

   b) Configure your connection to the MySQL datasource (again) in /home/<userid>/bwbuild/mysql/**context.xml**
   - set the url to point at your database, e.g. url="jdbc:mysql://127.0.0.1:3306/bedework3p5"
   - set the username and password to that of the user created in step 5.

   c) Modify /home/<userid>/bwbuild/mysql/**cal.options.xml**:
   - Set the default tzid (timezone id) and the systemid on lines 110 & 111, e.g. <tzid>America/New York</tzid> and <systemid>calendar@mysite.edu</systemid>

7. Build the system:

```
./bw -bwc mysql clean.deploy.debug
```

8. Unzip and prepare the newly built dump/restore utility:

```
cd <quickstartDirectory>
unzip bedework/dist/dumpres.zip
cd dumpres
chmod +x bwrun
```

9. Download the JDBC Driver for MySQL (Connector/J). Unzip the file and place the jar file from the package (e.g. mysql-connector-java-5.1.8.bin.jar) in both of these folders:

   a) <quickstart>/apache-tomcat-5.5.17/common/lib/

   b) <quickstart>/dumpres/lib/

10. Use the dump/restore utility to build the database tables and initialize the database.

    a) Create the tables:

```
cd <quickstart>/dumpres
./bwrun schema-export
```

    a) Initialize the database and set the path to the Lucene indexes with either (but not both) of these commands:

    - initialize with the default quickstart data:

```
./bwrun initdb -indexroot <someDirectoryPath>/bedeworkIndexes
```

    - `OR initialize with the bare minimum of data:`

```
./bwrun restore
<quickstart>/bedework/projects/dumprestore/resources/initbedework-
sparse.xml -indexroot <someDirectoryPath>/bedeworkIndexes
```

11. Verify that Bedework starts up and that the web clients are working as expected: (note the use of the -bwc parameter rather than -quickstart)

```
cd <quickstartDirectory>
./bw -bwc mysql start
```

12. Use your local LDAP directory for authentication:

    a) FIRST, login to the Bedework admin client and add a superuser that exists in your local LDAP directory:

    - login as admin
    - click the “System” tab
    - select “Manage System Preferences”
    - In the third row, add a username (probably yourself) to the comma separated list of

superusers.

b) Configure Tomcat to point at your local LDAP server:
   - Edit <quickstart>/apache-tomcat-5.5.17/conf/server.xml
   - Change the jndi realm on line 169 to point at your local LDAP server

You should now have a production-ready Bedework system running.

A few common things not covered in this tutorial:

- Copying war files into and configuring settings for an existing Tomcat distribution.
- Setting JVM parameters in Tomcat (see section 3.3) – these are not set to production values in the quickstart distribution, so you will likely want to change them.
- Moving the Bedework themes to a different web server (see section 3.11) – you do not have to do this, but it will probably make theming easier.
- Defining virtual hosts for the applications (done in the cal.properties file)

Chapter 4 Public Events Calendaring

4.1 Planning your Bedework public events system

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, feeds & subscriptions, and direct downloads. External calendar subscriptions can be aggregated with the central event pool, and users can pull filtered feeds of events (e.g. ical feeds, rss, caldav, json).

Each group in your organization that needs to publish events is represented by an administrative group. Members within the same group can see and edit each other's events in the administrative web client. The first step in establishing a Bedework public events system is to identify which groups within your organization should be represented by administrative groups and how fine grained the groups should be.

Calendar suites control how events are tagged and filtered. Each administrative group works within the context of a calendar suite, and the second step in establishing a Bedework public events system is building your calendar suites.

4.2 Definitions

- **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**: 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

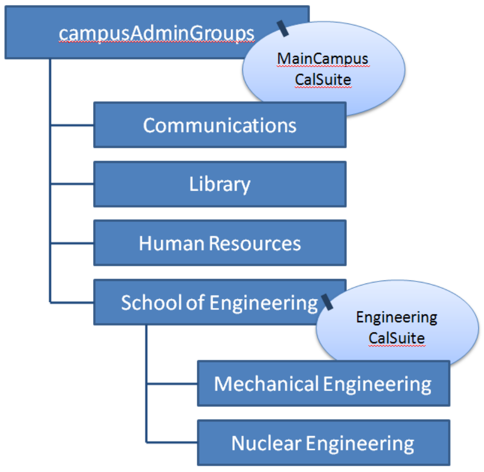
category.

- **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.

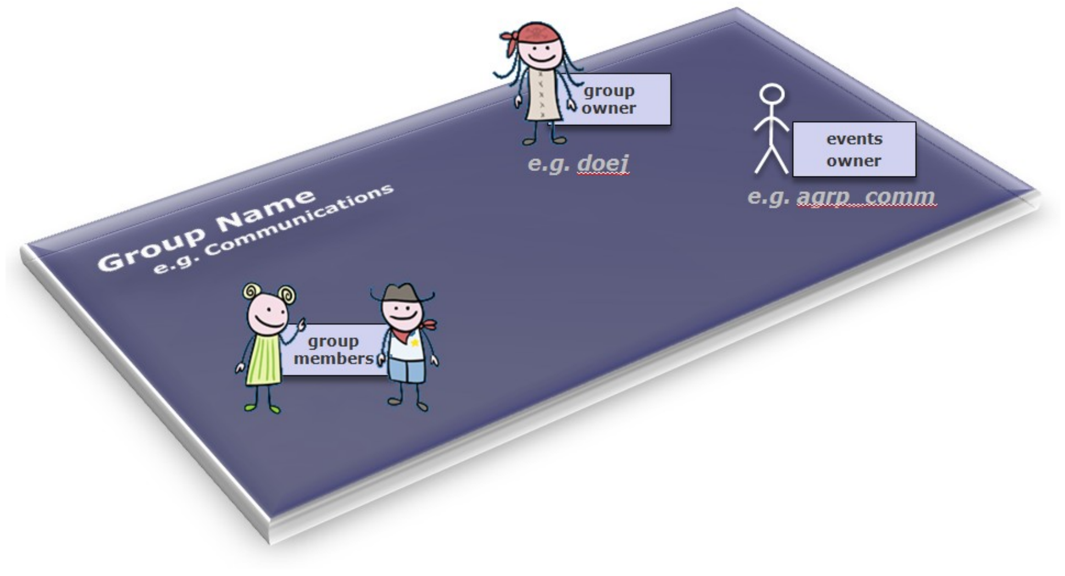
4.3 Managing Users and Admin Groups

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.

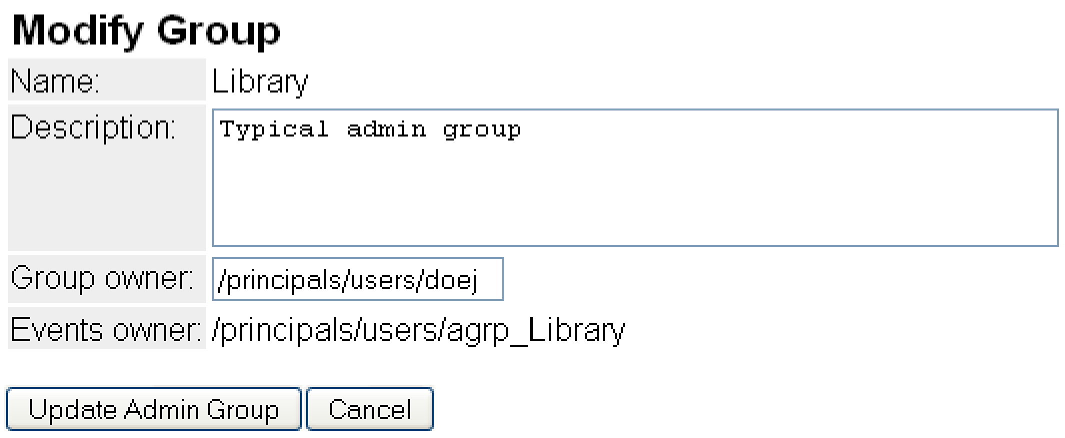
Calendar suites are attached to groups. An event administrator will add and edit events in the context of the calendar suite that is found first when traversing back up the tree.



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 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 “agrp_”.


Modify Group

Name:	Library
Description:	Typical admin group
Group owner:	/principals/users/doej
Events owner:	/principals/users/agrp_Library

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. The super user(s) 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".

While it is possible to close branches of the /public sub-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.

Event administrators and superusers

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).

A superuser may be added to the system by selecting System (tab) → Manage system preferences, and adding a user id to the “Super Users” field. The field accepts a list of comma separated user ids.

4.4 Understanding and architecting your calendar hierarchy

Introduction

Bedework 3.5 adopts a single calendar model for publishing public events. This model provides flexibility and power for Bedework deployers while simplifying the user interface for event administrators. The 3.5 release represents a major architectural change from previous releases and is intended to meet the demands of the community running Bedework's public events system. Bedework 3.5 provides the features of Bedework 3.4.1.1 with the addition of filters, greatly enhanced subscriptions, and tighter control over categories and calendar suites. Much of what was done in the 3.4.1.1 stylesheets to filter events can now be accommodated in the 3.5 administrative client.

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 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 filters on output.

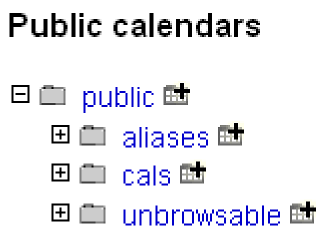
Tagged input: Subscriptions in a calendar suite may be marked as “Topical Areas” and assigned categories. Topical Areas appear in the add event form for the calendar suite and provide a means for event administrators to tag events by one or many categories. Topical Areas allow superusers and calendar suite owners to control which categories an event administrator may apply to an event. A subscription that is not tagged 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 can tag events with the "Topical Areas" defined for that suite. Each topical area may apply one or many categories to an event. All details of this are hidden from the end user. This approach simplifies the event admin's user interface substantially: the user doesn't need to know what categories to select to make an event appear in a particular view or data feed; nor does the user need to know anything about the underlying calendar structure. Each suite defines its aggregate "topical areas", and the admin user needs only understand these. For example, a topical area named “Front Page Highlights” may be available only to event administrators working within the context of the “Communications” calendar suite and could be used to set several categories on an event that cause it to appear on the front of the organization's website.

