opentext™

GroupWise® Software Developer Kit

Web Services Events

April 2024

Legal Notices

The only warranties for products and services of Open Text and its affiliates and licensors (“Open Text”) are as may be set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Open Text shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Contents

About This Guide

GroupWise Web Services Events is an extension of GroupWise Web Services (SOAP) and provides access to events or actions that occur on a GroupWise user's mailbox.

IMPORTANT: Unless otherwise marked, the features in GroupWise Web Services Events work with GroupWise 8 and later versions.

This guide contains the following sections:

Audience

This guide is intended for GroupWise developers.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comment feature at the bottom of each page of the online documentation.

1 Overview

GroupWise Web Services Events is an extension of GroupWise Web Services that provides access to events or actions that occur on a GroupWise user's mailbox. Using events, developers can be notified in real time when a specific event occurs on a mailbox. In other words, GroupWise Web Services Events is a Web Service (similar to GroupWise Web Services) that allows you to programmatically configure and retrieve specific GroupWise events that have occurred on a user's mailbox.

For example, if you want to track when an item is deleted in a GroupWise user's mailbox, you create an Event Configuration that tracks deleted items. During the configuration, you can provide an IP address and a port on which to be notified when the specific event occurs. After you're notified, you can then take the appropriate action.

It is important to note that the events are only tracked on the user's mailbox. A user can have shared folders and shared personal address books that are shared from another user. When an item is changed in one of these folders or personal address books, the change is happening on the other user's mailbox. The event will be recorded in that user's mailbox. There will not be an event for the user that has the shared folder or personal address book shared with them. For example, User A has a folder (FolderX) shared to him from User B. An item (ItemY) is deleted from that folder. User A will not get an event. User B will get an event saying that ItemY was deleted.

GroupWise Web Services Events uses industry standards such as XML, SOAP, and HTTP for requests and responses from the GroupWise POA. The GroupWise Events schema defines the methods and objects that are used with the GroupWise POA. The GroupWise Web Services Events schema definition file (WSDL) provides the tools you need to hook into IDE frameworks that support Web Services.

GroupWise Web Services provides access to read and modify user's mailbox data through the same industry standards that are used in GroupWise Web Services Events. GroupWise Web Services Events and GroupWise Web Services can and should be used together. To learn more about GroupWise Web Services, see *GroupWise SDK: Web Services (SOAP)*.

Resources

Before using GroupWise Web Services Events, you should be familiar with XML, SOAP, and HTTP concepts. You can find more information at the following links:

- GroupWise Documentation (http://www.novell.com/documentation/groupwise.html)

- HTTP (http://www.w3.org/Protocols/Specs.html)
- SOAP Part 0 (http://www.w3.org/TR/soap12-part0/)
- SOAP Part 1 (http://www.w3.org/TR/soap12-part1/)
- SOAP Part 2 (http://www.w3.org/TR/soap12-part2/)
- XML Schema Specification (http://www.w3.org/XML/Schema)
- XML Schema Specification Part 0 (http://www.w3.org/TR/xmlschema-0)
- XML Schema Specification Part 1 (http://www.w3.org/TR/xmlschema-1)
- XML Schema Specification Part 2 (http://www.w3.org/TR/xmlschema-2)
- XML (http://www.w3.org/XML/)
- WSDL (http://www.w3.org/TR/wsdl)

WSDL and Schema Files

GroupWise Web Services are defined in the following XML schema files:

- `types.xsd` defines Web Services.
- `methods.xsd` defines Web Services.
- `events.xsd` defines events and references `types.xsd` also.

The WSDL file (`groupwise.wsdl`) imports all of the above schema files and can be imported into a development framework that supports Web Services. The framework parses the WSDL and schema files and creates a code library that can be used to access Web Services and GroupWise Web Services Events.

Frameworks simplify your work as a developer because the generated library abstracts the complexities of HTTP, SOAP, and XML. You can then concentrate on programming without worrying about the complexities of transport and XML parsing.

However, if you want complete control, you can handle HTTP, SOAP, and XML directly. GroupWise Web Services handles valid requests, no matter how each request is generated.

WSDL and Schema Location

After you have unzipped the files in the documentation, look for the GroupWise WSDL and schemas directory. In the directory structure, you'll find the different WSDL and schemas for GroupWise 8 and later versions.

The GroupWise WSDL and schemas are not available with the GroupWise install distribution. They are available only in the GroupWise Web Services documentation.

The .Net version of the schemas can be found in the subdirectory named MS.NET.

Event Life Cycle

This section helps you learn GroupWise Web Services Events from the perspective of an event listener application and the GroupWise POA. It covers the following topics:

Application Perspective

Applications can track events for one or many GroupWise users. In order to call any of the event methods, you first need to call loginRequest on a specific user. The event methods will apply to that user.

To track events, an application should perform the following basic steps:

1. Call the configureEventsRequest method to set up event configurations for each GroupWise user.

   In this step, applications tell the POA which event types they want to track. The configureEventsRequest method also tells the POA if the application should be notified when an event takes place.

   Applications should listen for notifications. Listening reduces network traffic and reduces the workload on the POA. If notifications are not used, the POA might need to handle dredging requests from applications for many users (which increases the work load on the POA).

   However, applications are not required to listen for notifications. An application can periodically call the getEventsRequest method to get a list of changes for a user.

2. If notifications are enabled, the application listens for notifications from the POA.

   A notification provides a userid and configuration key. A notification looks like the following:

   ```
   <notify>
      <userid>username</userid>
      <key>Key Name</key>
   </notify>\r\n
   ```

   Each notification ends with the `\r\n` characters.

3. When the application receives a notification from the POA, the application can spin off a worker thread to call getEventsRequest.

   The response from the getEventsRequest method provides additional information such as the event type, item ID, and time the item changed.

4. If the application requires more information, it calls getItemsRequest to retrieve the details for each item.

   The getItemsRequest method is part of GroupWise Web Services and allows you to retrieve a list of items, no matter where the item is located in a mailbox. For an example, see the getItemsRequest method.

   You should limit the number of items that you retrieve in a single iteration of getItemsRequest. We recommend that you retrieve no more than 250 items.

Post Office Agent Perspective

The GroupWise Post Office Agent (POA) handles the delivery and other events for each user on the post office. When an event occurs on a user's mailbox, the POA performs the following basic steps:

1. Checks to see if there are event configurations for the user.
2. If there are event configurations, it determines if any of the event configurations match the triggered event.
3. If there is a match, it creates an event record in the GroupWise user's store that describes the event.
4. If notifications are enabled, the POA notifies the application an event has occurred.

   After the notification, the application is removed from the notification list. However, the application can re-insert itself into the notification list by calling getEventsRequest and passing True for the notify element. Prior to GroupWise 8.0.2, the user was dropped from the notification list whether or not the notification is sent. Now, the user is not removed from the notification list, but is inactive. That is, the user will not be notified if a new event happens, until the application calls getEventsRequest and passes True for the notify element.
5. The POA does not wait for an ACK from the application because the notification was patterned after UDP.

To help reduce the load on the POA, we recommend that applications throttle the number of getEventsRequest methods that occur per user. A good idea is to have only one getEventsRequest method in progress per user at any single time.

Other POA Duties

The POA has other management responsibilities for GroupWise Web Services Events. The POA also accepts SOAP requests from applications to handle the following methods:

Method	Description
configureEventsRequest	Creates a new event configuration record for a specific user. It defines what type of events are stored in an event record in the user's store. It also defines the notification address.
getEventConfigurationRequest	Returns the selected configuration or all configurations for a specific user.
getEventsRequest	Returns the list of events that have occurred for a specific user.
getItemsRequest	Returns the specified items. This method is usually called after the application has been notified. The application then calls getEventsRequest and gets a list of items that have changed. The getItemsRequest method is used to retrieve the changed items.
removeEventConfigurationRequest	Removes an event configuration for a specific user.
removeEventsRequest	Removes event records stored in a user's store.
cleanEventConfigurationRequest	Deletes event records in a user's database, based on how long the item has been in the store. It is used for nightly maintenance to reduce the size of user's stores.

