

iCal2GW 2.8

Administrator Guide

About the Application

iCal2GW - Integrate Teaming with GroupWise Calendaring

The iCal2GW service synchronizes events, tasks and milestones from any calendar source in the iCal format into GroupWise calendars. This solution can be used to synchronize from sources on the Internet as well as with your Novell Teaming server - very cool!

The iCal2GW synchronizing service synchronizes events, tasks and milestones from any calendar source in the iCal format into GroupWise calendars. This can be sources on the Internet, like www.icalshare.com, or your Novell Teaming server. The content of a single source, like a folder on the Teaming server, can be synchronized with calendars of any number of users, even if their mailboxes are located on different GroupWise post offices. Also, the contents of several folders located on different source servers can be synchronized into the mailbox of a single user.

Tasks on your Novell Teaming server can be synchronized into GroupWise as tasks or as events, depending on which type of displaying is better for the respective folder. You can also synchronize milestones from your Novell Teaming server, however the GroupWise calendar doesn't include an appropriate type of item for milestones. Therefore, milestones are displayed as daylong events on the day, the date of which is provided in the "Until" field.

You can test iCal2GW without any risk on a live mailing system; at worst, an undesired calendar will be created in the mailbox, on which the application will be tested, and such calendar can be easily removed.

Here are some of the features and benefits:

- Replication of calendar, tasks and milestones from Novell T+C into any number of GroupWise accounts;
- Replication of static calendars in the iCal format, such as bank holidays, vacations, etc.;
- Automatic creation of calendar items in GroupWise user accounts;
- Calendar folders are in a standard form and can be replicated to mobile devices and shared and rules can be applied;
- When removing a calendar folder, a new folder is automatically recreated and the replication goes on (upon reset of a correlation database all contents are replicated);
- Possibility to synchronize between multiple Teaming and GroupWise servers;
- Settings can be created for all users, users of selected post offices, members on selected distribution sheets;

- The best way of distribution of calendar events to multiple users;
- Running on a server; requires JAVA SUN 1.6.0_11 or higher; can run on a GW server.

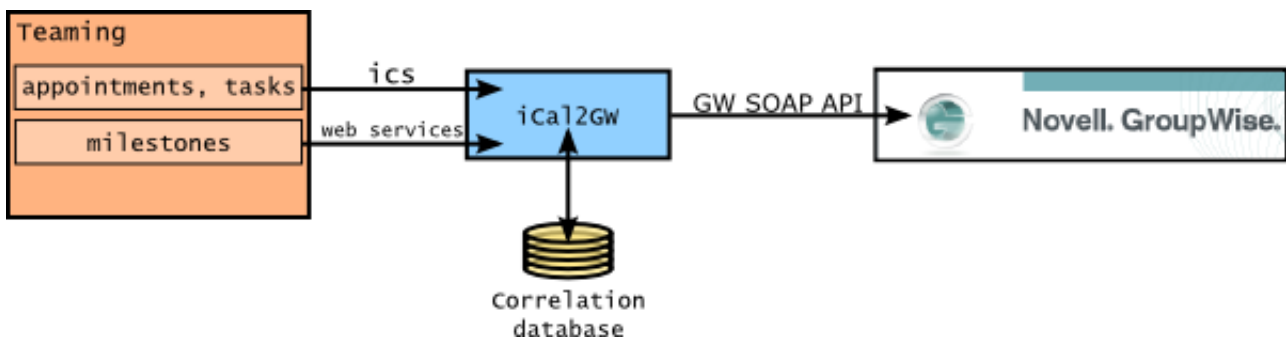
Correlation Database

The synchronization is performed periodically and always just once for every item (a calendar event, a task or a milestone). Upon successful transfer of an item, a record is made in the correlation database, and – during the next synchronization event – it is only checked whether any change has been done with the respective item on the source server.

The correlation database is stored in the folder containing the application in the sub-folder `corr`. Each user set in the configuration file has a sub-folder created in `corr` automatically with the respective user name and related database files.

If database files are deleted, all records will first be removed from the GroupWise calendar, then checked, and then transferred again in the database upon the next synchronization session.

Scheme of the application activity



Note – The synchronization is carried out in one direction only, which means that changes performed in GroupWise shall not be transferred to the source calendar. However, changes to and/or removals made in the source folder shall be seen in the GroupWise calendar.

Installation

System requirements:

Java Sun JRE/JDK 1.6.0_11 or newer; GroupWise 8.0 with Hot Patch 1 or newer; approximately 20 MB of free disk space; Teaming 1 or Teaming 2

The installation is very easy. Just copy the file called `ical2gw_2.8.bin` being a part of the archive, in which the application is distributed, to the server; using the command `chmod +x ical2gw_2.8.bin` add the authorization to launch and start it. The installation script will check whether the respective version of Java JRE/JDK is available and allow picking a path, where the application should be

installed.

