

Alexa Integration

You can integrate your KB with private Alexa for Business skills to allow your users to ask their Alexa devices for updates on content in your KB. Custom skills can be designed for most basic operations, such as creating, updating, and deleting records, searching and reading data from existing records, counting records, and running actions.

To integrate with Alexa, first develop an Alexa skill for your business needs and then connect it to your KB in the Alexa Extension wizard. After you connect a skill to your KB, you can map system contacts with individual users to integrate their Alexa devices with the KB. Actions taken by an Alexa device are attributed to these users, or to the default user assigned in the integration settings, as a backup.

Prerequisites

This integration requires an Alexa for Business account for your organization, and a developer account to handle skill creation.

If you run Agiloft on your own server, that server must be accessible online; must accept HTTP requests on port 443; and must support HTTP over SSL/TLS, using an Amazon-trusted certificate. Your web service's domain name must be in the Subject Alternative Names (SANs) section of the certificate.

Initial Setup

When you first set up your integration, you need to create an Alexa skill, link it to your KB, link your KB users to your Alexa users, and configure the skill itself.

Creating Alexa Skills

To integrate with Alexa, you need to first create the Alexa skill you want to run with your system. At this stage, simply create and save an empty skill to generate a skill ID. Make sure to choose a blank template when prompted.

Refer to Amazon documentation if you need additional guidance on creating an Alexa skill.

Integrating Alexa Skills

When you've created one skill, integrate that skill with Agiloft.

First, find the ID of the skill in the Alexa Developer Console:

1. Log in to the console with your developer account.

2. Navigate to the list of your Alexa skills.
3. Under the skill name, click View Skill ID.



4. Copy the skill ID.

Now that you have the skill ID, you can integrate it with your system:

1. Log in to the Agiloft KB whose data you want to access with Alexa. Make sure your Alexa skills are integrated with only one Agiloft KB to make sure data is reported and updated as expected.
2. Go to Setup > Integration.
3. Under Alexa Extension, click Deploy and wait for the process to finish. This opens the configuration wizard.
4. In the Skill ID field, paste your skill ID.
5. Select a Default User to serve as the default attribution for Alexa actions if the Alexa user isn't recognized.

Linking Agiloft and Alexa Users

You can connect users across systems to give users access to the correct records and information when they use Alexa.

1. In Agiloft, go to Setup > Integration and under Alexa Extension, click Configure.
2. In the Alexa Users embedded table, click New.
3. In the Full Name field, select the user account in Agiloft. Make sure the user's login populates when you make your selection.
4. Enter the email address that user linked to their Alexa.
5. If you have the Alexa User ID, you can enter it here, but this ID is not necessary for setup. The Alexa User ID is difficult to locate and usually isn't available to the user. Note that this is not the AWS Account ID.

Configuring the Alexa Skill

With the skill linked to your KB, you need to finish creating the skill so it works. You can use this sample skill to compare to your own skills for syntax and structure: `demo-skill.json`

✔ This section contains a general outline about how to create Alexa skills, and tips for working specifically with Agiloft integration. For more details and guidance on working with Alexa skills, refer to the [Amazon documentation](#).

To set up your skill:

1. In the Alexa Developer Console, log in with your developer account.
2. Edit your skill.
3. Set up a Skill Invocation Name. This determines how users start their requests. For example, if your Skill Invocation Name is Agiloft, users would say "Alexa, ask Agiloft..." to begin their requests.
4. In the Intents section, click Add.

Intents

There are two supported methods for designing your intents:

- Include table and field names in the intent name, so calling the skill always acts on a specific table and, if applicable, field. This ensures Alexa acts on the correct system location every time, but it limits flexibility. To use this method, name your intents with this prefix naming convention: `[TableName]_[FieldName]_[Intent]`. `FieldName` is not always necessary, but `TableName` is necessary for all the supported intents. Note that logical table and field names should be converted from the default formatting, lowercase with underscores between words, to no underscores with the first letter of each word capitalized, so `company_document` becomes `CompanyDocument`.
- Use only the intent name and require the table and field information in slots. This makes the skill more flexible, but it increases the likelihood of Alexa acting on the wrong system location, or misunderstanding the user's request.

✔ For more information about intents and slots, refer to the [Amazon documentation](#).

When you've determined your approach, name your intent, using `[TableName]_[FieldName]_[Intent]` or simply the Intent name. Choose one of the intents supported by Agiloft integration.