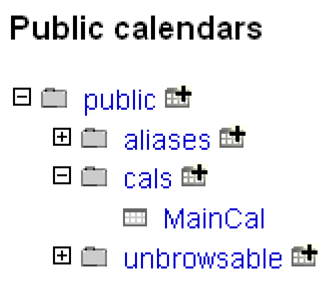
To summarize: subscriptions marked as topical areas are used to tag events on input, and subscriptions filter events on output based on the category filters applied to them. When a subscription points to an alias in the public tree, events are filtered by the union of all filters in the subscription 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 Structure

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:

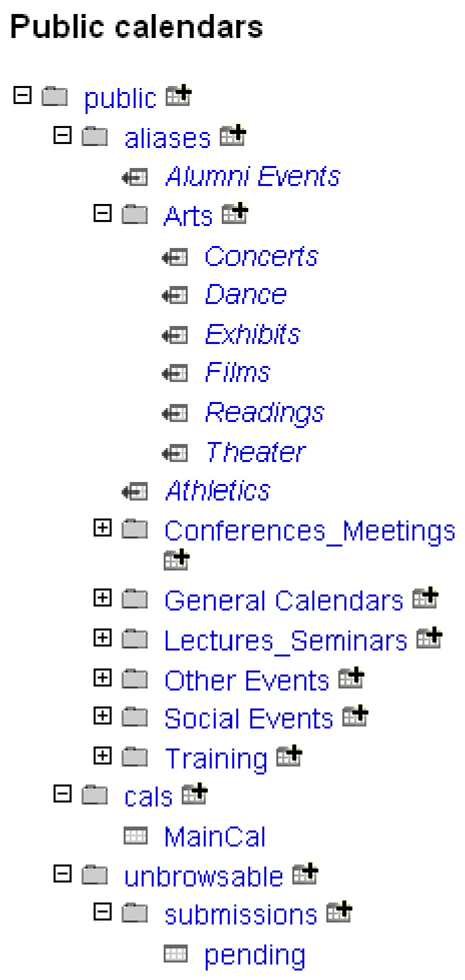


Public calendars

```
⊟ public
    ⊞ aliases
    ⊟ cals
        MainCal
    ⊞ unbrowsable
```

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



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

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.

4.5 Managing Calendar Suites

Bedework's web views of *public event information* are called "Calendar Suites". Calendar Suites provide a custom web context, custom look and feel, custom subscriptions, and custom views of event data. The quickstart comes with two suites, named MainCampus and SoEDepartmental. Calendar Suites are best used for large or significant groups within an organization. For example, a school may warrant a Calendar Suite, while a department within that school may only need a filtered view on the school's suite and an event data feed for embedding in its departmental website. All calendar suites for a single Bedework instance pull event data from the same central pool of events (plus any subscriptions to external calendars).

Calendar suite owners (or superusers) build a calendar suite structure by adding subscriptions to collections in the public calendar tree or to external calendars. The calendar suite owner may subscribe directly to /public/cals/MainCal or to any of the calendar aliases in the tree making filtering easier. Subscriptions marked as "topical areas" will be exposed to event administrators for tagging in the add event form.

The public calendar tree is not directly visible in the add event form. Rather, the event admin sees only the "Topical Areas" defined by the calendar suite owner.

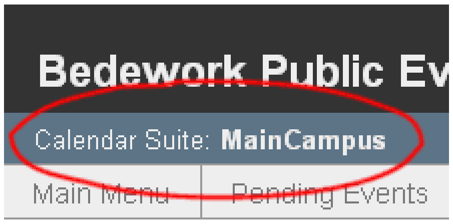
Note: while it is necessary to have at least one calendar suite for public events, you can pull data feeds of events for sub-organizations without creating a new suite. See section 4.6 "Managing Events" → "Getting events out."

Managing a calendar suite

Calendar Suites and Administrative Groups

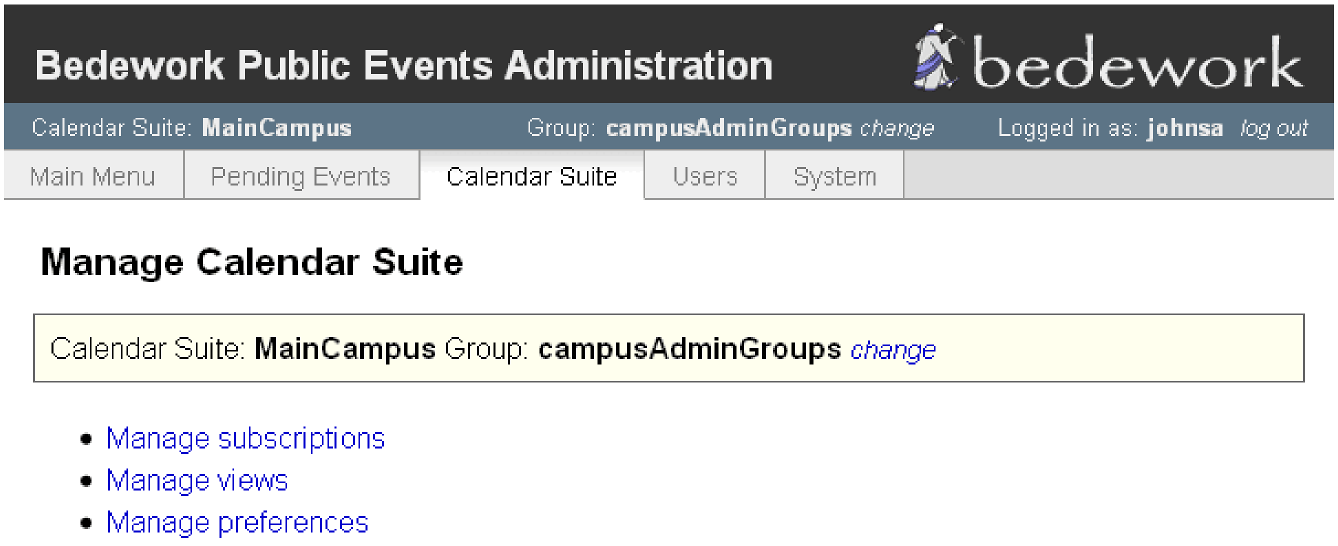
A calendar suite is associated with an administrative group. Members of this group (*calendar suite owners*) can administer the calendar suite's preferences, subscriptions, and views. Event admins in groups that are children of the calendar suite group tag events with topical areas defined in the calendar suite. When an event is added, categories are applied to the event based on the topical areas chosen.

The calendar suite you are working within is displayed in the upper left of the admin client user interface:



Administering a Calendar Suite

To administer a calendar suite, log in to the admin client as a member of the calendar suite group, or change groups once logged in. When logged in as a member of a calendar suite group, you will see a "Calendar Suite" tab:



Calendar Suite: **MainCampus** Group: **campusAdminGroups** *change* Logged in as: **johnsa** *log out*

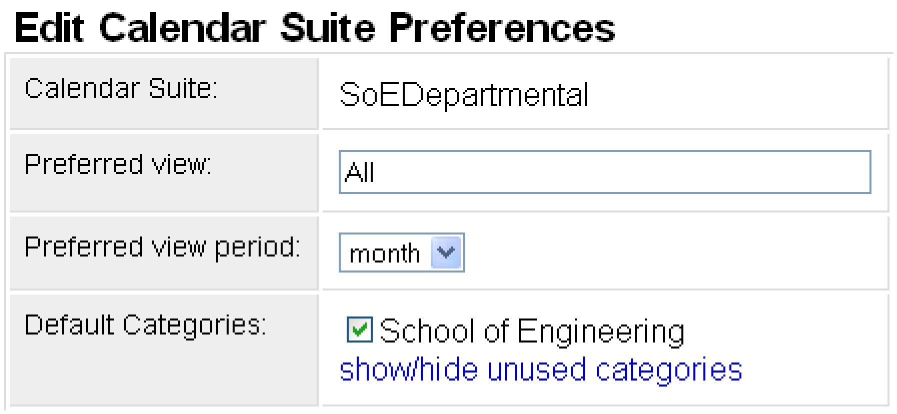
Manage Calendar Suite

Calendar Suite: **MainCampus** Group: **campusAdminGroups** *change*

- Manage subscriptions
- Manage views
- Manage preferences

It is here that you may mange subscriptions, views, and preferences for the suite.

Managing Calendar Suite Preferences:
The calendar suite preferences let you set the default view, default view period (day, week, month), and any default categories for the suite. When default categories are assigned, every event created in the context of the suite will be tagged with those categories.



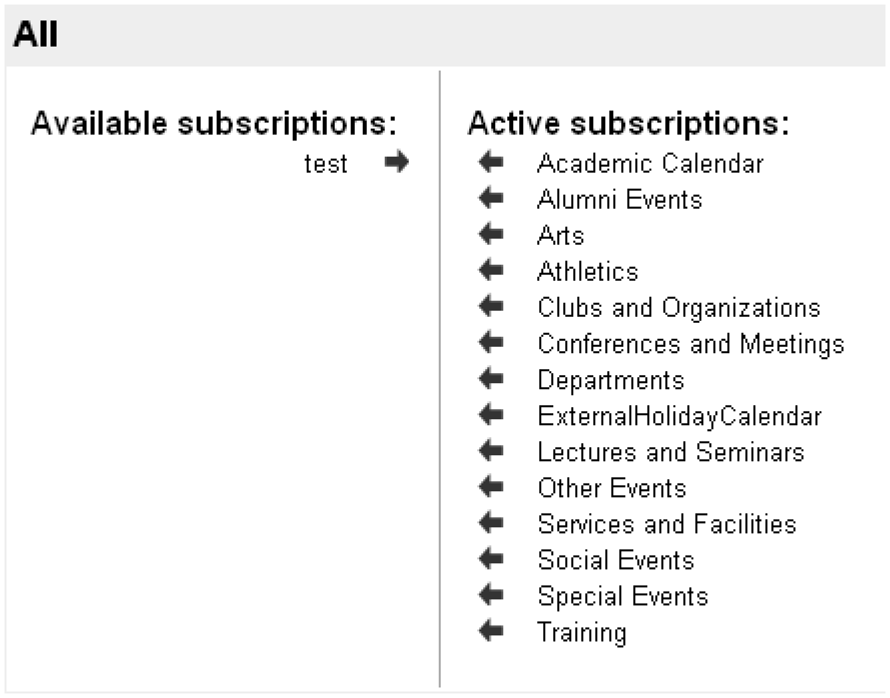
Edit Calendar Suite Preferences

Calendar Suite:	SoEDepartmental
Preferred view:	All
Preferred view period:	month
Default Categories:	✓ School of Engineering show/hide unused categories

Managing 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.

Managing Calendar Suite Views:
Subscriptions are aggregated into views for display in the public web client. The "preferred view" defined in the calendar suite preferences provides the default display of events when a user first visits the public web client for the calendar suite. The following screenshot shows an example view called "All" that contains most of the subscriptions available in the suite.

