Best Practices for Working with the SalesTim API

ABSTRACT
This article describes best practices that you can apply to help your applications get the most out of the SalesTim API - whether that involves improving app performance, or making your application more reliable for end users.

TABLE OF CONTENTS
[[toc]]

Handle responses effectively

Depending on the requests you make to SalesTim API, your applications should be prepared to handle different types of responses. The following are some of the most important practices to follow to ensure that your application behaves reliably and predictably for your end users.

Handling expected errors

While your application should handle all error responses (in the 400 and 500 ranges), pay special attention to certain expected errors and responses, listed in the following table.

Topic	HTTP error code	Best practice
User does not have access	403	If your application is up and running, it could encounter this error even if it has been granted the necessary permissions. In this case, it's most likely that the signed-in user or virtual app does not have privileges to access the resource requested. Your application should provide a generic "Access denied" error back to the signed-in user.
Not found	404	In certain cases, a requested resource might not be found. For example a resource might not exist, because it has not yet been provisioned or because it has been deleted.
Throttling	429	APIs might throttle at any time for a variety of reasons, so your application must **always** be prepared to handle 429 responses. This error response includes the `Retry-After` field in the HTTP response header. Backing off requests using the `Retry-After` delay is the fastest way to recover from throttling. For more information, see the Throttling article.

Topic	HTTP error code	Best practice
Service unavailable	503	This is likely because the services is busy. You should employ a back-off strategy similar to 429. Additionally, you should **always** make new retry requests over a new HTTP connection.

Evolvable enums

Client applications can be broken by the addition of members to an existing enum. For some newer enums in SalesTim API, a mechanism is available to allow for adding new members without incurring a breaking change. On these newer enums, you'll see a common *sentinel* member called `unknownFutureValue` that demarcates known and unknown enum members. Known members will have a number less than the sentinel member, while unknown members will be greater in value.

Reliability and support

To ensure reliability and facilitate support for your application, generate a unique GUID and send it on each SalesTim API REST request. This will help SalesTim investigate any errors more easily if you need to report an issue with SalesTim API. To do so, on every request to SalesTim API, generate a unique GUID, send it in the `client-request-id` HTTP request header, and also log it in your application's logs.