Cisco webex

Best Practices for Integrations using Cisco WebEx APIs

(XML API and URL API)

Version 1.0

cisco

Copyright

Last updated: 092214

www.webex.com

Table of Contents

WebEx Public API Best Practices

WebEx public APIs are open and available for use and testing with the WebEx SaaS Cloud. In order to guide you through easy and secure integration with WebEx, this document captures tips and best practices on the usage of WebEx APIs.

Developer Onboarding (GDP, Partner ID)

Work with your CSM or PSM to join the Gold Developer Program (GDP). This will provide you with a development site and a partner ID for your production site.

Work with your account manager (CSM or PSM) to set up a meeting with the API support team. They will assist you during the integration and also provide you with a questionnaire to better understand your integration.

Partner ID

The partner ID (PID) is a configurable value assigned to your organization at the time the API is enabled on your site. This parameter is required by the AT=SU command as an identifier that the API commence has the authority to create user accounts without signing in as a site administrator. This is a very valuable piece of information that should be protected just as you would any crucial password. Because this is security-sensitive information, it should be routed through the WebEx Client Services Manager, who will make sure the contact receiving the information is approved.

Request URL Configuration

Developers should not hard-code the URL of the XML API to which they post requests. Integrations should be configurable to run on different WebEx customer sites, which have different domains. In addition, the service directory of the XML API URL is subject to change. For example, WebEx deploys new versions of the XML API to a "/preview" directory before replacing the current version at the regular directory to allow testing of integrations before migrating to the production site.

Thus, developers should use two configurable parameters to build the XML API URL to post their request:

https://[site_domain]/[xml_directory]

where site_domain = WebEx customer site (i.e., mwapi.webex.com), and xml_directory = Regular preview directory (i.e. WBXService/preview/XMLService or WBXService/XMLService).

What not to do

- Don't use site id; use site name in API calls wherever possible. Site IDs sometimes change.
- If using site name, then don't hard code it. Make it configurable.
- If using site id, then don't hard code it. Make it configurable.
- If using partner id, then don't hard code it. Make it configurable.

Authentication

WebEx APIs support the following authentication flows:

- Username/password-based authentication flows for non-SSO sites
- One-time login ticket for SSO/non-SSO sites (can be used with URL API)
- Site admin username/password flow for both SSO and non-SSO
- Session Ticket obtained through SAML Response for SSO sites
- Partner SAML Response-based Session Ticket (on behalf of multiple sites) for both SSO and non-SSO sites.

Use the appropriate flow for the site setup. If you are a third-party integrator whose application needs to work with multiple WebEx sites, leverage a Partner SAML flow.

What not to do

- Don't hard code username/password.
- Don't use the same password for each user.
- If storing user passwords, hash them.
- SSO sites should not use username/password flows.

Caching, BackURL Support (Cross Domain), and Global Redirect

- If you are using BU parameter in URL API, make sure the back URL is in the approved/trusted list of domains in Site Administration.
- Data from “getSite” can be cached as it doesn’t change often.
- You can cache usernames and meeting keys. Meeting keys work fine unless the meeting/series is deleted.
- If you are using a Global Redirect feature, which is sometimes used in conjunction with API integration and branding, we recommend keeping it simple (like exceptions for some key pages only). As the product evolves, new URLs and pages will be introduced and there cannot be a guarantee of backward compatibility.

Note: Global Redirect is NOT an API feature.

What not to do

- Don’t cache meeting links. Always get it from an API like getJoinURLMeeting/getHostURLMeeting
- Don’t cache recording links. Always get it from lstRecording or other APIs.

Throttling

We recommend no more than a thousand API calls in a minute.