Note – You can install the application manually without using the installation script. If you launch the ical2gw_2.8.bin file with the "-x" parameter, its content shall load into the current folder. Next you have to proceed according to the manual – section Manual Installation Instructions.

To allow the application accessing GroupWise calendars without entering login data of individual users, it has to be registered as a "Trusted Application" on the GW server. This can be done, for example, using the GWTrustedApp.exe utility (in GWTrustedApp directory). After launching it, a path to the folder of the primary domain database is entered and confirmed. The utility then displays the respective TrustedKey (for example:

21A88AA102E50000B18DAD009E00C70021A88AA202E50000B18DAD009E00C700),

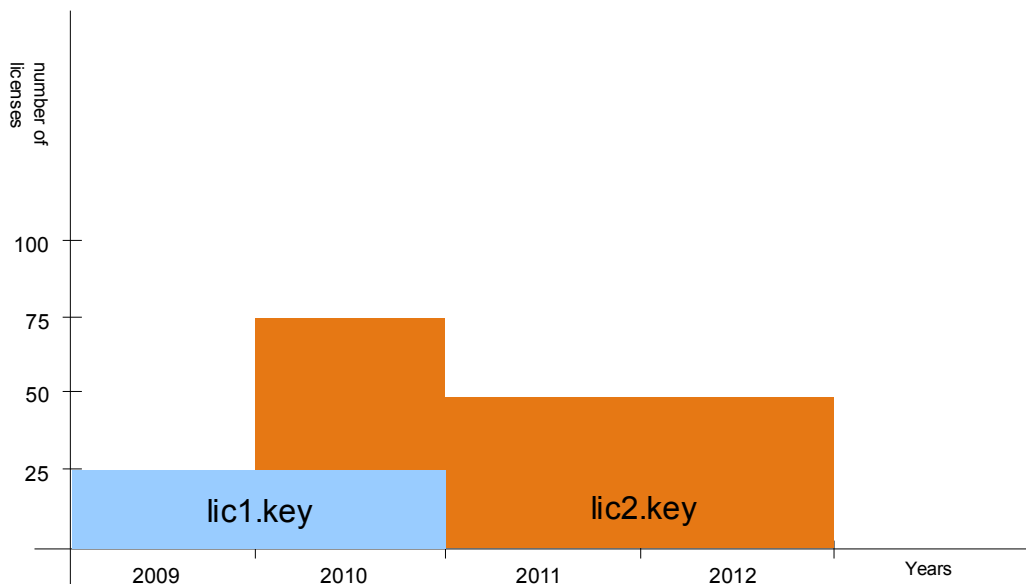
which must be entered into the configuration file (see Configuration); at the same time, the application is registered for the respective domain, which can be verified using, for example, ConsoleOne.

Licensing

The number of required licenses is based on the number of source items on the Novell Teaming portal or ics files, from which the synchronization should be performed – 1 source item = 1license. The license file(s) must be stored in the folder, in which the application is installed, in the licenses subfolder, or – respectively – in a folder defined in the configuration file (see Configuration).

Unless a license file is found in the respective folder upon the first start, the application shall be launched in the trial mode – the synchronization will be carried out, but the text TRIAL* will be entered into the "subject" field of all items transferred to GroupWise. The trial mode will be terminated automatically after the application detects any valid license file in the respective folder.

A single license file may contain any number of licenses. If more license files are stored in the respective folder, the number of licenses is accumulated. It means that you can simply increase the number of synchronized source items without any harm to licenses issued before.



The picture depicts a situation, in which the license file called `lic1.key` containing 25 licenses for a period of two years was bought at first. During the first year, the number of source folders to be synchronized was extended by another 50, thus the second license file called `lic2.key` was purchased containing 50 licenses for the period of 3 years. The total number of licenses grew to 75, but only for the period for which the original licensing file was in effect. At the moment of expiry of the original file the total number of active licenses dropped to 50 (only the second license file will be in effect). To maintain the synchronization from all source folders, it would be necessary to purchase another 25 licenses again (or as many as required to synchronize the content of all source folders).

Unless the folder contains a sufficient number of licenses for all source folders, the content of folders, to which no licenses apply, will not be synchronized and the folder names will be recorded in the log. Licenses are allocated to source folders step-by-step, and not necessarily in the sequence, in which they are listed in the configuration file. If in the case of too few licenses it is necessary to pick manually any folders not to be synchronized, just add the option `disabled="true"` in the configuration file in the source folders parameter `disabled="true"`, for example: `<ics id="MYcal" connection="teamingPortal_A" disabled="true">`.