Slots

With the intent defined, you can proceed to defining slots for your skill. Some slots can be resolved with pre-defined Alexa slot types, such as defining an ID slot using the AMAZON.NUMBER slot type. Dates can use AMAZON.DATE, and text inputs can use AMAZON.SearchQuery. However, AMAZON.SearchQuery is not appropriate for all text inputs because it sends parsed text from the user request, which might not be an exact match for choices and linked fields. If possible, provide a custom slot type for choices and linked fields, including possible values and value IDs from Agiloft.

VALUE 	ID (OPTIONAL) 	SYNONYMS (OPTIONAL) 		
Expired	3	Add synonym	+	
Canceled	2	Add synonym	+	
Renewed	1	Add synonym	+	
Signed	4	Add synonym	+	

If you chose not to use the prefix naming method for your Intents, make sure to accommodate table and field names in your slots. When those elements are accounted for, consider the recommended slots for your chosen intent, as well.

When you finish creating all the slots for your skill, configure the dialog settings for each slot:

1. For each slot, in the Actions column, click Edit Dialog.
2. Enable Slot Filling.
3. Provide content for Alexa speech prompts and User utterances. For more information on these, see the [Amazon documentation](#).

Utterances

Each intent also requires utterances. These are phrases that help Alexa recognize intent and slot values by defining common styles of phrasing expected from users. There aren't specific rules for utterances, but each supported intent provides examples. You can also refer to [Amazon documentation](#) for best practices.

Example

For example, consider these four sample utterances for an UpdateFieldIntent:

Intents / Contract_Wfstate_UpdateFieldIntent

Sample Utterances (4) 

What might a user say to invoke this intent?

Set contracts status to {contract_wfstate_value}

Set contract status to {contract_wfstate_value}

Update contract status to {contract_wfstate_value}

Update contract {id} status to {contract_wfstate_value}

If a user tells Alexa to "update contract 100 to Expired," it uses Contract_Wfstate_UpdateFieldIntent with slot values id = 100 and wfstate = Expired.

Dialog Delegation Strategy

For Agiloft skills, you need to set the Dialog Delegation Strategy to "disable auto delegation."

Permissions

In the Permissions section, enable Customer Name and select the Full Name option. Also enable Customer Email Address. These are used to access the user's email and link the user to their corresponding Agiloft account. When a user launches the skill, they are asked to give consent to share this information.

Finishing the Skill

With these components complete, finish up your skill:

1. Click Save Model.
2. Click Build Model.
3. Go to the Endpoint tab.

4. Select HTTPS as a Service Endpoint Type.
5. Provide your endpoint URL in the following format, where {host} is your server and {KBID} is the ID of your knowledgebase: `https://{host}/extension/alexa/connect/alexa/{KBID}`

 It's critical to enter your endpoint URL correctly. If you encounter problems testing your skill, double-check the endpoint URL entered here.

6. Select the SSL certificate type according to the application certificate.
7. Go to the Description tab.
8. Complete the required fields to provide a description of your skill. For more details, refer to the [Amazon documentation](#).
9. Go to the Availability tab.
10. Share the skill with your Alexa for Business organization.

Publishing and Testing the Skill

When everything is ready, submit your skill. This process can take several hours. After the skill is submitted and verified, you can enable it for your organization. These steps are required every time you make changes to your skill.

You can test your skill in the Test tab, including the current published version and the working version, if you have changes under development. This tab can show you user requests, server responses, and device logs. When you test in-development versions, make sure to save and build your skill model before testing.

Agiloft Intents

This section details each supported Alexa intent, including suggested slots and utterance examples.

CreateRecordIntent

This intent is designed to create records. Prefix naming is not required.

Prefix Naming: {TableName}_CreateRecordIntent. For example, Contract_CreateRecordIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.

Utterance examples for Contract_CreateRecordIntent:

- create new contract

Utterance examples for CreateRecordIntent:

- create new {table}
- create new record + dialog for table slot

Response:

{table} {id} created. For example, "contract 101 created." The record ID is saved in the Alexa session.

ReadFieldIntent

This intent is designed to read a record's field value. Prefix naming is recommended.

Prefix Naming: {TableName}_{Field}_ReadFieldIntent. For example, Contract_Wfstate_ReadFieldIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- field: If you don't use prefix naming, provide the field name using a slot value. In this case, create a Contract_Fields slot type with possible field values and assign it to the slot.
- id: Record ID. This slot is required. Use the AMAZON.NUMBER slot type.

Utterance examples for Contract_Wfstate_ReadFieldIntent:

- what is the status of contract {id}?
- what is the status of the contract? (if the record ID is already in the user's session)

Utterance examples for Contract_ReadFieldIntent:

- what is the {field} of contract {id}?

Utterance examples for ReadFieldIntent:

- what is the {field} of {table} {id}?

Response:

{table} {id} {field} is {value}. For example, "contract 101 status is Draft." The record ID is saved in the Alexa session.