Event Records

GroupWise stores event records in the user's database. Each record stores relevant data about the occurring event. For example, suppose an application creates an event configuration to track deleted items in a user's mailbox and a user deletes an item. The POA deletes the item and creates an event record that has the following fields:

```
<Event>
  <event type="EventType"/>
  <id type="uid">
  <sid type="unsignedInt">
  <timeStamp type="xs:dateTime" minOccurs="0"/>
  <field type="string"/>
  <container type="uid"/>
  <from type="uid"/>
  <key type="string"/>
  <uid" type="unsignedInt"/>
  <type type="ItemType"/>
</Event>
```

event

Describes the event.

id

Uniquely identifies the item.

sid

Short identifier of the item.

--For GroupWise 8.0 HP1 and later.

timestamp

Identifies the time the event occurred.

field

List of fields interested in being notified when they change on the item.

container

Identifies the folder or address book for which the event record was created.

from

Identifies the source folder or address book (used with FolderItemMove). For proxies, this field is the UserID of the user logging in as a proxy.

key

Identifies the application for which the event record was created.

uid

Uniquely identifies the event record.

type

Item type of the item affected.

An event record is not created unless you specifically add the event type in the configureEventsRequest method. For example, if you want to track when new folders are created, you must add the FolderAdd event type when you configure events by calling configureEventsRequest. If you don't add the FolderAdd event type, a configuration record is not created and your application is not notified when a folder is added.

Event records can require large amounts of disk space and significantly impede the performance of processing events on the POA. Therefore, event records need to be periodically removed from user's databases. By default, event records persist in a user's database for seven days, but this persistence value has a range of 0-20 days. You can modify this value by calling the configureEventsRequest method.

Old event records also need to be removed. Event records can be removed in two ways:

- You can clean up your own event records. We recommend that you clean up your event records as soon as possible. Events can be removed by calling getEventsRequest, removeEventsRequest, or removeItemsRequest.

  An application can remove a list of event UIDs by calling removeItemsRequest and passing events in the container element and the UIDs in the itemRefList element. This is a good way for an application to remove only the events it has processed.
- The GroupWise POA removes old event records during its nightly maintenance and also disables unused event configurations. The logic for removal and event configuration disabling during nightly maintenance is as follows:
  - Event records older than the persistence date are removed.
  - If an event record is older than the persistence date, the event configuration is disabled.
  - If the event configuration has not been enabled for 21 days, it is deleted.

There can be more than one event record for each event. For example, suppose application A and application B both register to receive event notifications for deleted items for user1. Two separate event records are created in user1's database, one event record for each registered application. Each application would need to remove its own event records.

Event Types and Fields

When a certain event takes place, an event record is created to track the change in the user's database. Event types fall into six categories: address book, address book items, folder, items, login, and proxy access.

All six event categories create events record when a user's database changes. All address book and address book item events apply only to personal address books. System address book changes can be tracked outside of GroupWise events.

Most of the event types are self-descriptive. For example, AddressBookAdd is an event type. An event record with an event type of AddressBookAdd is created when a personal address book is created. There are also event types that are created when an existing item is changed. For example, AddressBookModify is an event type that is created when specific GroupWise fields (such as an email address) are changed in the user's database.

When a user sends an item to a list of users, it is a distributed item to recipients. Recipients of a distributed item can modify only a small subset of that item's fields. For example, the personal subject and categories are the only fields that can be changed by a recipient on a received message. An event type of ItemModify is triggered on a distributed item when the personal subject or category is modified.

GroupWise personal or posted items have a larger set of fields that can be modified by the recipient. On personal or posted items, the only recipient is the sender of the item. The message body, subject, and other fields can create an ItemModify event record.

The following table lists the event categories, event names, and when a specific event occurs.

Event Category	Event Name	Event Occurs When...
Address Book	AddressBookAdd	Personal Address Book is created
	AddressBookDelete	Personal Address Book is deleted
	AddressBookModify	Personal Address Book is modified
Address Book Item	AddressBookItemAdd	Address book item is created
	AddressBookItemDelete	Address book item is deleted
	AddressBookItemModify	Address book item is modified
	PersonalGroupItemAdd	Item is added to a personal group.
	PersonalGroupItemDelete	Item is deleted from a personal group
Folder	FolderAccept	Shared folder is accepted
	FolderAdd	Folder is added
	FolderDeleted	Folder is deleted
	FolderItemAdd	Item is added to a folder
	FolderItemDelete	Deprecated.
	FolderItemMove	Item is moved from one folder to another
	FolderModify	Folder is modified

Event Category	Event Name	Event Occurs When...
Item	ItemAccept	Calendar item is accepted
	ItemArchive	Item is archived
	ItemComplete	Item is marked complete
	ItemDecline	Calendar item is declined
	ItemDelete	Item is deleted
	ItemForward	Item is forwarded
	ItemMarkPrivate	Item is marked private
	ItemMarkUnprivate	Item is marked as not private
	ItemMarkRead	Item is marked read
	ItemMarkUnread	Item is marked unread
	ItemModify	Item is modified
	ItemPurge	Item is purged
	ItemReply	A reply is made to the item
	ItemUnaccept	Item is not accepted
	ItemUnarchive	Item is unarchived
	ItemUncomplete	Item is marked uncomplete
	ItemUndelete	Item is undeleted
Login	Login	User logs in to GroupWise
	ProxyLogin	Proxy user logs in to GroupWise
	TrustedApplicationLogin	Trusted Application logs in to GroupWise
	Logout	User logs out of GroupWise
	SessionTimedOut	User session times out
Proxy Access	ProxyAccessAdd	Proxy rights granted to a user
	ProxyAccessDelete	Proxy rights deleted for a user
	ProxyAccessModify	Existing proxy rights are modified
Rule	RuleAdd	A rule is created
	RuleDelete	A rule is deleted
	RuleModify	A rule is modified
	RuleExecute	A rule is executed

Event Notifications

Event notifications are used to notify listening applications that an event they are interested in has occurred. The POA keeps a notification list in memory, which contains a list of registered applications. (Applications add themselves to the notification list by calling the getEventsRequest method with the notify element set to True.) Each application in the notification list can be listening for events for many users.

Be aware of the following when using event notifications:

- Because there could be network problems or applications might not be listening when a notification is sent, we cannot guarantee that you'll receive every notification. However, an event record is created in the user's store that describes the event. Your application can periodically read the events and determine if the local store is current.
- By default, the getEventsResponse method returns 256 event items. This default number can be changed by setting the count element to the desired count. However, we recommend using the default setting.

  For example, if you were processing 500 event items in a user's event queue, your application is notified that events are available. It would then call getEventsRequest and receive 256 items. However, the POA does not notify the application that there are more events in the queue. The first notification was for all 500 event items. You would need to call getEventsRequest one more time to retrieve the remaining 244 events.

  Notifications resume if a new event is added to the event queue (in addition to the original 500), and if the application is in the notification list
- The POA maintains a notification list in memory. If the notification list has five users associated with one application, the notification process on the POA opens and closes one connection for all the users in the list.
- Each event has a UID element. The UID is a unique identifier for the corresponding event record in the database. The UIDs are sequentially numbered in the database. For example, a database can have events 1086, 1087, 1088, 1089, and 1090 in ascending order. However, calling getEventsResponse might not return the event items in ascending order.

  UIDs that are created in a one-second duration are returned in descending order. For example, UID 1086 is created between seconds one and two. UIDs 1087, 1088, 1089 are created between seconds two and three. UID 1090 is created between seconds three and four. The getEventResponse method returns the event UIDs in the following order: 1086, 1089, 1088, 1087, and 1090, with the items created between seconds two and three in descending order.
- There might be a small time period where all events are not returned by calling getEventsRequest. The is a timing issue based on retrieving events using the from and until dateTime elements.

  For example, there are five separate events written to a user's store in one second and only three of the five events are in the database when the application retrieves the events. The other two events are written to the database after the retrieval. The application might assume that it retrieved all the event s in the database for that one second, but that might not be the case.