When a user selects this view, they will see all events revealed through the subscriptions contained in the view.


All

Available subscriptions:

test

Active subscriptions:

Academic Calendar
Alumni Events
Arts
Athletics
Clubs and Organizations
Conferences and Meetings
Departments
ExternalHolidayCalendar
Lectures and Seminars
Other Events
Services and Facilities
Social Events
Special Events
Training

Adding a New Calendar Suite

A Calendar Suite must be built, and the resulting .war file can be deployed alongside the other suites.

Building a Bedework Calendar Suite

1. Follow the instructions in chapter 3.2 for setting up your build environment.
2. Add calendar suites to your myconfig.properties and myconfig.options.xml files, 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 myconfig.properties file for the property: org.bedework.install.app.names (line 19 in Bedework 3.5).
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

Configuring a Bedework Calendar Suite

Once built, you must add the Calendar Suite to the admin client and associate it with an administrative group.

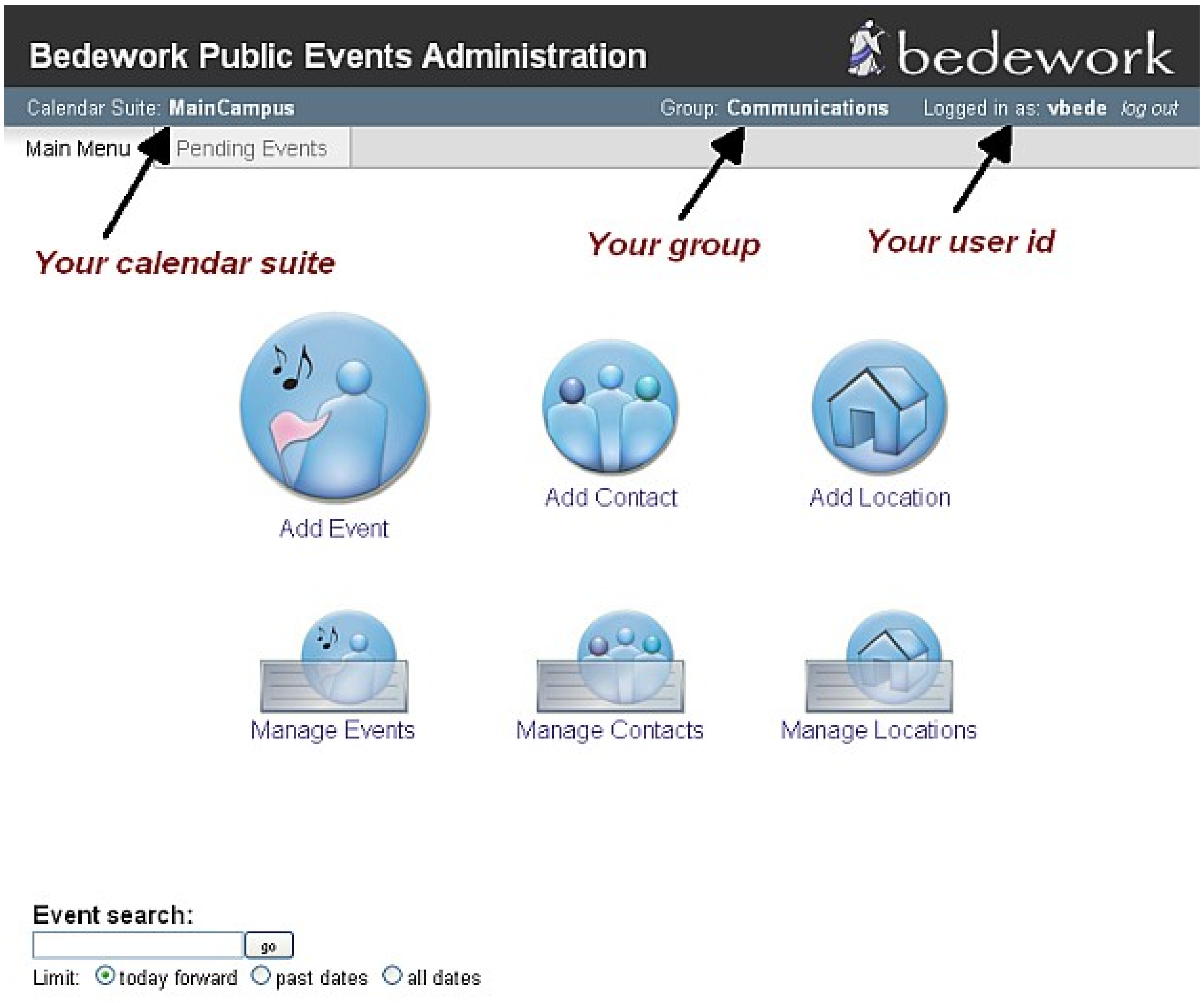
1. Create an admin group from which to hang the Calendar Suite:
    1. Log into the Bedework Admin Client as a superuser.
    2. Select the "Users" tab.
    3. Select "Manage admin groups"
    4. Click the button "Add a new group"
    5. Set the group's properties:
        1. Name: CalSuite-MyDept *(note: 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.)*
        2. Description: provide a reasonable description for the group, e.g. "calendar suite for MyDept"
        3. Group owner: a userid, probably of the main contact point for the group; this field is only informational
        4. 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 calendar suite
    1. Select the "System" tab.
    2. Select "Manage Calendar Suites"
    3. Click the button "Add Calendar Suite"
    4. Set the Calendar Suite's properties:
        1. 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".
        2. Group: this is the name of the group you defined in step 1, e.g. "CalSuite-MyDept".
        3. Root calendar: this will almost always be "/public", the root of the public calendar tree.

4.6 Managing Events

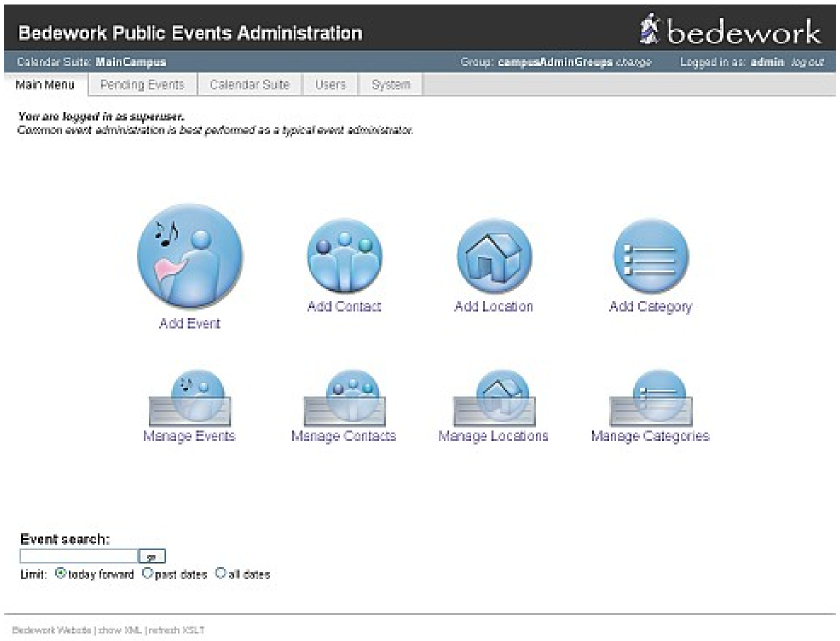
Getting events in

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:



for a typical event administrator, and like this:



Calendar Suite: **MainCampus** Group: **campusAdminGroups** *change* Logged in as: **admin** *log out*

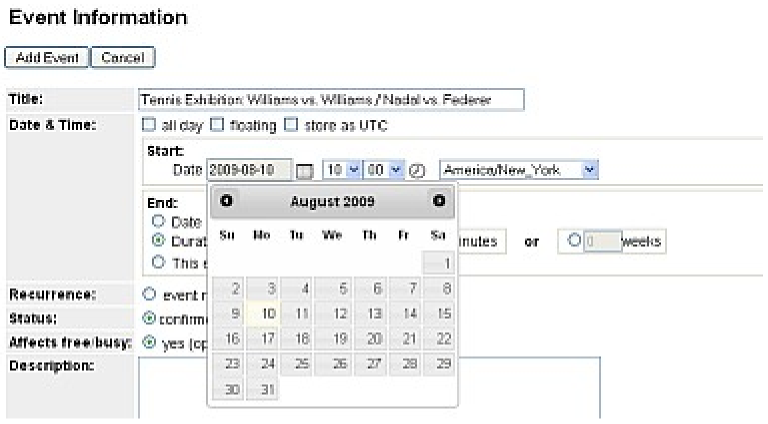
You are logged in as superuser.
Common event administration is best performed as a typical event administrator.

Event search:
Limit: [✓ today forward] past dates all dates

for a super user.

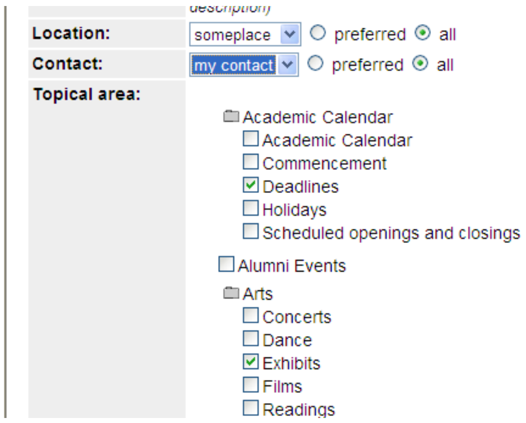
Creating events

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", or you can select your own.
4. Select "all day" if this is an all day event. Select floating if you don't want to be restricted by time zones. Or if you want to enter a specific start and end time for your event follow step 5.
5. Select the ▦ icon and select a start date for the event.