UpdateFieldIntent

This intent is designed to update a field in a record. Prefix naming is recommended.

Prefix Naming: {TableName}_{Field}_UpdateFieldIntent. For example, Contract_Wfstate_UpdateFieldIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- field: If you don't use prefix naming, provide the field name using a slot value. In this case, create a Contract_Fields slot type with possible field values and assign it to the slot.
- id: Record ID. This slot is required. Use the AMAZON.NUMBER slot type.
- {table}_{field}_value: This is the pattern for the value slot name, such as contract_wfstate_value. Create a value slot for each field this intent can update.

Utterance examples for Contract_Wfstate_UpdateFieldIntent:

- update status of contract {id} to {contract_wfstate_value}
- update status of contract to {contract_wfstate_value} (if the record ID is already in the user's session)

Utterance examples for Contract_UpdateFieldIntent:

- update contract (Alexa will ask for the field and value)

Confirmation:

The user will have to confirm the request. Alexa will ask: "I will change {table} {id} {field} to {value}. Is this correct?"

Response:

{table} {id} updated. For example, "contract 101 updated." The record ID is saved in the Alexa session.

DeleteRecordIntent

This intent is designed to delete table records. Prefix naming is not required.

Prefix Naming: {TableName}_DeleteRecordIntent. For example, Contract_DeleteRecordIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- id: Record ID. This slot is required. Use the AMAZON.NUMBER slot type.

Utterance examples for Contract_DeleteRecordIntent:

- remove contract {id}

Utterance examples for DeleteRecordIntent:

- remove {table} {id}?

Confirmation:

The user will have to confirm the request. Alexa will ask: "I will remove {table} {id}. Is this correct?"

Response:

{table} {id} removed. For example, "contract 101 removed."

CountRecordsIntent

This intent is designed to count records by filter. Prefix naming is not required.

Prefix Naming: {TableName}_{Field}_ CountRecordsIntent. For example, Contract_Wfstate_CountRecordsIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- field: If you don't use prefix naming, provide the field name using a slot value. In this case, create a Contract_Fields slot type with possible field values and assign it to the slot.
- id: Record ID. This slot is required. Use the AMAZON.NUMBER slot type.
- {table}_{field}_filter: This is the pattern for the filter slot name, such as contract_wfstate_filter. Create a filter slot for each field this intent should work with to filter records.

Utterance examples for Contract_Wfstate_CountRecordsIntent:

- count contracts with status {contract_wfstate_filter}

Utterance examples for Contract_CountRecordsIntent:

- count contracts with {field} {contract_wfstate_filter}

Response:

There are {count} records in {table} Table with {field} {filter}. For example, "There are 10 records in contracts Table with status Draft."

ActionIntent

This intent is designed to apply actions to records. Prefix naming is not required.

Prefix Naming: {TableName}_{Action}_ActionIntent. For example, Contract_Cancel_ActionIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- action: If you don't use prefix naming, provide the action name using a slot value. In this case, create a Contract_Actions slot type with possible action values and assign it to the slot.
- id: Record ID. This slot is required. Use the AMAZON.NUMBER slot type.

Utterance examples for Contract_Cancel_ActionIntent:

- Cancel contract {id}

Utterance examples for Contract_ActionIntent:

- {action} contract {id}

Response:

Action applied. The record ID is saved in the Alexa session.

SavedSearchIntent

This intent is designed to work with saved searches. Prefix naming is not required.

Prefix Naming: {TableName}_{SearchName}_SavedSearchIntent. For example, Contracts_MyDrafts_SavedSearchIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- search: If you don't use prefix naming, provide the search name using a slot value. In this case, create a Contract_Search slot type with possible search values and assign it to a slot.
- limit: The number of records to return. This slot is optional, and without it, 1 is used by default. Use the AMAZON.NUMBER slot type.

Utterance examples for Contracts_MyDrafts_SavedSearchIntent:

- what is my next contract with draft status?

Response:

Found {table} {ids}. For example, "Found contracts 101, 102."

CopyRecordIntent

This intent is designed to copy records. Prefix naming is not required.

Prefix Naming: {TableName}_CopyRecordIntent. For example, Contract_CopyRecordIntent.

Slots:

- table: If you don't use prefix naming, provide the table name using a slot value. In this case, create a Tables slot type and assign it to the slot.
- id: Record ID. This slot is required. Use the AMAZON.NUMBER slot type.

Utterance examples for Contract_CopyRecordIntent:

- copy contract {id}

Utterance examples for CopyRecordIntent:

- create record {id} + dialog for table slot

Response:

{table} {id} created. For example, "contract 101 created." The record ID is saved in the Alexa session.