We recommend using the following algorithm when retrieving events to ensure that events are not missed. (There might be other methods to achieve the same results.)

1. An application calls getEventsRequest and finds the newest event record returned by selecting the newest timeStamp in the returned list.
2. When the application calls getEventsRequest again, it uses the newest timeStamp and subtracts one second from it. Any duplicates can be removed by the application by tracking the UIDs of items that it has already processed.

- If a POA goes down, the notification list that is stored in memory is lost. Before GroupWise 8.0 SP2, the application needed to call getEventsRequest with the notify element set to True to add itself to the notification list again. Now, when the POA comes back up, it will automatically put the event configuration in the notification list and send a notification on the first event that triggers from the event configuration.
- GroupWise Event notifications do not work correctly when more than one POA runs against a Post Office because the notification list stored in a POA's memory is created and updated when an application calls the getEventsRequest with the notify element set to true.

  For example, we have POA1 and POA2 running against PO1, but our application knows only about POA1. The application calls getEventsRequest with the notify element set to True for POA1. POA1 is now building a notification list to tell the application when a specific event occurs. If a GroupWise client now connects to POA2, POA2 processes the action and creates the event in the user's database. However, POA2 does not have the notification list in memory to notify the application of events. Thus, notifications do not work as expected.

  If your application is interested only in retrieving event records outside of notifications, having two POAs running against one post office is acceptable.

2 Methods

This section describes the GroupWise Web Services Events methods, which include the following:

cleanEventConfigurationRequest

Deletes event records in a user's database based on how long the item has been in the database.

Definitions

all

Specifies whether all events and configurations are removed from a user's account.

status

Specifies whether the request was successful.

code

Provides the error number related to the event. 0 means that the request was successful.

Remarks

The POA calls cleanEventConfigurationRequest during nightly maintenance to reduce the size of user databases.

For a better understanding of nightly maintenance, see Event Records.

Example

```
<cleanEventConfigurationRequest>
   <all>1</all>
</cleanEventConfigurationRequest>

<cleanEventConfigurationResponse>
   <status>
      <code>0</code>
   </status>
</cleanEventConfigurationResponse>
```

configureEventsRequest

Defines the configuration name, what events should be tracked, event persistence, notifications, etc.

Request

```
<configureEventsRequest>
  <events type="Events"/>
</configureEventsRequest>

<Events enabled="">
  <key/>
  <persistence/>
  <ipAddress/>
  <port/>
  <http/>
  <ignoreAfter/>
  <definition type="EventDefinition"/>
</Events>

<EventDefinition>
  <events>
  <type>
  <field>
  <containers>
  <subType>
</EventDefinition>
```

Response

```
<configureEventsResponse>
   <status>
      <code>0</code>
   </status>
</configureEventsResponse>
```

Definitions

enabled

Boolean. Enables or disables event processing. True (1) creates event records. False (0) disables event processing for that event configuration. Other event configurations are not affected.

key

String. Uniquely identifies the event configuration in a user's database. It is up to the application to control the uniqueness of the application key. GroupWise Web Services Events uses any key that it is passed. If two applications or two instances of an application pass the same key, GroupWise Web Services Events maps both to one event configuration structure in the user’s database.

persistence

A duration, from 0-20 days, that specifies how long old event records remain in a user's database. If not defined, the default is seven days. For more information on how the persistence value is used to remove event records, see Event Records.

ipAddress

Identifies the IP address, DNS name, or the HTTP URL that should be used for event notification.

port

Identifies the IP port on which the application is listening for event notifications.

http

Specifies whether event notification should occur through TCP/IP or HTTP. If False (0), the value in ipAddress is treated as an IP address, and a TCP/IP stream is used for the notification. If True (1), the value in the ipAddress element is treated as an HTTP URL, such as http://www.acme.com/events.

ignoreAfter

Some applications are only concerned for events on items in a certain range of days. You can specify a number of days. If an event happens on an item that was created before the day limit, no event will be recorded.

--For GroupWise 8.0 SP2 and later.

events

Identifies the specific events that an application wants to track. For example, if an application wants to create event records when items are added and deleted from a folder, the event element contains FolderItemAdd ItemDelete. The list of event types is space-delimited. For a complete list of event types, see EventType in the events.xsd schema.

type

Identifies the specific item types that an application wants to track. For example, if an application wants to create event records only when a specific action occurs to appointments and tasks, the type element contains Appointment Task. The list of item types is space-delimited. For a complete list of item types, see itemType in the events.xsd schema.

field

Identifies the specific item fields that an application wants to track. The field element applies only to items that are modified. For example, if an application wants to create event records only when the PersonalSubject and Category fields are modified, the field element contains PersonalSubject Category. The list of fields is space-delimited. For a complete list of fields, see FieldList in the events.xsd schema.

containers

Identifies specific containers where an application wants to track folder events. If no container is specified, events are reported for all containers in the user's account, except folders shared with the user.

subType

Identifies a custom item type. Applications can create custom item types by adding the subType element when creating an item. The subType element is defined in the GroupWise types.xsd schema.

During the creation of the item, an application could provide a unique string in the subType element. For example, the application could provide Acme as the subType element and any items that are created have an Acme subType. Applications can search and filter items based on the subType element.

This is GroupWise Web Services implemention of className in the GroupWise Object API.

The GroupWise Web Services Events subType element can be used to track events only on the items that have a subType field that matches the subType. For example, if an application creates an item with a subType of Acme and wants to track all items with an Acme subType, the application provides Acme in the subType element while calling configureEventsRequest.

status

Specifies whether getEventsRequest was successful.

code

Provides the error number related to the event. 0 means that the request was successful.

Remarks

Applications call the configureEventsRequest method to create an event configuration for a GroupWise user. Event records are not created for a user unless this method is called.

Each configureEventsRequest takes a key as a parameter. The key is the name of the event configuration record. You can update an existing event configuration record by using the same key as an existing configuration record. Event definitions can be added at any time.

Event records will not be created unless the <enabled> element is set to True.

Users can have more than one event configuration. If an event applies to more than one configuration record, two separate event records are created, one for each event configuration.

In some cases, an event will not specify what container an item is in, such as an ItemModify event. It can be quite time consuming to retrieve the item because we need to validate that the application has rights to read the item. To speed up the validation, we put in a special format for an id when getting the item using getItemsRequest. The format is id@Event:uid, where id and uid are from the event. For example, here is an event:

```
<event>ItemModify</event>
  <id>4D51518E.domain.PO1.100.1776172.1.25DEE.1</id>
  <sid>155118</sid>
  <timeStamp>2011-02-08T21:26:02Z</timeStamp>
  <field>MessageBody</field>
  <key>test</key>
  <uid>585352</uid>
  <type>Mail</type>
</event>
```

The call to getItemsRequest would then be:

```
<getItemsRequest>
  <container>folders</container>
  <items>
    <item>4D51518E.domain.PO1.100.1776172.1.25DEE.1@Event:585352</item>
  </items>
</getItemsRequest>
```

We use the event record to validate that the user has access to the item. Thus, for this to work, the event record cannot be deleted before you try to read the item. To save one database read, the id can be formatted with the <sid> value instead of the <id> value. So the id for the above item could have been:

```
<item>155118@Event:585352</item>
```

Example

```
<configureEventsRequest>
   <events enabled="1">
      <key>AcmeEvents</key>
      <persistence>7</persistence>
      <ipAddress>app1.widgets.com</ipAddress>
      <port>5221</port>
      <event>
         <event>FolderItemAdd</event>
         <event>FolderItemMove</event>
         <event>FolderItemRemove</event>
         <event>ItemAccept</event>
         <event>ItemComplete</event>
         <event>ItemDecline</event>
         <event>ItemDelete</event>
         <event>ItemMarkRead</event>
         <event>ItemMarkUnread</event>
         <event>ItemPurge</event>
         <event>ItemUndelete</event>
            <type>Appointment Mail Note Task</type>
        </event>
      <containers>
         <container>7.AutoDomain.AutoPO1.100.0.1.0.1@16</container>
         <container>A.AutoDomain.AutoPO1.100.0.1.0.1@19</container>
      </containers>
   </events>
</configureEventsRequest>
```

getEventConfigurationRequest

Returns one or all event configuration definitions for a GroupWise user.