Note – License files can be added while the application is running; the number of licenses is checked before every synchronization event.

In the user box, which is set as `adminAccountId` in the configuration file, messages will be automatically created notifying of approaching expiry of any license file (thirty day in advance for the first notice, and then one day before the expiry another one). A warning will be created also if any license file with expired validity or any invalid or damaged file is stored in the license folder – in such cases the message will be created every day (or as soon as the application is started).

Configuration

The application is configured using the configuration xml file (see Appendices), the name of which is used as a parameter when the application is launched. In the initiation script, the name `config.xml` is used as default. The file must be stored in the application folder. The names of parameters in the configuration file are case sensitive!

Note – For any potential configuration changes to apply, the application has to be restarted after the configuration file is modified.

A detailed description of parameters of the configuration file:

```
<param name="basePath" value="/opt/tdp/ical2gw/" />
```

The path to the location, in which the application is stored

```
<param name="timeZoneID" value="Europe/Prague" />
```

Time zone – a parameter important for the correct setting of time for all-day items transferred into GroupWise.

```
<param name="workerThreads" value="2" />
```

A custom and optional parameter that allows setting the synchronization processing in more threads (the value determines the number of threads) – allowing faster processing during synchronization from many sources on servers with multi-core processors. Unless it is set, the synchronization is carried out in a standard manner in a single thread.

```
<param name="adminAccountId" value="GWuser1" />
```

The `adminAccountId` parameter determined one of the synchronized GroupWise accounts, in which the application will create messages about potential expiry of licenses (see Licensing) or about synchronization not being performed. If an account with the user name "admin" is one of the synchronization accounts, the `adminAccountId` parameter does not have to be set – messages will be automatically created on the admin account. However, if an account with a different user name is set as `adminAccountId`, messages will be created on such account, although the synchronization will be performed also on the admin account. The content of the news can be modified using templates (see Message Templates).

```
<param name="licensePath" value="/opt/lic" />
```

An optional parameter, which sets the path to the folder with license files. Unless the path is set the default path from the `basePath` parameter will apply (in such case, files must be stored in the given path in the `licenses` subfolder).

```
<param name="updateDelay" value="60" />
```

The waiting interval between individual synchronization sessions – entered in seconds. Unless the parameter is set, the default value of 30 will be used.

```
<param name="watchdogTimeout" value="1800"/>
```

The interval, in which it is checked whether the iCal2GW application is running. It is set in seconds. Unless the synchronization is carried out during the interval, which consists of the stated value and the value set for `updateDelay` (i.e. 1860 in this case), an e-mail message will be created in the administrator's box (or, respectively, on the account set in the `adminAccountId` parameter) containing information about the fact that the synchronization is not carried out (the content of the message can be adjusted in the respective template – see Message Templates).

Note – The monitoring the process concerns the activities of the iCal2GW application only, and not, for example, whether the synchronization is not carried out due to the malfunction of the Teaming portal or the GroupWise server!

```
<connections>
  <gwSOAP id="Gwserv1" address="http://172.22.30.71" port="7191" trustedAppName="iCal2GW"
trustedAppKey="3E519CF40DF30000823F9C00A00038003E519CF50DF30000823F9C00A0003800"/>
  <gwSOAP id="Gwserv2" address="http://172.22.30.81" port="7191" trustedAppName="iCal2GW"
trustedAppKey="F30000823F9C00A00B41038000038003E01519CF503E519CF40DF30000823F9"/>
  <teamingWS id="teamingPortal_A" address="http://172.22.30.60" port="80" username="admin"
password="pswd" algorithm="MD5"/>
  <teamingWS id="teamingPortal_B" address="http://168.90.25.61" port="80" username="admin2"
password="pswd2" algorithm="SHA"/>
</connections>
```

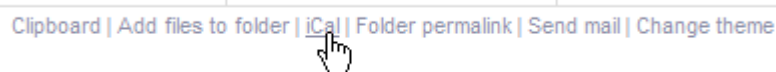
In the "connections" section, paths to the GroupWise and Teaming servers are provided. For GroupWise servers a `trustedAppKey` must be set, which shall allow the application to work with GroupWise as a "trusted application" (see Installation). The `trustedAppName` parameter is set as the name of the application used upon the registration into the domain database (iCal2GW). For the Teaming portal, login data related to the user account with the administrator rights are provided here. The `algorithm` parameter is defined algorithm used to encrypt authentication data. Algorithm is selected during the Teaming installation (i.e. in Teaming 1.0 is the default MD5 algorithm, SHA is default for Teaming 2.0). If is set wrong algorithm, iCal2GW will not be able to login to the portal and perform synchronization. If there are more than one source or target servers they are distinguished by IDs.

```
<sources>
  <ics id="MYcal" connection="teamingPortal_A">
    <url><![CDATA[http://172.22.30.60/ssf/ical/basic.ics?
zn=liferay.com&bi=161&ui=22&pd=4d55e228ea391d4ad5ab6babddd0e4dc93d86236]]></url>
  </ics>
  <ics id="TEAMtasks" connection="teamingPortal_B">
    <url><![CDATA[http://168.90.25.61/ssf/ical/basic.ics?
zn=liferay.com&bi=165&ui=22&pd=67933293d9ce7dc6630bcf0a7e278795fde57505]]></url>
  </ics>
  <teamingFolder connection="teamingPortal_A" id="MYmilestones" binderId="166"/>
</sources>
```

