SOAP API Call Basics

The SOAP API calls implement specific knowledgebase operations that your client applications can invoke at runtime to perform tasks, such as:

- Search and read data in your knowledgebase.
- Add, update, and delete data in your knowledgebase.

Using your development environment, you can construct Web Service (WS) client applications that use the standard WS protocols to:

- Log in to the knowledgebase and receive the session token to be used for subsequent calls.
- Perform SQL-based searches across the knowledgebase data.
- Create, read, update, and delete data.
- Trigger workflow transitions and business rules processing.

Characteristics of API Calls

All SOAP API calls are:

- Service requests and responses - your client application prepares and submits a service request to the server via the API, the agiloft.com Web Service processes the request and returns a response, and the client application handles the response.
- Synchronous - once the API call is invoked, your client application waits until it receives a response from the service. Asynchronous calls are not supported.
- Committed automatically - every operation that writes to an agiloft.com object is committed automatically. This is analogous to the AUTOCOMMMIT setting in SQL. Each API call is done in one transaction.
- Followed by a delay inserted after the operation has completed. The delay is set to one second by default and is configurable via the global variable Web Services Delay (WSDelay).
    - An operation on a record may invoke rules and other functions, which need to be allowed enough time and resources to complete.
    - Additionally, the delay is needed because client applications could mistakenly used web services, resulting in a flood of requests.

Factors that Determine Data Access

When using the API, the following factors determine access to your organization's data:

- Your knowledgebase has to have SOAP API access enabled.
- The generated WSDL lists all data structures and fields that existed in the knowledgebase at the time of generation. The generated WSDL file contains all of the objects that are available in your knowledgebase. Via the API, a client application can access objects that are defined in your WSDL file.
- The generated WSDL is user-agnostic, but when invoking a method in the SOAP Interface with credentials of a certain user,The the runtime permissions of that user come into play. This may result in some fields appearing empty when the user doesn't have read permission, for example.

Run-time access via the SOAP API is governed by the same access permissions as the GUI. Certain operations on the tables and fields can be performed only if the combined permissions in the logged-in user's group list permit such access.
For example, values for the fields that are not visible to a given user are not returned in the results of the query. Another often reported issue is when one is not able to write a structure, retrieved in the very previous call, back to the server simply because the user has the necessary read permissions, but not the write ones.
Please refer to the Groups section in your User Manual or Online Help for full details regarding permissions.

- As such, the API respects table-level permissions configured for the given user group in the knowledgebase.
- Further, the API fully respects the ownership concept and related table-level and field-level permissions configured in the knowledgebase.
- Ownership for a record is determined in runtime at the time of the call and if modified in the call itself, comes into effect immediately upon the completion of the API call.
- Ownership changes to one object instance do not normally automatically cascade to other object instances unless specifically configured so via a Linked Field with changes propagation. For example, if ownership changes for a given Account, ownership does not then automatically change for any Contract associated with that Account - each ownership change must be made separately and explicitly by the client application.
- The configuration of the workflow for a given knowledgebase may influence the ability to create or delete a record in a certain state or to change the state of a record.
- Business rules, especially validation ones, may affect the success of create and modification operations.
- Success also depends on whether a particular change would compromise the referential integrity of your knowledgebase. For example, the data used in Linked Fields sets is validated in create and update calls. In a similar fashion the delete calls are handled in a way to ensure the integrity of the data is maintained and will fail otherwise.
- Certain features that affect the agiloft.com user interface are not accessible or implicitly enforced via the API. For example:

- Layouts define whether a given field is shown or hidden, but the API does not enforce such layout-specific field restrictions or validations in create and update calls. It is up to the client application to enforce any such constraints, if applicable.
- Record types can control which layouts users with different profiles can see. However, via API all fields available for the table are presented.
- Dependent choices and conditional visibility are not enforced by the API.

If any such constraints are required, it is up to the logic in the client application to enforce them explicitly.

Error Handling

In the event of an error the API calls return SOAP fault messages with additional information.
There are currently six types of faults distinguished by the SOAP API:

- EWIntegrityException

Signals that the knowledgebase setup has become incompatible with the client application logic/expectations.

- EWOperationException

Signals that the operation may not be performed because of some dependencies between agiloft.com functions. This message should be considered in the context of the operation. It may signal a permanent condition, for example workflow forbidding transition from one state to another; or a temporary one, for example the record is locked by another user from another GUI or API session.

- EWPermissionException

Signals that the username used to trigger the API call lacks sufficient privileges to perform certain operations.

- EWSessionException

Signals that client session has expired or has been removed.

- EWUnexpectedException

Signals that an unknown, not handled and unexpected exception has happened on the server side. These exceptions should be reported to the vendor for further investigation. The message contains a token that helps to trace the root cause of the problem.

- EWWrongDataException

Signals that data passed by client is wrong in the context of the operation.