Request

```
<getEventConfigurationRequest>
   <key/>
</getEventConfigurationRequest>
```

Response

```
<getEventConfigurationRequest>
   <events enabled="">
      <key/>
      <persistence/>
      <ipAddress/>
      <port/>
      <http/>
      <event>
      <event>
       . . .
      <type/>
      <field/>
      <containers/>
      <subType/>

   </events>
   <status>
      <code/>
   </status>
</getEventConfigurationRequest>
```

Definitions

enabled

Boolean. Enables or disables event processing. True (1) creates event records. False (0), disables event processing for that event configuration. Other event configurations are not affected.

key

String. Uniquely identifies the event configuration in a user's database. It is up to the application to control the uniqueness of the application key. GroupWise Web Services Events uses any key that it is passed. If two applications or two instances of an application pass the same key, GroupWise Web Services Events maps both to one event configuration structure in the user's database. If a key is not provided, all event definitions for the user are returned.

persistence

A duration, from 0-20 days, that specifies how long old event records remain in a user's database. If not defined, the default is seven days. For more information on how the persistence value is used to remove event records, see Event Records.

ipAddress

Identifies the IP address, DNS name, or the HTTP URL that should be used for event notification.

port

Identifies the IP port on which the application is listening for event notifications.

http

Specifies whether event notification should occur through TCP/IP or HTTP. False (0) indicates that the value in ipAddress is treated as an IP address and a TCP/IP stream is used for the notification. True (1) indicates that the value in ipAddress is treated as an HTTP URL, such as http://www.acme.com/events.

event

Identifies the specific events that an application wants to track. For example, if an application wants to create event records when items are added and deleted from a folder, the event element contains FolderItemAdd ItemDelete. The list of event types is space-delimited. For a complete list of event types, see EventType in the events.xsd schema.

type

Identifies the specific item types that an application wants to track. For example, if an application wants to create event records only when a specific action occurs to appointments and tasks, the type element contains Appointment Task. The list of item types is space delimited. For a complete list of item types, see itemType in the events.xsd schema.

field

Identifies the specific item fields that an application wants to track. The field element applies only to items that are modified. For example, if an application wants to create event records only when the PersonalSubject and Category fields are modified, the field element contains PersonalSubject Category. The list of fields is space-delimited. For a complete list of fields, see FieldList in the events.xsd schema.

containers

Identifies specific containers where an application wants to track folder events. If no container is specified, events are reported for all containers in the user's account, except folders shared with the user.

subType

Identifies a custom item type. Applications can create custom item types by adding the subType element when creating an item. The subType element is defined in the GroupWise types.xsd schema.

During the creation of the item, an application could provide a unique string in the subType element. For example, the application could provide Acme as the subType element and any items that are created have an Acme subType. Applications can search and filter items based on the subType element. This is GroupWise Web Services implementation of className in the GroupWise Object API.

The GroupWise Web Services Events subType element can be used to track events only on the items that have a subType field that matches the subType. For example, if an application creates an item with a subType of Acme and wants to track all items with an Acme subType, the application provides Acme in the subType element while calling configureEventsRequest.

status

Specifies whether the getEventConfigurationRequest method was successful.

code

Provides the error number related to the event. 0 means that the request was successful.

Remarks

If the key element is provided in the call to getEventConfigurationRequest, only that specific event definition is returned. If the key element is not provided, all event definitions for the user are returned.

Example

```
<getEventConfigurationResponse>
   <events>
      <event enabled="1">
         <key>Acme</key>
         <persistence>8</persistence>
         <ipAddress>app1.widgets.com</ipAddress>
         <port>5221</port>
         <http>0</http>
         <event>
            <event>ItemAccept</event>
            <event>ItemComplete</event>
            <event>ItemDecline</event>
            <event>ItemDelete</event>
            <event>ItemPurge</event>
            <event>ItemMarkRead</event>
            <event>ItemUndelete</event>
            <event>ItemMarkUnread</event>
            <event>FolderItemAdd</event>
            <event>FolderItemMove</event>
            <event>FolderItemRemove</event>
         </event>
         <type>Appointment Mail Note Task</type>
      </event>
      <event enabled="1">
         <key>AB</key>
         <persistence>0</persistence>
         <ipAddress>http://prestons/</ipAddress>
         <port>5221</port>
         <http>1</http>
         <event>
            <event>AddressBookDelete</event>
```

```
            <event>AddressBookAdd</event>
            <event>AddressBookItemDelete</event>
            <event>AddressBookItemAdd</event>
         </event>
      </event>
      <containers>
         <container>7.AutoDomain.AutoPO1.100.0.1.0.1@16</container>
         <container>A.AutoDomain.AutoPO1.100.0.1.0.1@19</container>
      </containers>
   </events>
   <status>
      <code>0</code>
   </status>
</getEventConfigurationResponse>
```

getEventsRequest

Returns the list of events that are accumulating in the user's account.

Request

```
<getEventsRequest>
   <key/>
   <from/>
   <until/>
   <uid/>
   <count/>
   <remove/>
   <notify/>
   <view/>
</getEventsRequest>
```

Response

```
<getEventsResponse>
   <events>
      <event>
      <event>
         <id/>
         <timeStamp/>
         <container/>
         <key/>
      </event>
      . . .
   </events>
   <status>
      <code/>
   </status>
</getEventsResponse>
```

Definitions

key

String. Uniquely identifies the event configuration in a user's database. It is up to the application to control the uniqueness of the application key. GroupWise Web Services Events uses any key that it is passed. If two applications or two instances of an application pass the same key, GroupWise Web Services Events maps both to one event configuration structure in the user’s database.

from

Provides the starting date for the list of events to be returned.

until

Provides the ending date for the list of events to be returned.

count

Specifies how many events to return. If not specified, all events are returned. We recommend providing a count of 250 or less.

remove

Specifies whether to remove the event. True (1) indicates to remove the event from the database upon a successful response to getEventsRequest.

notify

Specifies whether the application wants to be notified the next time the event it is tracking occurs. The notification process (POA) maintains a list of all applications that want to be notified when that event occurs. Every time an application receives an event notification, it is removed from the notification list. To be placed on the notification list again, an application must send getEventsRequest with the notify element set to True (1). The IP address and port must be in the event definition to be added to the notification list.

event

Identifies a GroupWise event that has occurred for a user, as defined in EventType in events.xsd.

id

Identifies the item, as defined in types.xsd.

timeStamp

Specifies the date and time that the GroupWise event occurred.

container

Specifies the location in the GroupWise account where the event occurred.

status

Specifies whether getEventsRequest was successful.

code

Provides the error number related to the event. 0 means that the request was successful.

view

Specifies the elements that are returned for each item. The view reduces the amount of data returned. If a view is not specified, all elements are returned.

Remarks

After a successful call to configureEventsRequest, events are created in user's database.

The from and until date elements can be used to return a subset of events, based on dates. The from element can be used by itself to return all events greater than the specified time. Another way to return a subset of events is to provide the UID for a specific event and provide a count. A list of events is returned, starting at the UID for the count specified.

Applications should emove events as soon as possible. One way to remove events is to set the remove element to True in getEventsRequest. After the events are returned, the events are purged from the user's store.