In the "sources" section, specific source folders are defined, the contents of which should be synchronized. The value of the ID parameter is then provided as a value of the source parameter `calendar_source` in the section for setting target calendars in GroupWise.

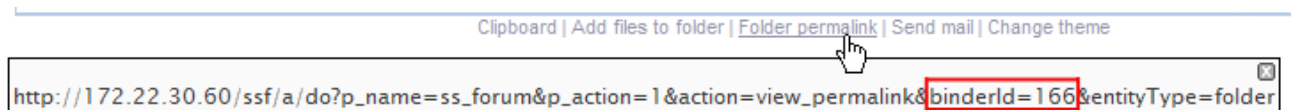
Each folder can have an appropriate source server defined using the `connection` parameter. If there is one source server only, the parameter does not have to be used (however, no ID parameter should be set for the source server in the `connection` section).

The Url address in the tag `<url><![CDATA[http://172.22.30.60/ssf/...` represents a link to the iCal folder (tasks or calendar) in the Teaming portal. The address can be obtained upon right clicking on the iCal link shown below the detail of the respective calendar or tasks folder, which should be synchronized, and choosing the `Copy link location` option in the context menu. The address can then be inserted into the configuration file to the appropriate line using, for example, the `Ctrl+V` key combination.



Note – The URL address can also refer to a calendar in the ics format published on a web server (such as Google Calendar).

In the case of the milestones folder, which is transferred in a different manner, the `binderId` is used instead of a url address. It is a part of a permanent link to the folder, which is displayed upon clicking on `Permanent link to folder`.



If any source item has to be temporarily excluded from the synchronization process (for example, due to an insufficient number of licenses), you can add the value `disabled="true"` into the respective parameter of the source folder, for example:

```
<ics id="MYcal" connection="teamingPortal_A" disabled="true">
```

To renew the synchronization just remove this option, or change the value to `disabled="false"`.

Using a locally store ics file as a synchronization source

A calendar file in the ics format stored locally in the "in" folder (to be found in the folder, in which the applications is installed) can also be used as a synchronization source. Like calendars loaded from web servers, source files are defined in the "sources" section.

```
<ics id="UKholidays" fileName="UK32Holidays.ics"/>
```

The `fileName` parameter is used to define the name of the source file.

Note – The calendar, which should contain data loaded from a locally stored file, will be created at the moment when the file with the respective name is found by the application in the "in" folder.

```

<targets>
  <gwAccount id="GWuser1" connection="Gwserv1">
    <calendars>
      <calendar source="MYcal" name="calendar1" color="#55FF60"/>
      <calendar source="MYmilestones" name="calendar2" color="#A03030"/>
      <calendar source="TEAMtasks" name="calendar3" color="#FFB070"/>
    </calendars>
  </gwAccount>

  <gwAccount id="GWuser2" connection="Gwserv2">
    <calendars>
      <calendar source="TEAMtasks" name="calendarA" color="#FFB070"
tasksAsAppointments="true"/>
      <calendar source="UKholidays" name="Holidays" color="#100fff" sync="full"/>
    </calendars>
  </gwAccount>
</targets>

```

A user name for the respective GroupWise account is used as `gwAccount id`. The `connection` parameter is used to select the respective GroupWise server using its ID, as defined in the "connections" section (if only one server is used, the parameter does not have to be used; however, no ID parameter should be set in the `connection` section).

The `calendar source` parameter is used to assign the calendar in GroupWise to the source folder defined in the "sources" section (using the appropriate ID). The value of the `name` parameter is determined by the name of the calendar in GroupWise. If the calendar with the respective name does not exist in the folder of the user, it will be created automatically. The `color` parameter is used to set a hexadecimal code of the GroupWise calendar color. The color can be changed later in the client.

The optional parameter `tasksAsAppointments="true"` can be set to have tasks from the Teaming portal transferred into the appropriate GroupWise calendar as common meetings. If the parameter is not set at all or its value is set on "false", tasks will be transferred in a standard manner (tasks are transferred including the information about the progress in percents and the "completed" attribute).