Title: Tennis Exhibition: Williams vs. Williams / Nadal vs. Federer
Date & Time: all day floating store as UTC
Start:
Date 2009-08-10 10 00 America/New_York
End:
Date
[✓ Durat] …inutes or 0 weeks
This
Recurrence: event r
Status: [✓ confirm]
Affects free/busy: [✓ yes (op]
Description:

6. Select a start time
7. Select the ▦ icon and select an end date for the event
8. Select an end time or duration. Or select the "this event has no duration or end date" radio button if the event has none.
9. Under recurrence leave at "this event does not recur". See the next section for recurring events.
10. Leave the event status as "confirmed". Or if the event will change, select "tentative". If an event has been cancelled, select "cancelled" - this will display a cancellation notice in the event title in calendar suites and data feeds.
11. Enter more details about the event in the description.
12. Enter a cost for the event (optional).
13. Enter a URL for the event, if you have a website with more information about it (optional).
14. You can add an image to the event (optional).
15. Select a location for the event. For a more comprehensive list of locations, select the "all" radio button. The preferred listing will display the locations you've already used.
16. Select a contact for the event. For a more comprehensive list of contacts, select the "all" radio button. The preferred listing will display the contacts you've already used.
17. Select the topical area(s) with which you want to tag the events



Location: someplace ( ) preferred (● all)
Contact: my contact ( ) preferred (● all)
Topical area:
- Academic Calendar
  - ☐ Academic Calendar
  - ☐ Commencement
  - ☑ Deadlines
  - ☐ Holidays
  - ☐ Scheduled openings and closings
- ☐ Alumni Events
- Arts
  - ☐ Concerts
  - ☐ Dance
  - ☑ Exhibits
  - ☐ Films
  - ☐ Readings

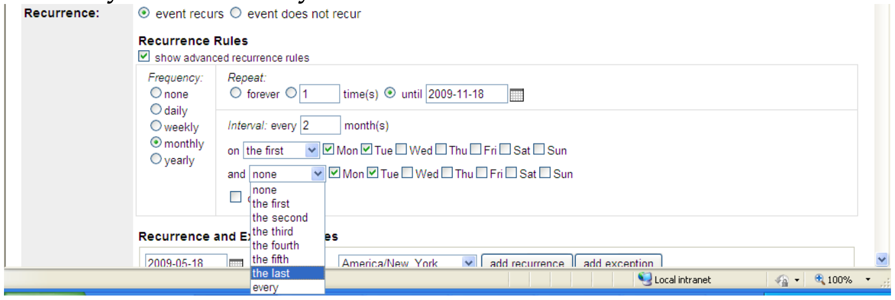
18. Select the "add event" button

To create a recurring event

1. Follow steps 1-8 above
2. Select "event recurs"
3. Select how often the event recurs under frequency... for example if the event recurs every month, select the "monthly" radio button
4. Leave at forever if you want the event to repeat forever. If not specify how many times

you want the event to recur by selecting the "times" radio button and typing the number times to repeat the event. Or alternatively select the "until" radio button and specify 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 select the "show advanced rules" check box. You might need to select the "monthly" (or frequency of your choice) radio button again to see the advanced rules.
6. If you want the event recurring every other month type in 2 under the "interval" label or any interval of your choice.
7. 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,



Recurrence: [✓ event recurs] event does not recur

Recurrence Rules

[✓] show advanced recurrence rules

Frequency: none daily weekly [✓ monthly] yearly

Repeat: forever 1 time(s) [✓ until] 2009-11-18

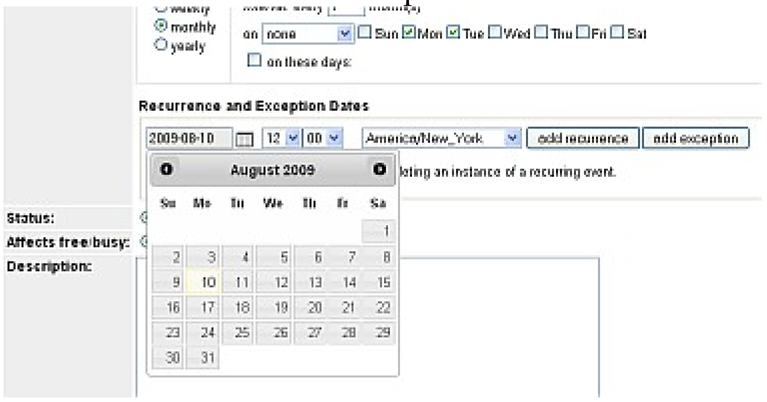
Interval: every 2 month(s)

on the first [✓ Mon] [✓ Tue] Wed Thu Fri Sat Sun

and none [✓ Mon] [✓ Tue] Wed Thu Fri Sat Sun

select as shown in the diagram.

8. 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 selecting a date and time under the "Recurrence and exception dates" field and select the "add exception" button.



Frequency: [✓ monthly] yearly

on none Sun [✓ Mon] [✓ Tue] Wed Thu Fri Sat

on these days:

Recurrence and Exception Dates

2009-08-10 12 00 America/New_York

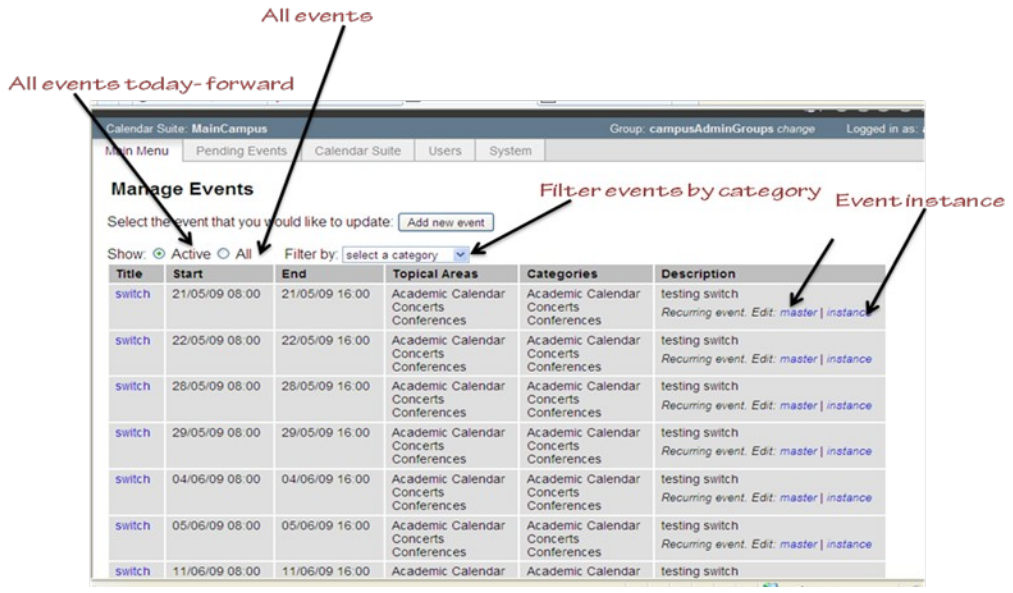
Su	Mo	Tu	We	Th	Fr	Sa
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Note: the time you enter for the exception must be the same as the start time for the event. ***Deleting a recurrence instance of an event also creates an exception.*** *See section on editing/ deleting events.*

9. 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.
10. Follow steps 10-18 above to add the event.

To Edit/delete an event

1. Select the “manage events” link under main menu. Alternatively you can search for the event by typing the event name under the “event search” label in the main menu.



Calendar Suite: MainCampus Group: campusAdminGroups change Logged in as:

Manage Events

Select the event that you would like to update:

Show: [✓ Active] All Filter by: select a category

Title	Start	End	Topical Areas	Categories	Description
switch	21/05/09 08:00	21/05/09 16:00	Academic Calendar Concerts Conferences	Academic Calendar Concerts Conferences	testing switch Recurring event. Edit: master | instanc
switch	22/05/09 08:00	22/05/09 16:00	Academic Calendar Concerts Conferences	Academic Calendar Concerts Conferences	testing switch Recurring event. Edit: master | instance
switch	28/05/09 08:00	28/05/09 16:00	Academic Calendar Concerts Conferences	Academic Calendar Concerts Conferences	testing switch Recurring event. Edit: master | instance
switch	29/05/09 08:00	29/05/09 16:00	Academic Calendar Concerts Conferences	Academic Calendar Concerts Conferences	testing switch Recurring event. Edit: master | instance
switch	04/06/09 08:00	04/06/09 16:00	Academic Calendar Concerts Conferences	Academic Calendar Concerts Conferences	testing switch Recurring event. Edit: master | instance
switch	05/06/09 08:00	05/06/09 16:00	Academic Calendar Concerts Conferences	Academic Calendar Concerts Conferences	testing switch Recurring event. Edit: master | instance
switch	11/06/09 08:00	11/06/09 16:00	Academic Calendar	Academic Calendar	testing switch

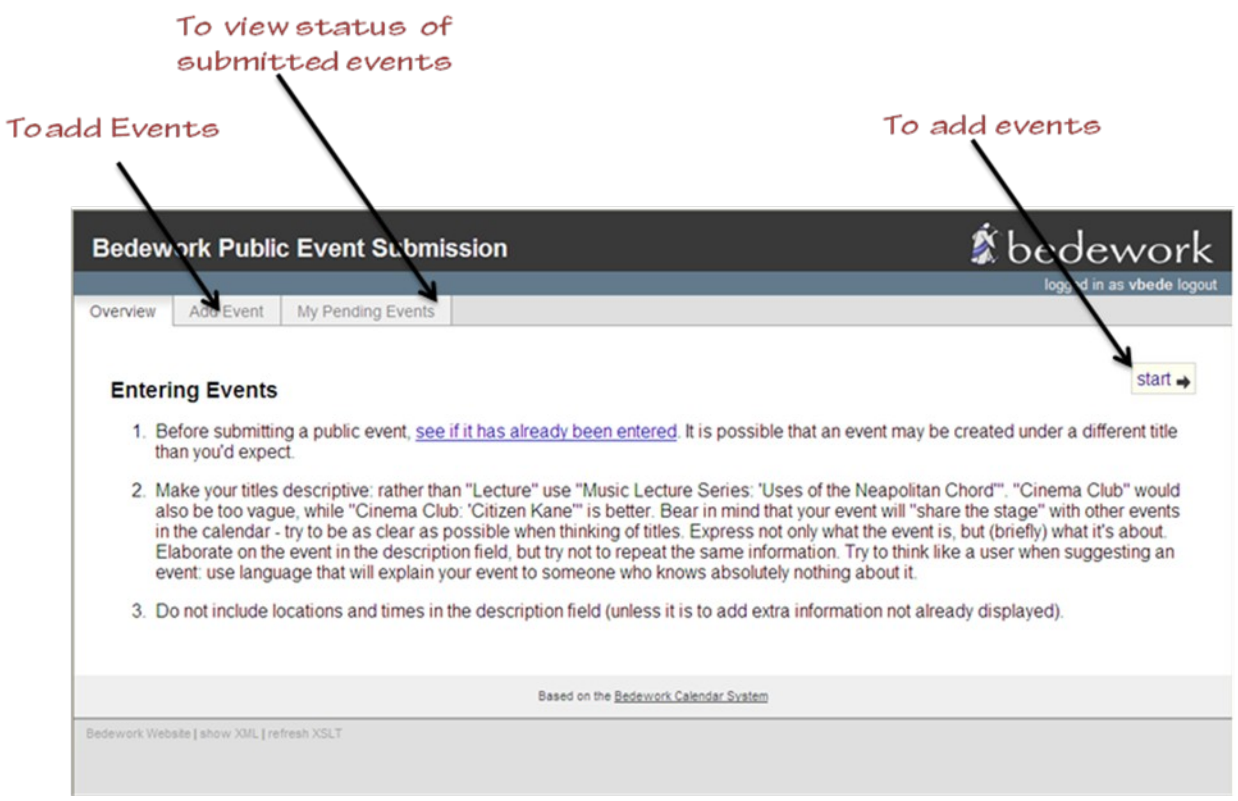
2. You can show all events or active events or you can filter events by selecting a category.
3. To edit an event select the event under the “Title” column or if the event recurs select “master” under the “Description” column. Changes you make to the master affect all instances of the event. To edit/ delete a single instance of an event, select the “instance” link under description.
4. When you are done with your changes select the “update event” button.