Example

```
<getEventsRequest>
   <key>Acme</key>
   <remove>1</remove>
   <notify>1</notify>
</getEventsRequest>
```

Following is a sample response to getEventsRequest:

```
<getEventsResponse>
   <events>
      <event>
         <event>FolderItemAdd</event>
         <id>41937EE0.AutoDomain.AutoPO1.100.1363230.1.272D.1</id>
         <timeStamp>2012-11-11T22:01:55Z</timeStamp>
         <container>7.AutoDomain.AutoPO1.100.0.1.0.1@16</container>
         <key>Acme</key>
      </event>
     <event>
        <event>FolderItemAdd</event>
        <id>41937F2C.AutoDomain.AutoPO1.100.1363230.1.272E.1</id>
        <timeStamp>2012-11-11T22:03:10Z</timeStamp>
        <container>7.AutoDomain.AutoPO1.100.0.1.0.1@16</container>
        <key>Acme</key>
     </event>
   </events>
   <status>
      <code>0</code>
   </status>
</getEventsResponse>
```

getItemsRequest

Retrieves items returned in the response to getEventsRequest.

Request

```
<getItemsRequest>
   <container/>
   <filter>
      <element>
         <op/>
         <field/>
         <value/>
      </element>
   </filter>
</getItemsRequest>
```

Definitions

container

Specifies the container the item is in. To search all containers except folders shared with me, specify “folders.”

count

Specifies how many items to retrieve in one response. If not specified, all items are returned.

element

Identifies the item to retrieve.

Remarks

The getEventResponse method returns the ID of items that match the configureEventsRequest definition, but it does not return the items themselves. It is up to the application to retrieve the items.

For more information on getItemsRequest and filtering, see the appropriate definition in the GroupWise Web Services document.

Example

The following example shows how to retrieve event items over all folders. The container element tells the POA to search all folders (except folders shared with me) for the item.

```
<getItemsRequest>
   <container>folders</container>
   <view>default peek id container @type message recipients attachments
        subject</view>
   <filter/>
   <count>-1</count>
</getItemsRequest>
```

removeEventConfigurationRequest

Removes the event configuration definition for the specified key.

Request

```
<removeEventConfigurationRequest>
   <key/>
</removeEventConfigurationRequest>
```

Definitions

key

String. Uniquely identifies the event configuration in a user's database. It is up to the application to control the uniqueness of the application key. GroupWise Web Services Events uses any key that it is passed. If two applications or two instances of an application pass the same key, GroupWise Web Services Events maps both to one event configuration structure in the user's database.

status

Specifies whether getEventsRequest was successful.

code

Provides the error number related to the event. 0 means that the request was successful.

Example

```
<removeEventConfigurationRequest>
   <key>GW1</key>
</removeEventConfigurationRequest>

<removeEventConfigurationResponse>
   <status>
      <code>0</code>
   </status>
</removeEventConfigurationResponse>
```

removeEventsRequest

Removes the events in the user's database for the specified key.

Request

```
<removeEventsRequest>
   <key/>
   <from/>
   <until/>
</removeEventsRequest>
```

Definitions

key

String. Uniquely identifies the event configuration in a user's database. It is up to the application to control the uniqueness of the application key. GroupWise Web Services Events uses any key that it is passed. If two applications or two instances of an application pass the same key, GroupWise Web Services Events maps both to one event configuration structure in the user's database.

from

Provides the starting date for the events to be removed.

until

Provides the ending date for the events to be removed.

status

Specifies whether getEventsRequest was successful.

code

Provides the error number related to the event. 0 means that the request was successful.

Example

```
<removeEventsRequest>
   <key>GWEvents</key>
</removeEventsRequest>

<removeEventsResponse>
   <status>
      <code>0</code>
   </status>
</removeEventsResponse>
```

3 Event Examples

This section contains examples of events that are returned by various items in the GroupWise system.

The examples in the following sections use the GroupWise Windows client (with default settings) to determine what type of events are created. You might want to anticipate a different set of events based on the customization of GroupWise clients by end users. The examples also listened for all events except Login, Logout, and TrustedApplicationLogin.

Folders

Folder management consists of creating, deleting, and modifying all folder types, including personal, shared, IMAP, NNTP, and query folders. The following sections contain examples of events that are returned for IMAP, NNTP, and Personal folders:

Creating a Folder

Creating a folder creates a FolderAdd event.

```
<gwe:event>
   <gwe:event>FolderAdd</gwe:event>
   <gwe:id>44E06767.domain1.po1.100.16E3837.1.EED.1</gwe:id>
   <gwe:timeStamp>2012-08-14T18:07:03Z</gwe:timeStamp>
   <gwe:container>C.domain1.po1.100.0.1.0.1@21</gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10286</gwe:uid>
</gwe:event>
```

Deleting a Folder

Deleting a folder creates a FolderDelete event.

```
<gwe:event>
   <gwe:event>FolderDelete</gwe:event>
   <gwe:id>44E06767.domain1.po1.100.16E3837.1.EED.1</gwe:id>
   <gwe:timeStamp>2012-08-14T18:12:19Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10288</gwe:uid>
</gwe:event>
```

Modifying a Folder

Modifying a folder name creates a FolderModify event.

```
<gwe:event>
   <gwe:event>FolderModify</gwe:event>
   <gwe:id>44E06767.domain1.po1.100.16E3837.1.EED.1</gwe:id>
   <gwe:timeStamp>2012-08-14T18:10:48Z</gwe:timeStamp>
   <gwe:field>Name</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10287</gwe:uid>
</gwe:event>
```

Moving a Folder

When a folder is moved, a FolderModified event is triggered. On the event, the id is the folder that is moved. The field has the keyword “Parent”.

```
<gwe:events>
<gwe:event>
   <gwe:event>FolderModifiy</gwe:event>
   <gwe:id>48F5F3AC.domain1.po1.100.16A6163.1.138.1</gwe:id>
   <gwe:timeStamp>2012-10-15T19:44:40Z</gwe:timeStamp>
   <gwe:field>Parent</gwe:field>
   <gwe:key>Events 1</gwe:key>
   <gwe:uid>30</gwe:uid>
</gwe:event>
</gwe:events>
```

Shared Folders

Folders can be shared with other users. The following definitions (as defined in types.xsd for shared folders) are used in the examples in this section:

- When a user shares a folder with other users, it is referred to as isSharedByMe.
- When a user accepts a folder shared by another user, it is referred to as isSharedToMe.

The following sections contain examples of events that are returned for shared folders:

- “Creating Shared Folders” on page 35
- “Deleting Shared Folders” on page 35

Creating Shared Folders

When a user shares a folder with other users, you can expect the following events:

- **FolderAdd:** Created for the new folder.
- **FolderItemAdd:** Created for the sent item shared folder notification.

```
<gwe:event>
   <gwe:event>FolderAdd</gwe:event>
   <gwe:id>44E05582.domain1.po1.100.16E3837.1.EE2.1</gwe:id>
   <gwe:timeStamp>2012-08-14T16:50:42Z</gwe:timeStamp>
   <gwe:containerC.domain1.po1.100.0.1.0.1@21/gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10261</gwe:uid>
</gwe:event>
```

```
<gwe:event>
   <gwe:event>FolderItemAdd</gwe:event>
   <gwe:id>44E05584.domain1.po1.100.16E3837.1.EE3.1</gwe:id>
   <gwe:timeStamp>2012-08-14T16:50:44Z</gwe:timeStamp>
   <gwe:container>7.domain1.po1.100.0.1.0.1@16</gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10262</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Deleting Shared Folders

When a folder that was shared to others (isSharedByMe) is deleted, you can expect the following event:

```
<gwe:event>
   <gwe:event>FolderDelete</gwe:event>
   <gwe:id>44E05582.domain1.po1.100.16E3837.1.EE2.1</gwe:id>
   <gwe:timeStamp>2012-08-14T17:22:18Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10263</gwe:uid>
</gwe:event>
```

Modifying Shared Folders

When a folder is renamed, you can expect a FolderModify event.