Another optional parameter `sync="full"` can be used to set a full synchronization of a target calendar within every cycle. Unlike the standard manner, in which only changes found in the source folder or file are transferred, the content of the target calendar is first deleted and then filled with data from the respective source. If the parameter is not set at all or its value is set on "incremental", the synchronization is carried out in a standard manner.

Calendar Collections

If it is necessary to synchronize several identical calendars into boxes, you can use the option of defining a collection of source calendars.

```
<collections>
  <collection id="testCollection">
    <calendars>
      <calendar source="MYcal" name="calendar1" color="#55FF60"/>
      <calendar source="MYmilestones" name="calendar2" color="#A03030"/>
      <calendar source="TEAMtasks" name="calendar3" color="#FFB070"/>
    </calendars>
  </collection>
</collections>
```

Calendars are added into a collection in the same manner, in which they are listed individually with a GroupWise account. Subsequently, they are assigned to accounts using the respective `collection id`.

```
<targets>
  <gwAccount id="GWuser1" connection="Gwserv1" collections="testCollection"/>
</targets>
```

In the account heading, provide the `collection` parameter with the name of the respective collection. Any and all calendars contained in the group will be synchronized with the respective account. A single account can have several groups assigned at the same time with their names separated by a comma in the parameter. Also, individual source calendars can be assigned along with groups to an account in the standard manner described above.

Synchronization into mailboxes defined by distribution lists, post offices or intended for all system users

Source calendars can be synchronized also into mailboxes defined by distribution lists, for all users of the selected post office, or even for all users of the GroupWise system. Therefore, it is not necessary to set the configuration file and restart the application every time another user is added with a mailbox, into which the synchronization should be performed.

Just like in the case of synchronization into specifically listed mailboxes (`gwAccount id`), the target calendars can be defined by a list or using a collection (see Calendar Collections).

```
<targets>
  <gwDistributionList id="DList1" connection="Gwserv2" userId="GWuser2">
    <calendars>
      <calendar source="MYcal" name="calendar1" color="#55FF60"/>
      <calendar source="MYmilestones" name="calendar2" color="#A03030"/>
      <calendar source="TEAMtasks" name="calendar3" color="#FFB070"/>
    </calendars>
  </gwDistributionList>
</targets>
```

The name of the distribution list to be used to define target mailboxes is specified in the "targets" section in the `gwDistributionList id` parameter. The `userId` parameter must include a user with at least a reading access to the respective distribution list.

Note – The "connection" must state the `gwSoap id` of the post office containing the respective distribution list. Otherwise, the list won't be found and the synchronization into the specified accounts will not be performed!

```
<targets>
<gwPostOffice connection="Gwserv1" userId="Gwuser1">
  <calendars>
    <calendar source="MYcal" name="calendar1" color="#55FF60"/>
    <calendar source="MYmilestones" name="calendar2" color="#A03030"/>
    <calendar source="TEAMtasks" name="calendar3" color="#FFB070"/>
  </calendars>
</gwPostOffice>
</targets>
```

If it is necessary to carry out the synchronization into all mailboxes within a post office, the `gwPostOffice` should be provided in the "targets" section. The `gwSoap id` of the respective post office must be set at the value of the "connection" parameter. A user with an active account in the respective post office must be set as `userId`.

```
<targets>
<gwAllUsers connection="Gwserv1" userId="Gwuser1">
  <calendars>
    <calendar source="MYcal" name="calendar1" color="#55FF60"/>
    <calendar source="MYmilestones" name="calendar2" color="#A03030"/>
    <calendar source="TEAMtasks" name="calendar3" color="#FFB070"/>
  </calendars>
</gwAllUsers>
</targets>
```

The synchronization can be performed using the `gwAllUsers` parameter also into mailboxes of all users within the entire system. A user with an active account in the respective system must be set as `userId`.

Launching the Application

The application is run on the server as a service (daemon) and is launched automatically when the server is started. If it needs to be launched, terminated or restarted manually, you can do it using the command `/etc/init.d/iCal2GW start (or stop/restart)`

Message Templates

Templates define the content of messages to inform an authorized person about the expiring effectiveness / invalidity of license files and about potential malfunctioning of the application (see Licensing and Configuration sections). They are located in the application folder, in the `templates` subfolder. One template is available for each message type: `expiringLicense.flt` for the information about the expiring validity of a license file; `expiredLicense.flt` for the information about the expired validity of a license file; `invalidLicense.flt` for the information about an invalid or damaged license file; and `watchdogTimeout.flt` for the information about synchronization not being carried out. Files are created in the UTF-8 format without BOM and can

be modified in any text editor supporting this format.