Importing events

Superusers can import events in the admin client by going to the system tab and selecting “Upload ical file”. The currently active group will own the events. At the moment 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. You may also wish to consider subscribing to the ical file instead of direct import.

Community submissions client

The community 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/


To view status of submitted events
To add Events
To add events
Bedework Public Event Submission
bedework
logged in as vbede logout
Overview
Add Event
My Pending Events
start

Entering Events

1. Before submitting a public event, see if it has already been entered. It is possible that an event may be created under a different title than you'd expect.
2. Make your titles descriptive: rather than "Lecture" use "Music Lecture Series: 'Uses of the Neapolitan Chord'". "Cinema Club" would also be too vague, while "Cinema Club: 'Citizen Kane'" is better. Bear in mind that your event will "share the stage" with other events in the calendar - try to be as clear as possible when thinking of titles. Express not only what the event is, but (briefly) what it's about. Elaborate on the event in the description field, but try not to repeat the same information. Try to think like a user when suggesting an event: use language that will explain your event to someone who knows absolutely nothing about it.
3. Do not include locations and times in the description field (unless it is to add extra information not already displayed).

Based on the Bedework Calendar System

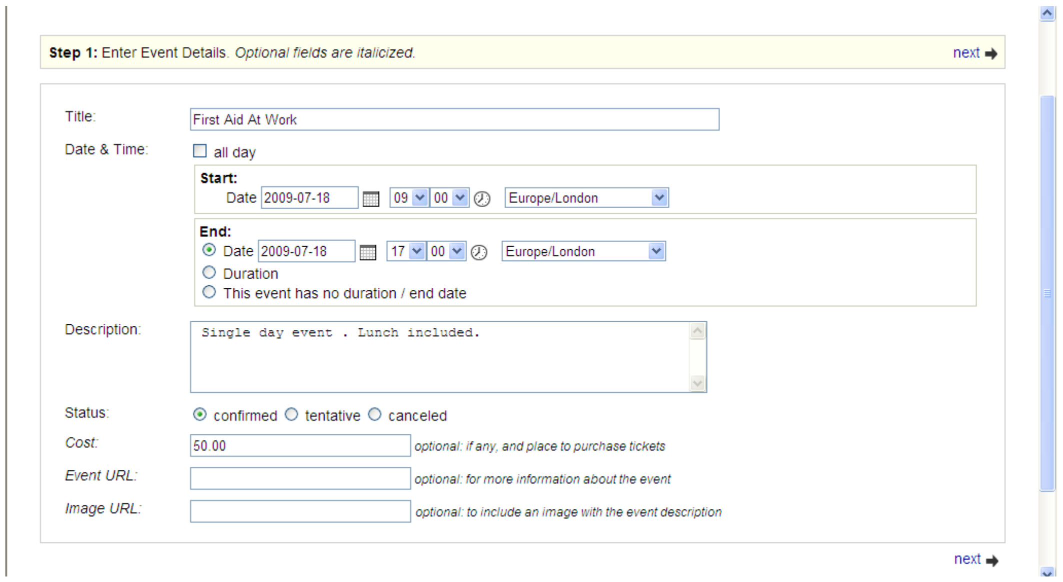
Bedework Website | show XML | refresh XSLT

Submitted events do not automatically go into the public calendar. They are viewed by an event administrator in the admin client for approval. The event administrator can approve, reject or edit the event. If the event is approved, it is displayed in the public calendar for the administrator's calendar suite. You can view the status of the event by clicking the "My Pending Events" tab. You can create an event for submission by selecting the "Start" link or the "Add Event" tab.

To add an event

1. Type in a title for the event
2. Select "all day" if this is an all day event. Or if you want to enter a specific start and end time for your event follow step 3.
3. Select the ▦ icon and select a start date for the event
4. Select a start time
5. Select the ▦ icon and select an end date for the event
6. Select an end time or duration. Or select the "this event has no duration or end date" radio button if the event has none. *Note: the format of the dates, times, and duration presented to the user will be the user's preferences as set in the personal client. If you are not using the personal client, the user will be presented with the default preferences for all users as configured at build time.*
7. Leave the event status as "confirmed". Or if the event will change, select "tentative".

8. Enter more details about the event in the description.
9. Enter a cost for the event (optional).
10. Enter a URL for the event, if you have a website with more information about the event (optional).
11. You can add an image to the event (optional).
12. Select the "Next " link



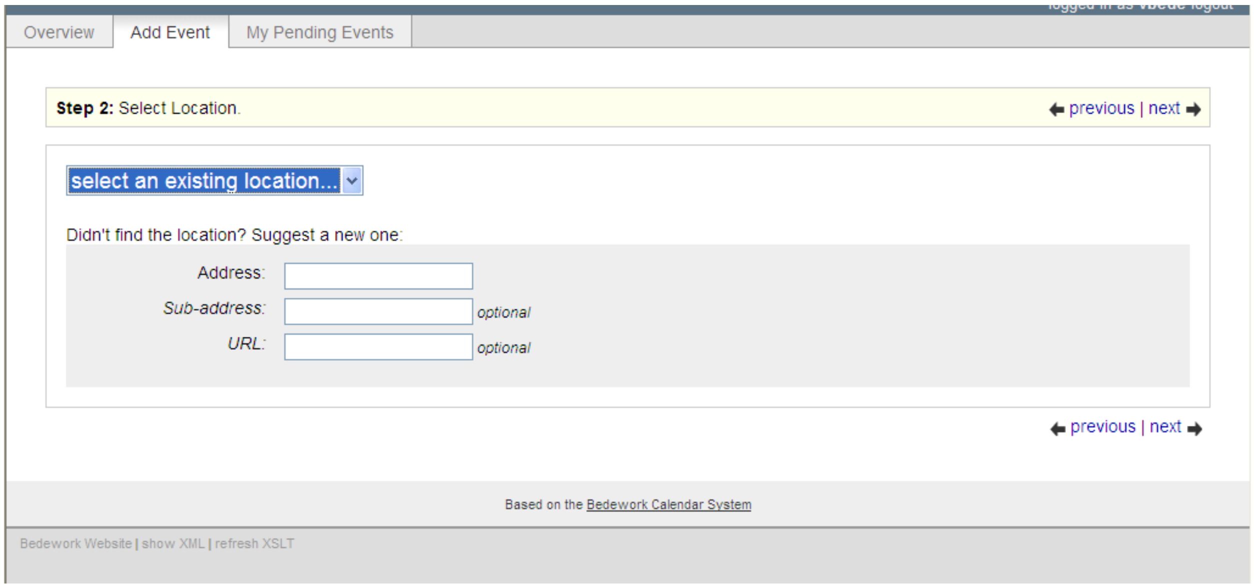
Step 1: Enter Event Details. *Optional fields are italicized.* next

Title:	First Aid At Work
Date & Time:	☐ all day
	Start: Date 2009-07-18 09 00 Europe/London
	End: [✓ Date] 2009-07-18 17 00 Europe/London; Duration; This event has no duration / end date
Description:	Single day event . Lunch included.
Status:	[✓ confirmed] tentative canceled
Cost:	50.00 *optional: if any, and place to purchase tickets*
Event URL:	*optional: for more information about the event*
Image URL:	*optional: to include an image with the event description*

next

Note: You cannot submit a recurring event. However, recurrence information can be described on the last page of the add event wizard in the box for user comments or instructions.

13. Select a location by clicking on the down arrow button. Or you can suggest a new location by entering the information in the "Address" "Sub-Address" and "URL" fields. New locations you enter here needs to be approved by a user with administrative rights for the calendar suite.
14. Select the "next" link. You can select the "previous" link to edit the information on the previous page


Overview
Add Event
My Pending Events

Step 2: Select Location.

select an existing location...

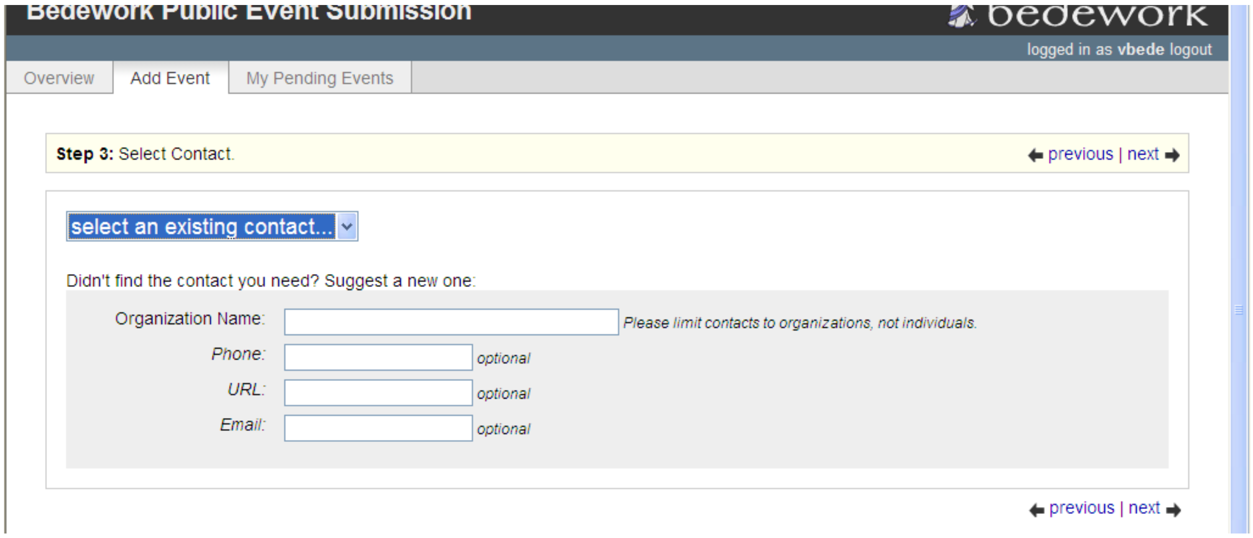
Didn't find the location? Suggest a new one:

Address:		
Sub-address:		*optional*
URL:		*optional*

Based on the Bedework Calendar System

Bedework Website | show XML | refresh XSLT

15. Select a contact. Or you can suggest a new contact by entering the information in the “Organisation Name” field, you can also fill out the optional fields. New contacts you enter here needs to be approved by a user with administrative rights for the calendar suite.
16. Select the “next” link


Bedework Public Event Submission
bedework
logged in as vbede logout
Overview
Add Event
My Pending Events

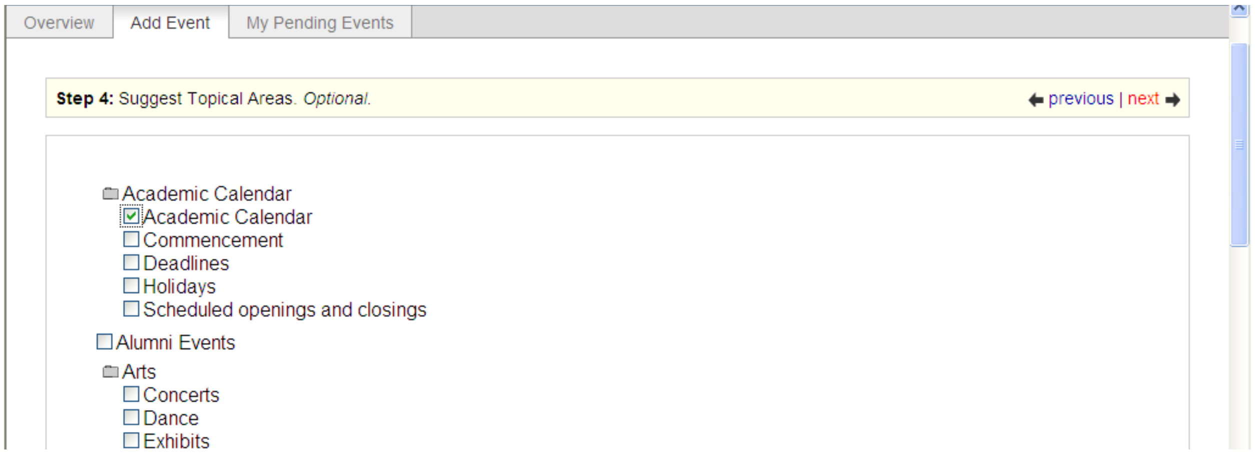
Step 3: Select Contact.

select an existing contact...

Didn't find the contact you need? Suggest a new one:

Organization Name:		*Please limit contacts to organizations, not individuals.*
Phone:		*optional*
URL:		*optional*
Email:		*optional*

17. Select the topical areas for the event



Step 4: Suggest Topical Areas. *Optional.*

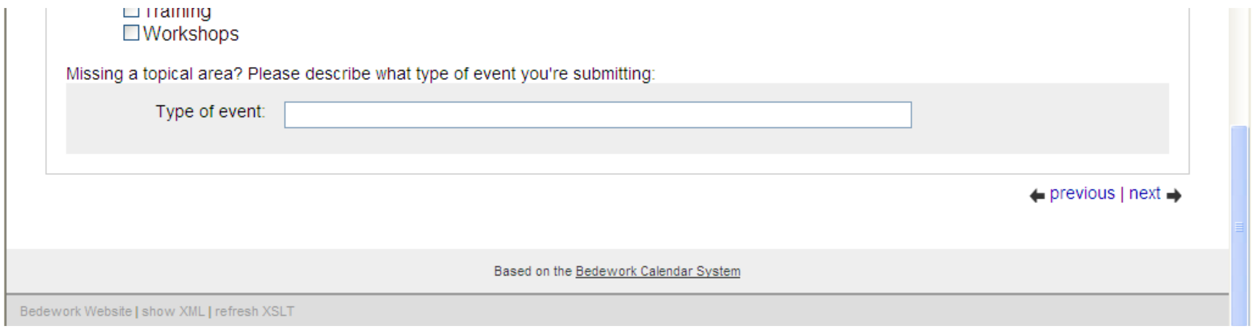
previous | next

☐ Academic Calendar
- ☑ Academic Calendar
- ☐ Commencement
- ☐ Deadlines
- ☐ Holidays
- ☐ Scheduled openings and closings

☐ Alumni Events

Arts
- ☐ Concerts
- ☐ Dance
- ☐ Exhibits

18. Alternatively you can enter the event type if none of the topical areas fit the type of event you are submitting


- ☐ Training
- ☐ Workshops

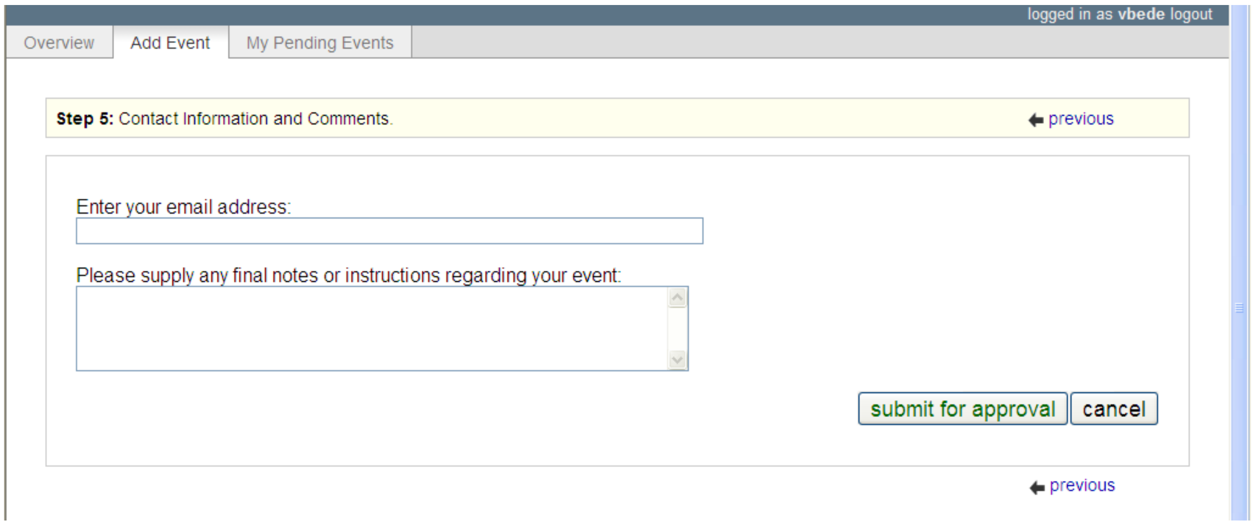
Missing a topical area? Please describe what type of event you're submitting:

Type of event:

previous | next

Based on the Bedework Calendar System

19. Select the " next" link
20. Enter your email address
21. Enter additional notes or instructions for the event. For example, you can request that the administrator book a room for the event.
22. Select the "Submit for Approval" button to submit the event or select the "cancel" button to cancel the event.



Step 5: Contact Information and Comments.

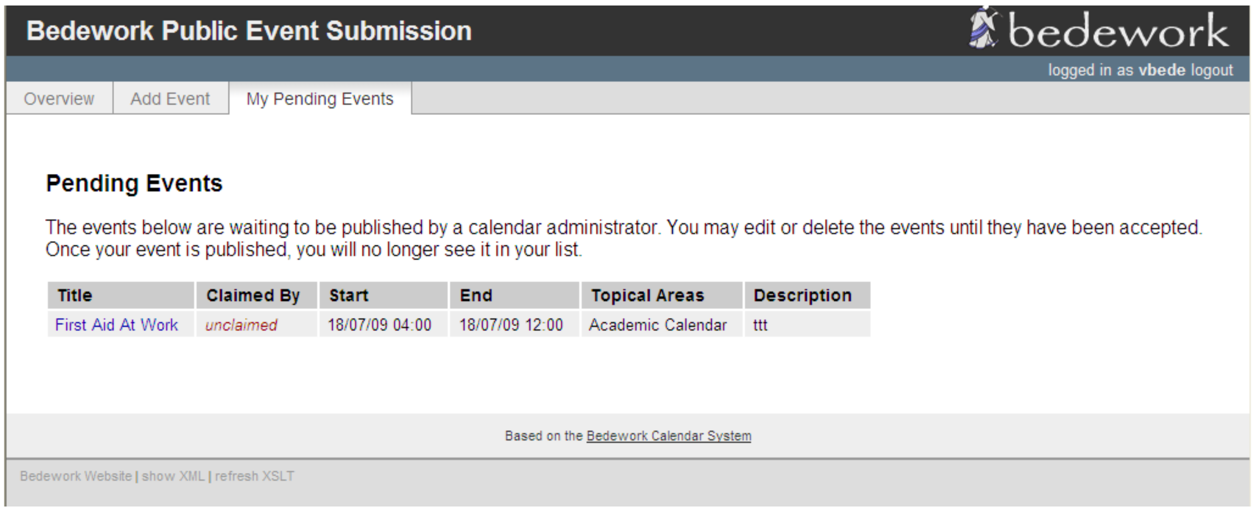
Enter your email address:

Please supply any final notes or instructions regarding your event:

You can view the status of the events you submit for approval.

To view the status of an event

1. Select the "My Pending Events" tab


Bedework Public Event Submission
bedework
logged in as vbede logout
Overview
Add Event
My Pending Events
Bedework Website | show XML | refresh XSLT

Pending Events

The events below are waiting to be published by a calendar administrator. You may edit or delete the events until they have been accepted. Once your event is published, you will no longer see it in your list.