```
<gwe:event>
   <gwe:event>FolderModify</gwe:event>
   <gwe:id>44E05D7B.domain1.po1.100.16E3837.1.EE4.1</gwe:id>
   <gwe:timeStamp>2012-08-14T17:24:59Z</gwe:timeStamp>
   <gwe:field>Name</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10266</gwe:uid>
</gwe:event>
```

When users are added or removed from folder sharing, a folderItemAdd event is returned for the notifications.

Sharing Folders With Others (isSharedToMe)

When a folder is shared with other users (isSharedByMe), a shared folder notification is sent to all the users that the folder was shared with (isSharedToMe). This notification appears in the mailbox folder. If your application is listening for the FolderItemAdd event types on the mailbox folder, you can expect the following event:

```
<gwe:event>
   <gwe:event>FolderItemAdd</gwe:event>
   <gwe:id>44E040CA.domain1.po1.100.16E3837.1.EE0.1</gwe:id>
   <gwe:timeStamp>2012-08-14T15:22:18Z</gwe:timeStamp>
   <gwe:container>7.domain1.po1.100.0.1.0.1@16</gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10253</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Accepting a Shared Folder (isSharedToMe)

When a user accepts a shared folder notification, you can expect the following events:

- **FolderAccept:** Created when the user accepts the shared folder notification.
- **FolderAdd:** Created when the folder is added to the folder tree.
- **FolderItemMove:** Created when the original shared folder notification is marked hidden and moved to the newly created folder.

```
<gwe:event>
   <gwe:event>FolderAccept</gwe:event>
   <gwe:id>44E040CA.domain1.po1.100.16E3837.1.EE0.1</gwe:id>
   <gwe:timeStamp>2012-08-14T15:41:40Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10255</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

```
<gwe:event>
```

```
   <gwe:event>FolderAdd</gwe:event>
   <gwe:id>44E04554.domain1.po1.100.16E3837.1.EE1.1</gwe:id>
   <gwe:timeStamp>2012-08-14T15:41:40Z</gwe:timeStamp>
   <gwe:containerC.domain1.po1.100.0.1.0.1@21/gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10256</gwe:uid>
</gwe:event>

<gwe:event>
   <gwe:event>FolderItemMove</gwe:event>
   <gwe:id>44E040CA.domain1.po1.100.16E3837.1.EE0.1</gwe:id>
   <gwe:timeStamp>2012-08-14T15:41:41Z</gwe:timeStamp>
   <gwe:field>Hidden</gwe:field>
   <gwe:container>44E04554.domain1.po1.100.16E3837.1.EE1.1@13<
   gwe:container>
   <gwe:from>7.domain1.po1.100.0.1.0.1@16</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10257</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Deleting Shared Folders

When a user deletes an isSharedToMe folder, you can expect the following events:

- **ItemPurge:** The original shared folder notification that was moved to the shared folder and marked hidden.
- **FolderDelete:** The folder being deleted.

```
<gwe:event>
   <gwe:event>ItemPurge</gwe:event>
   <gwe:id>44E040CA.domain1.po1.100.16E3837.1.EE0.1</gwe:id>
   <gwe:timeStamp>2012-08-14T16:31:06Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10259</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>

<gwe:event>
   <gwe:event>FolderDelete</gwe:event>
   <gwe:id>44E04554.domain1.po1.100.16E3837.1.EE1.1</gwe:id>
   <gwe:timeStamp>2012-08-14T16:31:06Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10260</gwe:uid>
</gwe:event>
```

Modifying Shared Folders

When a folder is renamed, you can expect a FolderModify event.

```
<gwe:event>
   <gwe:event>FolderModify</gwe:event>
   <gwe:id>44E05D7B.domain1.po1.100.16E3837.1.EE4.1</gwe:id>
   <gwe:timeStamp>2012-08-14T17:24:59Z</gwe:timeStamp>
   <gwe:field>Name</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10266</gwe:uid>
</gwe:event>
```

Items

Users can act on items. For example, a user can accept an item. Accepting an item can create an event record with an event type of ItemAccept. The following sections explain different actions on items and the resulting event records that are created.

FolderItemDelete is obsolete. Do not use it because it will be removed from the schemas in a future release.

PersonalGroupItemAdd and PersonalGroupItemDelete are defined in the GroupWise 7.0.1 schemas. However, they are not used at this time but are reserved for future use.

Accepting an Item

When a user accepts a distributed appointment, note, or task, you can expect the following events:

- **ItemAccept:** Created when the distributed item is accepted.
- **ItemModify:** The item has been modified.

- **FolderItemMove:** Reports the moving of the item from the mailbox folder to the calendar.

The first ItemModify event example shows changes to the status of the item, such as the item being opened and read. The second ItemModify example reports the change to the recipient status on the sent item record for the appointment. This ItemModify appears only if the sender is the recipient.

```
<gwe:event>
   <gwe:event>ItemAccept</gwe:event>
   <gwe:id>44E18F4D.domain1.po1.100.16E3837.1.F05.1</gwe:id>
   <gwe:timeStamp>2012-08-15T15:09:43Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10366</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

```
<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44E18F4D.domain1.po1.100.16E3837.1.F05.1</gwe:id>
   <gwe:timeStamp>2012-08-15T15:09:43Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10368</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

```
<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44E18F4C.domain1.po1.100.16E3837.1.F04.1</gwe:id>
   <gwe:timeStamp>2012-08-15T15:09:43Z</gwe:timeStamp>
   <gwe:field>RecipientStatus</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10367</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

```
<gwe:event>
   <gwe:event>FolderItemMove</gwe:event>
   <gwe:id>44E18F4D.domain1.po1.100.16E3837.1.F05.1</gwe:id>
   <gwe:timeStamp>2012-08-15T15:09:43Z</gwe:timeStamp>
   <gwe:container>A.domain1.po1.100.0.1.0.1@19</gwe:container>
   <gwe:from>7.domain1.po1.100.0.1.0.1@16</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10369</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

Archiving an Item

When a user archives an item, you can expect the following events:

- **ItemArchive:** Created when the item is archived. The item is then deleted and purged from the on-line account.
- **ItemPurge:** Created when an item is purged from the online account.

```
<gwe:event>
   <gwe:event>ItemArchive</gwe:event>
   <gwe:id>44E1A331.domain1.po1.100.16E3837.1.F1B.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:34:47Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10438</gwe:uid>
</gwe:event>

<gwe:event>
   <gwe:event>ItemPurge</gwe:event>
   <gwe:id>44E1A331.domain1.po1.100.16E3837.1.F1B.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:34:47Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10439</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Completing an Item

When a user completes a task, you can expect the following events:

- **ItemComplete:** Created when the item is completed.
- **ItemModify:** Created when the recipient status of the sent item is changed to completed.

```
<gwe:event>
   <gwe:event>ItemComplete</gwe:event>
   <gwe:id>44E1A571.domain1.po1.100.16E3837.1.F1E.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:52:35Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10502</gwe:uid>
   <gwe:type>Task</gwe:type>
</gwe:event>

<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44E1A571.domain1.po1.100.16E3837.1.F1D.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:52:35Z</gwe:timeStamp>
   <gwe:field>RecipientStatus</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10503</gwe:uid>
   <gwe:type>Task</gwe:type>
</gwe:event>
```

Declining an Item

Deleting a calendar item (appointment, note, or task) creates an ItemDecline event.

```
<gwe:event>
   <gwe:event>ItemDecline</gwe:event>
   <gwe:id>44E0A919.domain1.po1.100.16E3837.1.F00.1</gwe:id>
   <gwe:timeStamp>2012-08-15T19:49:25Z</gwe:timeStamp>
   <gwe:from>A.domain1.po1.100.0.1.0.1@19</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10544</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

Deleting an Item

Deleting a mail or phone item creates an ItemDelete event.

```
<gwe:event>
   <gwe:event>ItemDelete</gwe:event>
   <gwe:id>44E0A919.domain1.po1.100.16E3837.1.F00.1</gwe:id>
   <gwe:timeStamp>2012-08-15T19:49:25Z</gwe:timeStamp>
   <gwe:from>A.domain1.po1.100.0.1.0.1@19</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10544</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

Adding an Item To a Folder

When items are created in a folder, a FolderItemAdd event is created.