Log

The application creates records of activities and saves them in files in the logs folder, or – respectively – in the folder set in the logback.xml file (the default setting is: /opt/tdp/ical2gw/logs.). Two files are created automatically for every day with named containing the date of the respective day (such as ical2gw-2009-01-06) and filename extensions of .log for records on common application procedures, and .err containing only potential errors and problems to be found easily if needed.

Appendices

Sample configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<cfg>

<param name="basePath" value="/opt/tdp/ical2gw/" />
<param name="logbackConfig" value="/opt/tdp/ical2gw/logback.xml" />
<param name="timeZoneID" value="Europe/Prague" />
<param name="adminAccountId" value="GWuser1" />
<param name="licensePath" value="/opt/lic" />
<param name="updateDelay" value="60" />
<param name="watchdogTimeout" value="1800" />

<connections>
  <gwSOAP id="GWServ1" address="http://172.22.30.71" port="7191" trustedAppName="iCal2GW"
trustedAppKey="3E519CF40DF30000823F9C00A00038003E519CF50DF30000823F9C00A0003800" />
  <gwSOAP id="GWServ2" address="http://172.22.30.81" port="7191" trustedAppName="iCal2GW"
trustedAppKey="F30000823F9C00A00B41038000038003E01519CF503E519CF40DF30000823F9" />
  <teamingWS id="teamingPortal_A" address="http://172.22.30.60" port="80" username="admin"
password="pswd" / algorithm="MD5">
  <teamingWS id="teamingPortal_B" address="http://168.90.25.61" port="80" username="admin2"
password="pswd2" / algorithm="SHA">
</connections>

<sources>
  <ics id="MYcal" connection="teamingPortal_A">
    <url><![CDATA[http://172.22.30.60/ssf/ical/basic.ics?
zn=liferay.com&bi=161&ui=22&pd=4d55e228ea391d4ad5ab6babddd0e4dc93d86236]]></url>
  </ics>
  <ics id="TEAMtasks" connection="teamingPortal_B">
    <url><![CDATA[http://168.90.25.61/ssf/ical/basic.ics?
zn=liferay.com&bi=165&ui=22&pd=67933293d9ce7dc6630bcf0a7e278795fde57505]]></url>
  </ics>
  <teamingFolder connection="teamingPortal_A" id="MYmilestones" binderId="166" />
  <ics id="UKholidays" fileName="UK32Holidays.ics" />
</sources>

<targets>
  <gwAccount id="GWuser1" connection="GWServ1">
    <calendars>
      <calendar source="MYcal" name="calendar1" color="#55FF60" />
      <calendar source="MYmilestones" name="calendar2" color="#A03030" />
      <calendar source="TEAMtasks" name="calendar3" color="#FFB070" />
    </calendars>
  </gwAccount>
  <gwAccount id="GWuser2" connection="GWServ2">
    <calendars>
      <calendar source="TEAMtasks" name="calendarA" color="#FFB070" />
      <calendar source="UKholidays" name="calendarB" color="#100fff" />
    </calendars>
  </gwAccount>
</targets>
</cfg>
```

Example of using a group of calendars

```
<collections>
  <collection id="testCollection">
    <calendars>
      <calendar source="MYcal" name="calendar1" color="#55FF60"/>
      <calendar source="MYmilestones" name="calendar2" color="#A03030"/>
      <calendar source="TEAMtasks" name="calendar3" color="#FFB070"/>
    </calendars>
  </collection>
</collections>

<targets>
  <gwAccount id="GWuser1" connection="Gwserv1" collections="testCollection"/>
</gwAccount>
</targets>
```

Note – Values indicated in the same color represent IDs of related connections. Unless the application finds corresponding counterparts in the configuration folder when launched, it notifies the user of missing connections and closes.

Manual Instalation Instructions

```
# for iCal2GW 2.8 on Linux (tested on SLES 10 SP1, but should work on other Linuxes too)
# login as root/superuser
# copy ical2gw_2.8.bin into the current directory
# download Sun JRE/JDK 1.6_11 or newer 'jre-6u13-linux-i586.rpm.bin'
```

```
# JAVA installation:
```

```
chmod +x jre-6u13-linux-i586.rpm.bin
./jre-6u13-linux-i586.rpm.bin
```

```
# follow the installation instructions (use space bar to scroll through the the license agreement faster)
```

```
# iCal2GW installation:
```

```
chmod +x ical2gw_2.8.bin
./ical2gw_2.8.bin -x
```

```
# this will extract installation files in current directory into 'ical2gw'
```

```
cp -r ical2gw/opt/* /opt/
cp ical2gw/etc/init.d/ical2gw /etc/init.d/
```

```
chmod +x /etc/init.d/ical2gw
```

```
# edit /etc/init.d/ical2gw:
```

```
#
```

```
# JAVA_HOME - set to path where jre is installed (probably something like
'/usr/java/jre1.6.0_13' or '/usr/java/current')
```