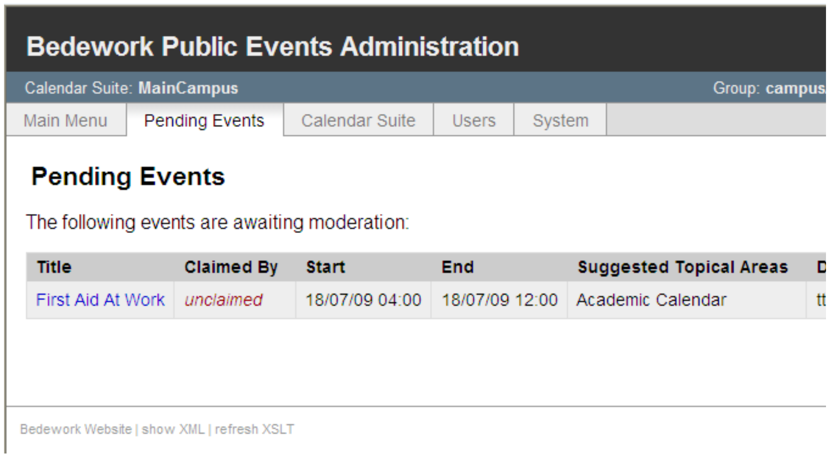
Title	Claimed By	Start	End	Topical Areas	Description
First Aid At Work	*unclaimed*	18/07/09 04:00	18/07/09 12:00	Academic Calendar	ttt

Based on the Bedework Calendar System

Here you can find information regarding the status of your event. The "Claimed By" column lets you know which administrator is handling your request. You can also select the event title link in the "Title" column to edit or delete the event. An administrator for the calendar suite can claim the event by login to the administrative client.

To administer a submitted event

1. Login the administrative client
2. Select the "Pending Events" tab


Bedework Public Events Administration
Main Menu
Pending Events
Calendar Suite
Users
System
Bedework Website | show XML | refresh XSLT

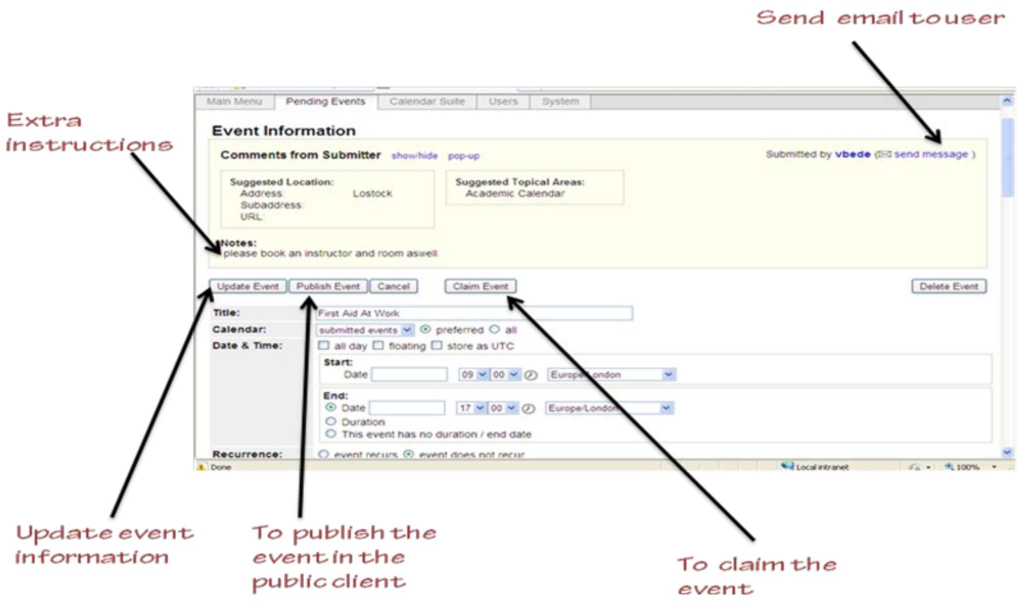
Calendar Suite: **MainCampus** Group: **campus**

Pending Events

The following events are awaiting moderation:

Title	Claimed By	Start	End	Suggested Topical Areas	D
First Aid At Work	*unclaimed*	18/07/09 04:00	18/07/09 12:00	Academic Calendar	tt

3. Select the event title link under the “title” column



Event Information

Comments from Submitter show/hide pop-up

Submitted by **vbede** (send message)

Suggested Location:
Address: Lostock
Subaddress:
URL:

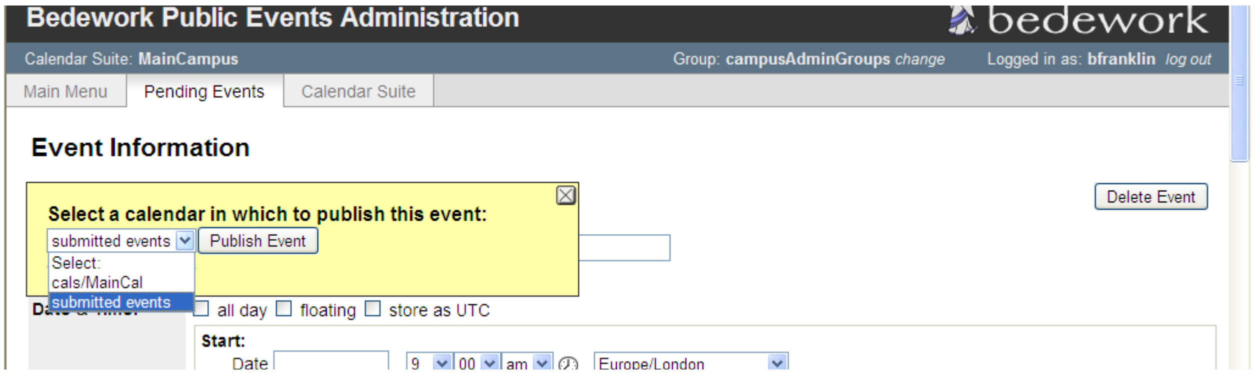
Suggested Topical Areas:
Academic Calendar

Notes:
please book an instructor and room aswell.

Title:	First Aid At Work
Calendar:	submitted events [✓ preferred] all
Date & Time:	all day, floating, store as UTC
	Start: Date 09 00 Europe/London
	End: [✓ Date] 17 00 Europe/London; Duration; This event has no duration / end date
Recurrence:	event recurs [✓ event does not recur]

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 selecting the “show / hide” link. You can also edit the other fields on the form such as entering a topical area.

- To claim the event, select the “Claim Event” button.
- To publish the event, select the “Publish Event” button. When you select this, the event can be viewed by users in the public client. If your system uses the single calendar model (this is the default set up in Bedework 3.5) you are done.
- 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”



Getting events out

Calendar suites

Each calendar suite in your Bedework implementation can produce a custom web site, and you will need at least one calendar suite to display public events or to produce data feeds. For more information, see the topics already covered in this chapter (section 4.5).

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).

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

Data feeds can be produced from any Bedework view, but 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 Bedework's theme directories.

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. Several 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.)

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. *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.

Examples:
/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=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
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/v3.5/DesignGuide/jsonFeeds

4.7 Adding custom properties (X-Properties)

X-Properties are a means of extending ical to define custom fields. The are very 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

Chapter 5 Personal and Group Calendaring

5.1 Overview.

Personal calendars allow users to carry out all the normal calendaring functions and provide a customized view of public events through subscription to public calendars. Users can access their personal calendars using the Bedework personal and group calendaring web client and over CalDAV using Apple's iCal, the iPhone default calendar app, Mozilla Lightening, the ZideOne plugin for Outlook, 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.

5.2 Scheduling Meetings

The scheduling features of Bedework are almost complete; there is certainly sufficient 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 **calendar 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.edu would 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 fro class lists etc.

Chapter 6 Bedework Theming

This chapter 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.

We will focus primarily on the public web interface, but the concepts apply to all of Bedework's web clients.

6.1 Overview

Skins

A generic skin is provided with each web application in the quickstart release. The public client's MainCampus calendar suite comes with three simple color schemes (red, green, and blue) to get you started. Fonts, colors, and most layout specifics are controlled by css.

MainCampus skin

Within the quickstart directory, the source skin files for the web applications can be found in the following three directories:

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

Each application has a default skin named *default.xsl* and one or more associated css stylesheets in the same directory. Public calendar suite skin sets are found in the webpublic directory. Here you will also find skins for producing rss feeds, javascript feeds, and output appropriate for use on PDAs and cell phones, as well as television screens.

Examples of how members of the Bedework community have implemented skins can be viewed by selecting from the “production examples” pull-down list in the demo public client or 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

The "quickstart" distribution comes with skins in place and ready to use. If you rebuild the application (you will want to do this to create a production release), the skins are copied into:

```
jakarta-tomcat-5.0.28/webapps/ROOT/calrsrc.MainCampus and
[ jakarta-tomcat-5.0.28/webapps/ROOT/calrsrc.OtherCalSuite and ]
jakarta-tomcat-5.0.28/webapps/ROOT/ucalrsrc and
jakarta-tomcat-5.0.28/webapps/ROOT/caladminrsrc
```

Once deployed, the skins can be manipulated directly within the Tomcat webapps directory for quickest development (or copied in from the source folder -- our recommended approach); but, if you want to keep your changes, make certain to keep or copy edited files in the source folder, or your work will be overwritten when the application is rebuilt. If you have followed our recommendation to place your templates on a separate web server, you will be safe from a rebuild overwrite.

The locations of the skins and their deployment directories are defined in the file:

```
bedework/config/configs/{clone}.properties
```

It is strongly recommended that you begin with the "demoskins" build (default) and that you make a backup copy of the source folder before you begin.

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). It is highly recommended that you read through the specifications for these technologies at the World Wide Web Consortium.

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.

Structure of the Template Directories

The skin directory for a client deployment of the calendar is called the application root or "appRoot". This is the directory in which the designer will work. The appRoot has the following structure:

appRoot = *http://a-web-server-path/to/your/template/directory/*
which for the default quickstart build is
appRoot = *http://localhost:8080/calrsrc/* and
appRoot = *http://localhost:8080/ucalrsrc/* and
appRoot = *http://localhost:8080/caladminrsrc/*

The appRoot is then further modified at run time for portals and calendar suites. If we running under a portal the approot has "." + portalPlatform appended Continuing with the above example, if the portal platform is “uPortal2” the root becomes
appRoot = *http://localhost:8080/ucalrsrc*.uPortal2

In addition, we append the calendar suite name for public clients.
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*

```
appRoot/
    |-- default/
    |     | -- default/
    |             |
    |             |-- default.xsl
    |              -- default.css
     -- images/
```

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. This level is also a convenient location for template *images*.

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, and PDA.

Bedework can be forced to use a specifc browser type and skin and can have multiple skins per browser type. See Actions & Parameters for more information about this. (Note: PDA browser sniffing is very much out-of-date; to use the PDA directory found in the quickstart release, add the request parameter *&browserTypeSticky=PDA. &browserTypeSticky=default* will reset this to the default path.)