Provisioning, Roles, and Accounts

- Use host accounts for scheduling meetings, meeting lists, join, host, etc.
- Use site administrator accounts or partner accounts to provision or pull history data.
- Provisioning on Non-SSO Sites:
  - Before provisioning a user through API, check if the user already exists or allow the user to provide a WebEx ID. Do not provision a user unless required.
  - When provisioning users using APIs, use the latest password policy for secure passwords. Allow users to select their own passwords.
  - Per security policies, it is highly recommended that passwords be changed periodically. Ensure your integration has a workflow, which will allow users to change their passwords either using APIs or the WebEx pages.
  - Ensure a workflow exists for de-activating users.
- Provisioning on SSO sites:
  - Unless using Just In Time, SSO sites could need APIs for provisioning.
  - Check to ensure username is unique.
  - Do not set a password for the user. It will be ignored.
  - If you are a third-party integrator, use Partner SAML flows for user-provisioning API calls.
  - Ensure a workflow exists for de-activating users.

- Non-authenticated operations are supported for a limited set of APIs and should be used to Join Meeting or Attendee Registration.
  - XML API: lstSummarySession, getAPIVersion, getJoinURLMeeting
  - URL API: m.php?AT=JM

Tip:

When de-provisioning/de-activating the user, rename the user's WebEx ID <johndoe> to <johndoe_date_time> and email address <johndoe@example.com> to <johndoe_date_time@example.com.old>. This will allow re-use of the old email address for provisioning a new user.

What not to do

- Do not use site admin accounts for scheduling, editing, retrieving meeting lists etc., on behalf of host.

CEP (Common Entry Point)

Common Entry Points are not really API but a URL to land on a WebEx page. We are discouraging use of these CEP links, as we will shortly be deprecating them.

Security

- When setting/editing meeting passwords or user passwords, obtain the password rules from WebEx using the *getSite* API.
- Always use POST for all API calls.
- Always use HTTPS for all API calls.
- For SSO sites, we only support SAML 2.0-based tickets in API calls.
- IP Referrer
  - Creating a host account using the URL API command (AT=SU) uses a server-to-server connection. To make sure the data passing over the wire is secure (in addition to using HTTPS/POST), you can request the use of IP Referrer on your WebEx site. The IP Referrer option checks the IP address from which any AT=SU command originates, and validates the source. If an AT=SU command does not come from a valid IP address, the request will be denied.
  - To use the IP Referrer option, you will need to provide WebEx with the IP address of your server, or the IP of the last exit point before the Internet if you use a firewall or proxy configuration. You have the option to provide one IP address or a range of IP addresses. Forward this information and requests for enabling the IP Referrer option on your site to your WebEx Client Services Manager.
- Domain Referrer
  - Although the login command (AT=LI) is not a server-to-server command, it does have an additional security option in the Domain Referrer. The Domain Referrer makes sure that no one, even a host with a valid login and password, can use the API to access your site unless they are on your domain. In other words, when you implement the Domain Referrer option, you will deny anyone outside your network the ability to sign in

using the WebEx API. If they know their WebEx ID (WID) and password (PW), they will be able to visit the WebEx site to sign in through the normal web interface, but will not be able to use the API. Requests for site changes should be directed to the WebEx Client Services Manager.

Note: The URL API is supported for use in browser-based applications. WebEx uses a number of methods in its process of operation that do not work well when the browser is not allowed to "do the work." If Java or similar technologies are used, the WebEx XML API is a better choice depending on the goal of the implementation.

Time Zones and Countries

- WebEx users can be simultaneously scattered across the globe. Therefore, all scheduled times need to specify a time zone. In the XML API, all WebEx time elements require an accompanying <timeZoneID> code. Many XML functions take a <timeZoneID> code and never parse the <timeZone> strings. The strings returned in the <timeZone> elements (such as "GMT-08:00, Pacific (San Jose)); list the city and hours relative to GMT during Standard Time. These <timeZone> string values are constant. Thus, during Daylight Savings Time, the <timeZone> string hours relative to GMT appear incorrect. If you plan to use these strings in your user interfaces, you must handle this conversion logic on your own. A more robust approach is to generate user interface strings from the <timeZoneID>.
- Some XML API use <country code> and <country name> for input. These inputs are validated. As a best practice, always pass a country code so the validation does not fail. See Appendix A of the XML API Guide to obtain the correct country codes.