```
<gwe:event>
   <gwe:event>FolderItemAdd</gwe:event>
   <gwe:id>44E34395.domain1.po1.100.16E3837.1.F3D.1</gwe:id>
   <gwe:timeStamp>2012-08-16T22:11:01Z</gwe:timeStamp>
   <gwe:container>7.domain1.po1.100.0.1.0.1@16</gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10633</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Moving an Item

When an item is moved from one folder to another folder, a FolderItemMove event is created. The from container is the source folder and the container is the destination folder.

```
<gwe:event>
   <gwe:event>FolderItemMove</gwe:event>
   <gwe:id>44E34395.domain1.po1.100.16E3837.1.F3D.1</gwe:id>
   <gwe:timeStamp>2012-08-16T22:13:04Z</gwe:timeStamp>
   <gwe:container>C.domain1.po1.100.0.1.0.1@21</gwe:container>
   <gwe:from>7.domain1.po1.100.0.1.0.1@16</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10634</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Marking an Item Private

Private items cannot be viewed by proxy users. Marking an item private creates an ItemMarkPrivate event.

```
<gwe:event>
   <gwe:event>ItemMarkPrivate</gwe:event>
   <gwe:id>44E1F6FE.domain1.po1.100.16E3837.1.F2A.1</gwe:id>
   <gwe:timeStamp>2012-08-15T22:34:59Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10577</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Marking an Item Read

When an item is opened, the opened and read status flags on an item record are set.

Marking an item read creates an ItemMarkRead event.

```
<gwe:event>
   <gwe:event>ItemMarkRead</gwe:event>
   <gwe:id>44E1F6FE.domain1.po1.100.16E3837.1.F2A.1</gwe:id>
   <gwe:timeStamp>2012-08-15T22:39:01Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10580</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Marking an Item Unprivate

By default, items are not private and can be seen by proxy users. Marking an item as not private creates an ItemMarkUnprivate event.

```
<gwe:event>
   <gwe:event>ItemMarkUnprivate</gwe:event>
   <gwe:id>44E1F6FE.domain1.po1.100.16E3837.1.F2A.1</gwe:id>
   <gwe:timeStamp>2012-08-15T22:35:35Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10578</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Marking an Item Unread

In order for an item to be marked unread, it needs to have been opened first. When an item is opened, the opened and read status flags on an item record are set. When an item is marked unread, the read status flag is reset to not read and the opened flag is not changed.

Marking an item unread creates an ItemMarkRead event.

```
<gwe:event>
   <gwe:event>ItemMarkUnread</gwe:event>
   <gwe:id>44E1F6FE.domain1.po1.100.16E3837.1.F2A.1</gwe:id>
   <gwe:timeStamp>2012-08-15T22:38:41Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10579</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Modifying an Item

ItemModify modifies an existing item.

A distributed item is sent to many users. A personal item is a posted or personal item. The number of fields that can be changed on a item depends on whether it is a distributed or a personal item. On a distributed item, only the category and personal subject can be modified.

Modifying a distributed item creates an ItemModify event. The field element defines the field that was changed on the item. In the following example, the Category and PersonalSubject were both changed.

```
<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44E1FB97.domain1.po1.100.16E3837.1.F2C.1</gwe:id>
   <gwe:timeStamp>2012-08-16T21:17:19Z</gwe:timeStamp>
   <gwe:field>Category PersonalSubject</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10589</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

If an item is personal, most fields can be changed on the item after its sent. In the example below, the Alarm, Category, Classification, and other fields were modified.

```
<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44E339DD.domain1.po1.100.16E3837.1.F30.1</gwe:id>
   <gwe:timeStamp>2012-08-16T21:30:19Z</gwe:timeStamp>
   <gwe:field>Alarm Category Classification Duration Place Security
        SendPriority StartDate Subject Alarm</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10592</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

Purging an Item

Items that are deleted are moved to the Trash folder. Items can then be purged by deleting them from the Trash. Items can also be purged without sending the item to the Trash. In a client, select **Delete and Empty**, and the item is removed from the user's account without storing the item in the Trash folder.

Purging an item creates an ItemPurge event.

```
<gwe:event>
   <gwe:event>ItemDelete</gwe:event>
   <gwe:id>44E1FB97.domain1.po1.100.16E3837.1.F2C.1</gwe:id>
   <gwe:timeStamp>2012-08-16T21:34:16Z</gwe:timeStamp>
   <gwe:from>7.domain1.po1.100.0.1.0.1@16</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10594</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>

<gwe:event>
   <gwe:event>ItemPurge</gwe:event>
   <gwe:id>44E1FB97.domain1.po1.100.16E3837.1.F2C.1</gwe:id>
   <gwe:timeStamp>2012-08-16T21:34:17Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10595</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Declining an Item

When an application uses the unacceptRequest method defined in GroupWise Web Services, the ItemUnaccept event is created.

```
<gwe:event>
   <gwe:event>ItemUnaccept</gwe:event>
   <gwe:id>44EAE897.domain1.po1.100.16E3837.1.F6A.1</gwe:id>
   <gwe:timeStamp>2012-08-23T15:24:26Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10747</gwe:uid>
   <gwe:type>Appointment</gwe:type>
</gwe:event>
```

Unarchiving an Item

When a user unarchives an item, the following events can be expected:

- **ItemUnarchive:** Created when the item is unarchived.
- **FolderItemAdd:** Created when the item is re-created in the online account

```
<gwe:event>
   <gwe:event>ItemUnarchive</gwe:event>
   <gwe:id>44E1A461.domain1.po1.100.16E3837.1.F1C.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:39:29Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10441</gwe:uid>
</gwe:event>
```

```
<gwe:event>
   <gwe:event>FolderItemAdd</gwe:event>
   <gwe:id>44E1A461.domain1.po1.100.16E3837.1.F1C.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:39:29Z</gwe:timeStamp>
   <gwe:container>7.domain1.po1.100.0.1.0.1@16</gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10440</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Marking an Item Not Complete

When a user marks a task as not being complete, a ItemUncomplete event is created.

```
<gwe:event>
   <gwe:event>ItemUncomplete</gwe:event>
   <gwe:id>44E1A571.domain1.po1.100.16E3837.1.F1E.1</gwe:id>
   <gwe:timeStamp>2012-08-15T16:51:50Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10492</gwe:uid>
   <gwe:type>Task</gwe:type>
</gwe:event>
```

Undeleting an Item

Items that are in the trash folder can be undeleted or moved back to the original folder. Undeleting an item creates an ItemUndelete event.

```
<gwe:event>
   <gwe:event>ItemUndelete</gwe:event>
   <gwe:id>44E1F6FE.domain1.po1.100.16E3837.1.F2A.1</gwe:id>
   <gwe:timeStamp>2012-08-16T21:42:14Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10596</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

Personal Address Books

Managing personal address books containers and manipulating items within the personal address book containers is different, as shown in the following sections:

- "Personal Address Book Management" on page 45
- "Personal Address Book Items" on page 46
- "Shared Address Books" on page 49

Personal Address Book Management

Personal Address Book management event types include the following:

- **AddressBookAdd:** Used when a new Personal Address Book is created.

- **AddressBookDelete:** Used when an existing Personal Address Book is deleted.
- **AddressBookModify:** Used when an existing Personal Address Book is modified.

Personal Address Book Items

Additions, modifications, and deletions to Personal Address Book items can produce more event records than expected. The reason for the extra events is because of the way the Personal Address Book code was written.

First, the Personal Address Book code creates the item. Next, it modifies the item. Thus, two event records are created for each Personal Address Book addition. For example, when adding a contact to a Personal Address Book, two different event records are created. The first record is AddressBookItemAdd. The second event is AddressBookItemModify (because the created item is now modified).

The following sections explain the various events associated with Personal Address Book actions:

- "Adding Personal Address Book Items" on page 46
- "Deleting Personal Address Book Items" on page 47
- "Modifying Personal Address Book Items" on page 47

Adding Personal Address Book Items

When adding a contact, group, resource, or organization to a Personal Address Book, you might get more events than just AddressBookItemAdd. For example, if you add a resource that specifies an owner that currently does not exist in the Personal Address Book, you receive the following three events:

- **AddressBookItemAdd:** Created for the owner that is not in the Personal Address Book.
- **AddressBookItemAdd:** Created for the resource itself.
- **AddressBookItemModify:** Created for the resource for the owner.