```
#
```

```
# in case you installed it to other location than /opt/tdp/ical2gw , also check/change following:
```

```
# APP_DIR in /etc/init.d/ical2gw
```

```
# basePath parameter in config.xml (/opt/tdp/ical2gw/config.xml)
```

```
# logging paths in logback.xml (contents of FileNamePattern tag - change only the path before
/logs/ical2gw-%d{... } )
```

```
# install the ical2gw service (daemon)
```

```
chkconfig ical2gw on
```

```
# start it
```

```
/etc/init.d/ical2gw start
```

Start script

(File called ical2gw)

```
#!/bin/sh
#
# System startup script for iCal2GW
#
### BEGIN INIT INFO
# Provides: ical2gw
# Required-Start: $local_fs $remote_fs
# X-UnitedLinux-Should-Start: $named $syslog $time
# Required-Stop: $local_fs $remote_fs
# X-UnitedLinux-Should-Stop: $named $syslog $time
# Default-Start: 3 5
# Default-Stop: 0 1 2 6
# Short-Description: iCal2GW daemon
# Description: Start iCal2GW daemon
### END INIT INFO
APP_DIR="/opt/tdp/ical2gw"

appIsRunning()
{
    app_ps_log=`mktemp /var/tmp/app-ps.log.XXXXXX`
    ps aux --cols 1024 >"$app_ps_log"
    app_is_running="false"
    if grep "jar ${APP_DIR}/iCal2GW.jar" "$app_ps_log" >/dev/null 2>/dev/null ; then
        app_is_running="true"
    fi
    rm -f "$app_ps_log"
    test "$app_is_running" = "true"
}

#save JAVA_HOME before it gets overwritten
OLD_JAVA_HOME=$JAVA_HOME
OLD_JAVA_OPTS=$JAVA_OPTS

JAVA_HOME=/usr/java/jre1.6.0_13

# test whether JAVA_HOME and JAVA_OPTS were overwritten ... if empty now, reset to old
values
test -n "$JAVA_HOME" || JAVA_HOME=$OLD_JAVA_HOME
test -n "$JAVA_OPTS" || JAVA_OPTS=$OLD_JAVA_OPTS

test -r "$APP_DIR/iCal2GW.jar" || { echo "$APP_DIR/iCal2GW.jar not installed"; }
test -x "$JAVA_HOME/bin/java" || { echo "$JAVA_HOME/bin/java not installed"; }

# pid set?
test -n "$APP_PID" || APP_PID="/var/run/ical2gw.pid"

# the following variables affects the server
export APP_PID APP_DIR JAVA_HOME JAVA_OPTS

# Shell functions sourced from /etc/rc.status:
# rc_check check and set local and overall rc status
# rc_status check and set local and overall rc status
# rc_status -v ditto but be verbose in local rc status
# rc_status -v -r ditto and clear the local rc status
# rc_failed set local and overall rc status to failed
# rc_failed <num> set local and overall rc status to <num><num>
```

```

# rc_reset      clear local rc status (overall remains)
# rc_exit       exit appropriate to overall rc status
. /etc/rc.status

# First reset status of this service
rc_reset

# Return values acc. to LSB for all commands but status:
# 0 - success
# 1 - generic or unspecified error
# 2 - invalid or excess argument(s)
# 3 - unimplemented feature (e.g. "reload")
# 4 - insufficient privilege
# 5 - program is not installed
# 6 - program is not configured
# 7 - program is not running
#
# Note that starting an already running service, stopping
# or restarting a not-running service as well as the restart
# with force-reload (in case signalling is not supported) are
# considered a success.

case "$1" in
start)
    echo -n "Starting iCal2GW ($APP_DIR)"
    ## Start daemon with startproc(8). If this fails
    ## the echo return value is set appropriate.
    # NOTE: startproc return 0, even if service is
    # already running to match LSB spec.
    if appIsRunning ; then
        rc_failed 0
    else
        cmd="$JAVA_HOME/bin/java -jar $APP_DIR/iCal2GW.jar $APP_DIR/config.xml"
        rm -f $APP_PID
        nohup $cmd >/dev/null 2>&1 & echo $! >$APP_PID
        sleep 1
        if appIsRunning ; then
            rc_failed 0
        else
            rc_failed 7
        fi
    fi
    rc_status -v
;;
stop)
    echo -n "Shutting down iCal2GW ($APP_DIR)"
    ## Stop daemon with killproc(8) and if this fails
    ## set echo the echo return value.
    if appIsRunning ; then
        kill `cat $APP_PID`
        # wait 60 sec for stop at maximum
        wait_sec=60
        while [ "$wait_sec" != "0" ] ; do
            sleep 1
            if ! appIsRunning ; then
                # the webmon server is stoped, end the loop
                wait_sec=0
                break
            fi
        done
    fi

```