For production deployment, we encourage placing the appRoot directories on an external web server where they can be modified without the need for redeploying the application.

Stylesheets are ***only*** loaded at first reference and are then cached. The stylesheets can be explicitly flushed using the "refreshXslt=yes" query parameter (see Actions and Parameters below).

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.

Making Basic Stylistic Changes

Each xslt skin is (mostly) self-contained in a file and is made up of a series of xsl *templates*. 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 (such as default.xsl/default.css). Skins are cached in memory, so if you choose to edit a live skin (e.g. in the Tomcat webapps directory) 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
```

NOTE: Be careful editing a live skin; though it is quick and convenient, if you do not copy your changes into the source skin directory, your edits will be overwritten during the next build/deploy. Again, we strongly encourage you to make a backup copy of the source folder before you begin.

Setting event colors in the calendar grid (Public Web Client)

Events can be colored by category in the public web client calendar grid (and the list as well). This is currently accomplished by customizing a template in the stylesheet. Search for the template that looks like this (around line 1386 in bedework/deployment/webpublic/webapp/resources/demoskins/MainCampus/default/default/default.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 as well.

Setting event colors in the calendar grid (Personal Web Client)

Events in the personal client will be colored based on the calendar to which they belong. However, because of the way 3.5 builds the view of events, *it is not possible at the moment to color subscriptions*. We are hoping to solve this for version 3.6.

6.2 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=20091121". This URL tells the personal calendar to set the current view to November 21, 2009.

Parameters can be strung together 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 guest and personal web clients are described in

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

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.

Parameters used primarily to effect skins are described in the next section: Bedework Skin Parameters.

Bedework Skin Parameters

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

1. **setappvar**=*key(value)*
    - applicaton variable: used to pass a key/value pair into the XML output
    - This feature is the equivalent of passing a parameter between pages in other frameworks.
    - You can pass as many appvars as you need.
    - appvars, once set, persist through a user session
    - To change the value of an appvar, send the same key with a different value.

2. **skinName**=*name*
3. **skinNameSticky**=*name*
    - These parameters explicitly select an xslt skin. skinName will switch the skin for one request/response; the sticky version will switch the skin for the remainder of the user session or until a different skin is called. name is the file name of an

XSLT document without the .xsl extension. For example:

```
    appRoot/
        |-- default/
        |     | -- default/
        |              |
        |              |-- default.xsl
        |              |-- default.css
        |              |-- shiny.xsl
        |              |-- shiny.css
        |              |-- lavared.xsl
        |               -- lavared.css
         -- images/
```

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

4. **browserType**=*default, MSIE, Netscape, Netscape4, Mozilla, PDA, other*
5. **browserTypeSticky**=*default, MSIE, Netscape, Netscape4, Mozilla, PDA, other*
    - These parameters explicitly select a browserType folder. browserType will switch the folder for one request/response; the sticky version will switch the folder for the remainder of the user session or until a different browserType is called. For example:

```
    appRoot/
        |-- default/
        |     | -- default/
        |     |        |
        |     |        |-- default.xsl
        |     |         -- default.css
        |     | -- Mozilla/
        |              |
        |              |-- default.xsl
        |               -- default.css
         -- images/
```

- 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.)

6. **refreshXslt**=*string*
   - 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. string can be any value; we typically set it to "yes". Example: update.do?catcenterkey=12&skinName=lavared&refreshXslt=yes

7. **noxslt**=*string*
   - 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. string can be any value; we typically set it to "yes". Example: update.do?catcenterkey=12&noxslt=yes

6.3 XML

Bedework XML Structure

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

1. 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.

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

```
bedework/projects/webapps/webclient/war/docs
and
bedework/projects/webapps/webadmin/war/docs
```

However, 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 XSL stylesheets 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.

6.4 Bedework XSLT

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/

Chapter 7 Integration With the Enterprise

7.1 Directories

7.2 Portals

Introduction

The intent is to have an application that can live inside and outside of a portal and be equally functional in both contexts. Some organizations that run bedework have no intention, at least currently, of running a portal. Others do not wish to make any of their portal applications visible outside of that framework.

To achieve this we decided to use the apache portals-struts bridge. This is a widely used package which, in theory, allows a struts based application to run inside a portal. In addition, that same deployed package is available outside of the portal if you so wish.

In reality, the portal context places a number of restrictions on the application. Some of these are easily met, (the action/render urls). Others restrictions cause problems, (see below).

While it is possible that organizations will want some different styling for the portal version, as much as possible we try to have a single stylesheet for both. Bedework is a complex application so inheriting styles from the portal may not work too well. At the very least however, much of the header and footers should be dropped for the portal.

Limitations of JSR168

To allow bedework to run in as many portals as possible we need a common framework. That framework in the portal world is JSR168 – the portlet specification.

Unfortunately, this specification left out some key pieces. The worst omission is the ability in some controlled way to set the content type. This prevents us from using a number of standard web practices in a portal agnostic fashion. We are unable to use Ajax, create pop-up windows or even to initiate a binary (or ics) download.

Most (possibly all) portals provide ways around these restrictions. However, because they are not part of any specification they are different for most portals. Thus, after the 3.3.1 release we reimplemented a number of features in the stylesheets which previously used popup windows. This does not prevent someone with sufficient knowledge of their portal framework from reimplementing such features using portal specific mechanisms. As our intent is to have an easily deployable portlet it does prevent us from distributing bedework with those features already in place.

Binary – or ics – downloads.
This still leaves us with the problem of binary downloads. In bedework this shows up in the downloading of ics files. At the moment it appears that the only portal agnostic solution to this problem is single-signon. Bedework as distributed will build links to the servlet version bypassing the portal altogether.

Without single signon, this leads to the authentication promp and, at the moment, a failure to download. Immediately following this with another attempt leads to a successful download.

With single signon, this process should work.

Popups
The possible solutions to popups are related to that for binary downloads. The portlet specification states (PLT.15.4) that "data stored in the `HttpSession` by servlets or JSPs is accessible to portlets". It appears it should be possible to share the current portlet session with the servlet and there is some suggestion that the current tomcat distibution allows this.

Hope on the horizon
The portlet specification is being worked upon and version 2 is on its way. This appears to deal with a number of the outstanding issues as well as handling a number of other areas passed over in the first version.

However, we will have to wait for portals to implement the new specification which may take some time.

7.3 Email

7.4 Other calendar systems

Chapter 8 Other Features of the Bedework Project

8.1 Timezone Server

Timezones are an important and at times awkward feature of calendaring. There is no official registry of timezones. The closest to such a registry is the Olson database which chooses to name timezones according to their continent and nearest large city, for example America/New_York.

System timezones

Bedework provides a set of timezones, the system timezones, which are available to all users of the system. The distributed system comes with timezones derived from the Olson database but any set of timezones could be used. The administrative application provides a means for replacing the system timezones by uploading an xml formatted data file.

Building timezone information.

First we need to convert the Olson database into a set of ics files. The data itself is available from ftp://elsie.nci.nih.gov/pub

The vzic program available via http://www.dachaplin.dsl.pipex.com/vzic/ is used to convert the zone information, Download and unpack the latest source and set the appropriate variables.

We set them as follows:

```
OLSON_DIR=wherever the data was unpacked.
PRODUCT_ID=-//BEDEWORK//NONSGML Bedework Calendar system//US
TZID_PREFIX=
```

The TZID_Prefix can be set to a value which indicates which system the timezone originated from, e.g. "/Bedework.org/". Having built vzic (make) and run it (./vzic) a directory named zoneinfo should be built. This is copied into the bwtolls/resources directory.

A copy of this generated data is available at http://Bedework.org/downloads/data/

The next step is to convert the data into an xml form for upload. Change directory into the bwtools project and run the following (all one line) which will create a file of xml timezone information:

```
java -cp bin/bw-tztools-3.2.jar:lib/log4j-1.2.8.jar
org.Bedework.tools.timezones.Timezones -dir
projects/bwtools/resources/zoneinfo -f tz.xml
```

A copy of this is generated file is also available at the link above. Finally we need to replace the system timezones in the production system.

Log in to the administrative client as a super user, go to “Upload and replace system timezones“, browse to the generated timezones file, then upload.

8.2 Bedework and CalDAV

Summary

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 their exists a certain tension between the needs of the different access methods.

Bedework is intended to be ultimately a complete implementation of CalDAV. At the moment we support most of the CalDAV operations (with varying degrees of success) but it is still a work in progress. 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.

8.3 Bw and CardDAV

8.4 Other Features

Freebusy URL

Bedework supports the use of a freebusy url to provide clients access to freebusy information. Users who wish to make their freebusy information available need to set schedule-freeebusy and read-freebusy access for the intended audience, for example to unauthenticated users, authenticated users or specific users or groups.

The freebusy url is currently being worked on by a CalConnect technical committee which will publish recommendations on request parameters and the form of the url so the implementation in Bedework may change.

Currently it takes the following form for the caldav servers:

```
<host-and-port>/<context>/fbsvc<parameters>
```

where:

- context is one of the following in the quickstart but may differ in production:
  - ucal for the authenticated user calendar
  - cal for the public unauthenticated calendar
  - ucaldav for the authenticated CalDAV server
  - caldav for the unauthenticated CalDAV server
- parameters is a list of request parameters as defined below.

Request parameters		
Name	**Value**	**Meaning**
user	A user account e.g. testuser01	A user account known to the system
cua	A calendar user address, e.g. mailto:testuser01@mysite.edu	This is the form that is usually found for an attendee.
start	Start date yyyy-mm-dd	Start of the period of interest
end	End date yyyy-mm-dd	End of the period of interest

The start and end may be omitted in which case a default period of information around the current time will be returned. The period is limited internally to a reasonable value.

Freebusy information is returned as a VFREEBUSY object.

While the information returned may be the same from each service there are some differences in usage. Currently, the web based services only support forms based authentication, the CalDAV services support basic and digest only.

An example url might be:

```
http://myhost.edu/ucaldav/fbsvc?user=test01&start=2007-09-06&end=2007-09-16
```

Real-time scheduling

This is currently under development and unsupported by most systems. Bedework allows the definition of known hosts (or more properly domains) which support realtime scheduling. When a calendar user address (CUA) is encountered which is not within the current system the default behavior is to mail the itip request to the user.

However, if the domain is in the list of known systems, the sever will attempt realtime scheduling with that server. This allows us to display at least the freebusy information for a user and perhaps send a scheduling itip message and receive the response.