When a contact, group, resource, or organization is added and it references another contact, group, resource, or organization that does not exist in the current PAB, the referenced address book object is created first. This produces an extra AddressBookItemAdd event.

```
<gwe:getEventsResponse>
   <gwe:events>
      <gwe:event>
         <gwe:event>AddressBookItemAdd</gwe:event>
         <gwe:id>44D0AAA2.domain1.po1.104.16E3837.1.DF.1</gwe:id>
         <gwe:timeStamp>2012-08-02T19:37:38Z</gwe:timeStamp>
         <gwe:container>42C510EA.domain1.po1.104.16E3837.1.3.1@53
                </gwe:container>
         <gwe:key>GWEvents</gwe:key>

         <gwe:uid>10086</gwe:uid>
         <gwe:type>Contact</gwe:type>
      </gwe:event>
```

```
      <gwe:event>
         <gwe:event>AddressBookItemModify</gwe:event>
         <gwe:id>44D0AAA2.domain1.po1.104.16E3837.1.E0.1</gwe:id>
         <gwe:timeStamp>2012-08-02T19:37:39Z</gwe:timeStamp>
         <gwe:field>Owner</gwe:field>
         <gwe:key>GWEvents</gwe:key>
         <gwe:uid>10088</gwe:uid>
         <gwe:type>Resource</gwe:type>
      </gwe:event>

      <gwe:event>
         <gwe:event>AddressBookItemAdd</gwe:event>
         <gwe:id>44D0AAA2.domain1.po1.104.16E3837.1.E0.1</gwe:id>
         <gwe:timeStamp>2012-08-02T19:37:39Z</gwe:timeStamp>
         <gwe:container>42C510EA.domain1.po1.104.16E3837.1.3.1@53
              </gwe:container>
         <gwe:key>GWEvents</gwe:key>
         <gwe:uid>10087</gwe:uid>
         <gwe:type>Resource</gwe:type>
      </gwe:event>
   </gwe:events>
   <gwe:status>
      <gwt:code>0</gwt:code>
   </gwe:status>
</gwe:getEventsResponse>
```

When creating a Personal Address Book group, you also create an AddressBookItemAdd event for each member of the group. For example, if a group is created with two contacts, the following four events are created:

- **AddressBookItemAdd:** Created for the group.
- **AddressBookItemModify:** Created for the group modification.
- **AddressBookItemAdd:** Created for the first contact in the group.
- **AddressBookItemAdd:** Created for the second contact in the group.

Deleting Personal Address Book Items

When you delete a Personal Address Book item, you can expect to receive the following events:

- **AddressBookItemDelete:** Created when deleting a contact, organization, or resource.
- **AddressBookItemDelete:** Created when deleting a Personal Address Book group itself and when deleting each member of the group.

Modifying Personal Address Book Items

When a Personal Address Book item is modified, an event record is created. By default, the event record does not include a field element because the event record is generic. It tells the listening application that a field on an address book item has changed, but it does not tell the application listener what specific fields have changed.

The following is an example of a generic event record that does not include a field element:

```
<gwe:event>
   <gwe:event>AddressBookItemModify</gwe:event>
   <gwe:id>438B0F64.domain1.po1.104.16E3837.1.90.1</gwe:id>
   <gwe:timeStamp>2012-11-28T21:25:39Z</gwe:timeStamp>
   <gwe:container>434E22A5.domain1.po1.104.16E3837.1.5D.1@53
         </gwe:container>
   <gwe:key>GWABEvents</gwe:key>
   <gwe:uid>2284</gwe:uid>
</gwe:event>
```

Generic events are returned when the following Personal Address Book fields change:

- Title
- Department
- Web site
- Birthday
- Comment
- Type

However, if the following fields are changed, a field element is created that contains the type of field that was changed:

- Category
- Contact
- E-mail address
- IM address
- Name
- Phone number
- Postal address

For example, if the name on a contact is changed, the following event record is created:

```
<gwe:event>
   <gwe:event>AddressBookItemModify</gwe:event>
   <gwe:id>438B0F64.domain1.po1.104.16E3837.1.90.1</gwe:id>
   <gwe:timeStamp>2012-11-28T21:39:05Z</gwe:timeStamp>
   <gwe:field>Name</gwe:field>
   <gwe:container>434E22A5.domain1.po1.104.16E3837.1.5D.1@53
       </gwe:container>
   <gwe:key>GWABEvents</gwe:key>
   <gwe:uid>2285</gwe:uid>
</gwe:event>
```

In this example, the field element contains the Name key word, which tells the listening application that some part of the name in a contact has changed. The listening application now has more information about the particular Personal Address Book change and can act accordingly.

Shared Address Books

Shared address books within a Personal Address Book container are similar to shared folders. When user1 shares an address book with user2, user2 receives a mail item in the In box. User2 can then accept the shared address book.

Applications receive the following events for shared address books:

- **FolderItemAdd:** Created for the new mail item in user2's In box.

Your application can call GetItemsRequest to retrieve the details of the shared address book item. The GetItemsResponse has a SharedNotification attribute, which states that the item is a SharedAddressBook.

```
<gwe:event>
   <gwe:event>FolderItemAdd</gwe:event>
   <gwe:id>44EC2654.domain1.po1.100.16E3837.1.F76.1</gwe:id>
   <gwe:timeStamp>2012-08-23T15:56:36Z</gwe:timeStamp>
   <gwe:container>7.domain1.po1.100.0.1.0.1@16</gwe:container>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10755</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

```
<gwt:item xsi:type="gwt:SharedNotification">
   ...
   <gwt:subject>Sharing of Address Book 'New U2 Book'.</gwt:subject>
   ...
   <gwt:notification>SharedAddressBook</gwt:notification>
   <gwt:description>You have been granted shared access to my address
      book named 'New U2 Book'.</gwt:description>
   <gwt:rights>
      <gwt:edit>1</gwt:edit>
   </gwt:rights>
</gwt:item>
```

When a user accepts the address book, the ItemAccept, ItemModify, and FolderItemMove events are created. These events mark the accepted item as hidden and as moved to the calendar folder, so that your application does not need to track the notification item any more.

```
<gwe:event>
   <gwe:event>ItemAccept</gwe:event>
   <gwe:id>44EC2654.domain1.po1.100.16E3837.1.F76.1</gwe:id>
   <gwe:timeStamp>2012-08-23T15:58:08Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10758</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

```
<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44EC2654.domain1.po1.100.16E3837.1.F76.1</gwe:id>
   <gwe:timeStamp>2012-08-23T15:58:09Z</gwe:timeStamp>
   <gwe:field>Hidden</gwe:field>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10759</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>

<gwe:event>
   <gwe:event>ItemModify</gwe:event>
   <gwe:id>44EC2654.domain1.po1.100.16E3837.1.F76.1</gwe:id>
   <gwe:timeStamp>2012-08-23T15:58:09Z</gwe:timeStamp>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10760</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>

<gwe:event>
   <gwe:event>FolderItemMove</gwe:event>
   <gwe:id>44EC2654.domain1.po1.100.16E3837.1.F76.1</gwe:id>
   <gwe:timeStamp>2012-08-23T15:58:09Z</gwe:timeStamp>
   <gwe:container>A.domain1.po1.100.0.1.0.1@19</gwe:container>
   <gwe:from>7.domain1.po1.100.0.1.0.1@16</gwe:from>
   <gwe:key>GWEvents</gwe:key>
   <gwe:uid>10761</gwe:uid>
   <gwe:type>Mail</gwe:type>
</gwe:event>
```

GroupWise Address Book

Because the GroupWise Address Book is not tied to a user's database, no GroupWise events are created when the GroupWise Address Book changes. Instead, it has its own database.

If you are interested in detecting when changes occur to the GroupWise Address Book, look at the GetDeltaInfoRequest and GetDeltasRequest methods in the *GroupWise SDK: Web Services (SOAP)* documentation.