```

        wait_sec=$((wait_sec -1))
    done
fi

# if tomcat is __still__ running, try it with kill -9
if appIsRunning ; then
    kill -9 `cat $APP_PID`
    # wait 60 sec for stop at maximum
    wait_sec=60
    while [ "$wait_sec" != "0" ] ; do
        sleep 1
        if ! appIsRunning ; then
            # theTomcat server is stoped, end the loop
            wait_sec=0
            break
        fi
        wait_sec=$((wait_sec -1))
    done
fi

# check the final status
if appIsRunning ; then
    rc_failed 1
else
    rc_failed 0
fi

rc_status -v
;;
try-restart)
    ## Stop the service and if this succeeds (i.e. the
    ## service was running before), start it again.
    ## Note: try-restart is not (yet) part of LSB (as of 0.7.5)
    $0 status >/dev/null && $0 restart

    # Remember status and be quiet
    rc_status
    ;;
restart)
    ## Stop the service and regardless of whether it was
    ## running or not, start it again.
    $0 stop
    $0 start

    # Remember status and be quiet
    rc_status
    ;;
force-reload)
    ## Signal the daemon to reload its config. Most daemons
    ## do this on signal 1 (SIGHUP).
    ## If it does not support it, restart.

    echo -n "Reload service iCal2GW $($APP_BASE)"
    ## if it supports it:
    #killproc -HUP $TOMCAT_BIN
    #touch /var/run/FOO.pid
    #rc_status -v

    ## Otherwise:

```

```

$0 stop && $0 start
rc_status
;;
reload)
## Like force-reload, but if daemon does not support
## signalling, do nothing (!)

# If it supports signalling:
#echo -n "Reload service FOO"
#killproc -HUP $TOMCAT_BIN
#touch /var/run/FOO.pid
#rc_status -v

## Otherwise if it does not support reload:
rc_failed 3
rc_status -v
;;
status)
echo -n "Checking for iCal2GW ($APP_DIR)"
## Check status with checkproc(8), if process is running
## checkproc will return with exit status 0.

# Status has a slightly different for the status command:
# 0 - service running
# 1 - service dead, but /var/run/ pid file exists
# 2 - service dead, but /var/lock/ lock file exists
# 3 - service not running

# NOTE: checkproc returns LSB compliant status values.
if appIsRunning ; then
    rc_failed 0
else
    rc_failed 3
fi
rc_status -v
;;
probe)
## Optional: Probe for the necessity of a reload,
## give out the argument which is required for a reload.
;;
*)
echo "Usage: $0 {start|stop|status}"
exit 1
;;
esac
rc_exit

```

Third Party Licenses

dom4j

Copyright 2001-2005 (C) MetaStuff, Ltd. All Rights Reserved.

Redistribution and use of this software and associated documentation ("Software"), with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain copyright statements and notices. Redistributions must also contain a copy of this document.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name "DOM4J" must not be used to endorse or promote products derived from this Software without prior written permission of MetaStuff, Ltd. For written permission, please contact dom4j-info@metastuff.com.
4. Products derived from this Software may not be called "DOM4J" nor may "DOM4J" appear in their names without prior written permission of MetaStuff, Ltd. DOM4J is a registered trademark of MetaStuff, Ltd.
5. Due credit should be given to the DOM4J Project - <http://www.dom4j.org>

THIS SOFTWARE IS PROVIDED BY METASTUFF, LTD. AND CONTRIBUTORS ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL METASTUFF, LTD. OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

FreeMarker

Copyright (c) 2003 The Visigoth Software Society. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. The end-user documentation included with the redistribution, if any, must include the following acknowledgement: "This product includes software developed by the Visigoth Software Society

(<http://www.visigoths.org/>).". Alternately, this acknowledgement may appear in the software itself, if and wherever such third-party acknowledgements normally appear.

3. Neither the name "FreeMarker", "Visigoth", nor any of the names of the project contributors may be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact visigoths@visigoths.org.
4. Products derived from this software may not be called "FreeMarker" or "Visigoth" nor may "FreeMarker" or "Visigoth" appear in their names without prior written permission of the Visigoth Software Society.

THIS SOFTWARE IS PROVIDED ``AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE VISIGOTH SOFTWARE SOCIETY OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This software consists of voluntary contributions made by many individuals on behalf of the Visigoth Software Society. For more information on the Visigoth Software Society, please see <http://www.visigoths.org/>

iCal4j

Copyright (c) 2008, Ben Fortuna. All rights reserved.

- Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Ben Fortuna nor the names of any other contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF

MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.