Telephony/Audio Support

- There are more than WebEx-only telephony options supported by the product. Your integration should obtain telephony and VoIP information using *getSite* and *getUser* API and identify user's options for audio (*WebEx, TSP, Other*).
- When scheduling meetings, avoid setting "*None*" for the <telephonySupport> element unless the site/user is set up for that option.

Meeting Types

- The *CreateMeeting* API requires a <meetingType> value. Developers should not hard code this value. This is especially true if the integration is meant to run on multiple WebEx customer sites. Meeting types can vary based on the different types of meetings that each site and user has access to. In addition, WebEx customers with the newer Named-Host pricing model will have different meeting types than other pricing models.
- There are two approaches to handling this issue in partner integrations:
  - The <meetingType> value can be a user-selectable option. The user *GetUser* API returns all the meeting types available for a user. Integration can use this function to dynamically present the user with a choice of available meeting types for the site. Then, the *GetMeetingType* API can be called if required to get detailed information for each meeting type. The meeting type selected by the user is then specified in the meeting.CreateMeeting function.
  - The <meetingType> value can be set to a configurable default value. This approach hides the meeting type issue from the user, simplifying the scheduling operation and eliminating the need to train users on the meanings of each available meeting type. The configuration of the default meeting type should be in a site administration area of the integrated solution.
- Lastly, both the URL and XML APIs have optional elements to specify a meeting type when creating or updating a user. Again, partner applications should not hard-code meeting type values in these calls. If left unspecified, the user is created with access to meeting types available on their site as determined by their service and site administrator.

Large Volume of Response (Pagination) and Search Result Limiting

Many XML API query requests can potentially return hundreds or thousands of response records. For performance reasons, the WebEx XML API caps the maximum number of records that can be returned in a single query. Developer applications should make multiple queries to "step through" the result set until all matching records are retrieved.

All query requests have <listControl>, <startFrom>, and <maximumNum> elements. These elements allow each query to return a fixed subset of the total number response records. Each subsequent query can step through the next subset until all response records are returned. Developer applications should check the <totalRecords> response value and increase the <startFrom> value in each subsequent query until all records are returned.

Here is Java pseudocode that makes multiple requests to eventually retrieve the entire list of WebEx users for a site:

```
int segment=20; // get first record subset call
LstSummaryUser with

<listControl> <startFrom>1</startFrom>

<maximumNum>segment</maximumNum> </listControl>;

process LstsummaryUserReponse; int totalRecords = value
for <matchingRecords><total>;

// loop to get additional subsets for (int n = 1 +
segment; n < totalRecords; ++n) {
```

```
call LstSummaryUser with <listControl>
<startFrom>n</startFrom>
<maximumNum>segment</maximumNum> </listControl>;
process LstsummaryUserReponse; }
```

For more, review the XML API Guide section Global Request Elements on list controls

Development Sites and Testing

- As part of the GDP program, you will receive a development site.
- With every major/minor release, API enhancements will be communicated to all GDP contacts.
- While the API enhancements will be backward-compatible, always perform regression testing of your integration on the development sites during periods identified in the communication sent to you.
- Understand the limitations of development sites here.
- We strongly recommend that you do not perform schema validation in your integration using XML API. Schemas will change. The API servers perform validation anyway.

Error Handling, Troubleshooting, and GSB Status

- The WebEx XML API returns <result>FAILURE<result> when a request cannot execute successfully. XML API returns an <exception ID> code along with the exception <reason> string. When coding XML API exception handing, developer integrations should only process the <exceptionID> codes and NOT parse the exception reason strings. Exception reason strings are subject to change in future releases. Developers who are currently parsing the XML API exception <reason> strings should change their integration code to process the <exceptionID> instead.
- Always log date/time, exception ID, reason string, GSB status, and subError. This will be useful for troubleshooting in case there are problems with the integration.
- For a list of exception ID codes and reasons, refer to The XML API Developer Reference Guide, Appendix E.