**Implement custom skills and include them in a skillset**



A skillset defines operations that generate textual content and structure from documents that contain images or unstructured text. Examples are optical character recognition (OCR) for images, entity recognition for undifferentiated text, and text translation. A skillset executes after text and images are extracted from an external data source, and after [field mappings](https://learn.microsoft.com/en-us/azure/search/search-indexer-field-mappings) are processed.

This article explains how to create a skillset using [REST APIs](https://learn.microsoft.com/en-us/rest/api/searchservice/skillsets/create), but the same concepts and steps apply to other programming languages.

Rules for skillset definition include:

* Must have a unique name within the skillset collection. A skillset is a top-level resource that can be used by any indexer.
* Must have at least one skill. Three to five skills are typical. The maximum is 30.
* A skillset can repeat skills of the same type. For example, a skillset can have multiple Shaper skills.
* A skillset supports chained operations, looping, and branching.

Indexers drive skillset execution. You need an [indexer](https://learn.microsoft.com/en-us/azure/search/search-howto-create-indexers), [data source](https://learn.microsoft.com/en-us/azure/search/search-data-sources-gallery), and [index](https://learn.microsoft.com/en-us/azure/search/search-what-is-an-index) before you can test your skillset.

**Add a skillset definition**

Start with the basic structure. In the [Create Skillset REST API](https://learn.microsoft.com/en-us/rest/api/searchservice/skillsets/create), the body of the request is authored in JSON and has the following sections:

A screenshot of a computer

Description automatically generated

After the name and description, a skillset has four main properties:

* skills array, an unordered [collection of skills](https://learn.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills). Skills can be utilitarian (like splitting text), transformational (based on AI from Azure AI services), or custom skills that you provide. An example of a skills array is provided in the next section.
* cognitiveServices is used for [billable skills](https://learn.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills) that call Azure AI services APIs. Remove this section if you aren't using billable skills or Custom Entity Lookup. If you are, attach [an Azure AI multi-service resource](https://learn.microsoft.com/en-us/azure/search/cognitive-search-attach-cognitive-services).
* knowledgeStore (optional) specifies an Azure Storage account and settings for projecting skillset output into tables, blobs, and files in Azure Storage. Remove this section if you don't need it, otherwise [specify a knowledge store](https://learn.microsoft.com/en-us/azure/search/knowledge-store-create-rest).
* encryptionKey (optional) specifies an Azure Key Vault and [customer-managed keys](https://learn.microsoft.com/en-us/azure/search/search-security-manage-encryption-keys) used to encrypt sensitive content (descriptions, connection strings, keys) in a skillset definition. Remove this property if you aren't using customer-managed encryption.

**Add skills**

Inside the skillset definition, the skills array specifies which skills to execute. Three to five skills are common, but you can add as many skills as necessary, subject to [service limits](https://learn.microsoft.com/en-us/azure/search/search-limits-quotas-capacity#indexer-limits).

The end result of an enrichment pipeline is textual content in either a search index or a knowledge store. For this reason, most skills either create text from images (OCR text, captions, tags), or analyze existing text to create new information (entities, key phrases, sentiment). Skills that operate independently are processed in parallel. Skills that depend on each other specify the output of one skill (such as key phrases) as the input of a second skill (such as text translation). The search service determines the order of skill execution and the execution environment.

All skills have a type, context, inputs, and outputs. A skill might optionally have a name and description. The following example shows two unrelated [built-in skills](https://learn.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills) so that you can compare the basic structure.

A screenshot of a computer

Description automatically generated

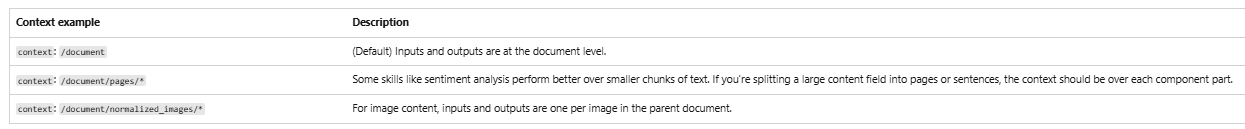
Each skill is unique in terms of its input values and the parameters that it takes. [Skill reference documentation](https://learn.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills) describes all of the parameters and properties of a given skill. Although there are differences, most skills share a common set and are similarly patterned.

**Set skill context**

Each skill has a [context property](https://learn.microsoft.com/en-us/azure/search/cognitive-search-working-with-skillsets#skill-context) that determines the level at which operations take place. If the context property isn't explicitly set, the default is "/document", where the context is the whole document (the skill is called once per document).



The context property is usually set to one of the following examples:



Context also determines where outputs are produced in the [enrichment tree](https://learn.microsoft.com/en-us/azure/search/cognitive-search-working-with-skillsets#enrichment-tree). For example, the Entity Recognition skill returns a property called organizations, captured as orgs. If the context is "/document", then an organizations node is added as a child of "/document". If you then want to reference this node in downstream skills, the path is "/document/orgs".

**Define inputs**

Skills read from and write to an enriched document. Skill inputs specify the origin of the incoming data. It's often the root node of the enriched document. For blobs, a typical skill input is the document's content property.

[Skill reference documentation](https://learn.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills) for each skill describes the inputs it can consume. Each input has a name that identifies a specific input, and a source that specifies the location of the data in the enriched document. The following example is from the Entity Recognition skill:



* Skills can have multiple inputs. The name is the specific input. For Entity Recognition, the specific inputs are *text* and *languageCode*.
* The source property specifies which field or row provides the content to be processed. For text-based skills, the source is a field in the document or row that provides text. For image-based skills, the node providing the input is normalized images.

A close up of a text

Description automatically generated

If the skill iterates over an array, both context and input source should include /\* in the correct positions.

**Define outputs**

Each skill is designed to emit specific kinds of output, which are referenced by name in the skillset. A skill output has a name and an optional targetName.

[Skill reference documentation](https://learn.microsoft.com/en-us/azure/search/cognitive-search-predefined-skills) for each skill describes the outputs it can produce. The following example is from the Entity Recognition skill:

A white rectangular object with a black border

Description automatically generated

* Skills can have multiple outputs. The name property identifies a specific output. For example, for Entity Recognition, output can be *persons*, *locations*, *organizations*, among others.
* The targetName property specifies the name you would like this node to have in the enriched document. This is useful if skill outputs have the same name. If you have multiple skills that return the same output, use targetName for name disambiguation in enrichment node paths. If the target name is unspecified, the name property is used for both.

Some situations call for referencing each element of an array separately. For example, suppose you want to pass *each element* of "/document/orgs" separately to another skill. To do so, add an asterisk to the path: "/document/orgs/\*".

Skill output is written to the enriched document as a new node in the enrichment tree. It might be a simple value, such as a sentiment score or language code. It could also be a collection, such as a list of organizations, people, or locations. Skill output can also be a complex structure, as is the case with the Shaper skill. The inputs of the skill determine the composition of the shape, but the output is the named object, which can be referenced in a search index, a knowledge store projection, or another skill by its name.

**Add a custom skill**

This section includes an example of a [custom skill](https://learn.microsoft.com/en-us/azure/search/cognitive-search-custom-skill-web-api). The URI points to an Azure Function, which in turn invokes the model or transformation that you provide. For more information, see [Add a custom skill to an Azure AI Search enrichment pipeline](https://learn.microsoft.com/en-us/azure/search/cognitive-search-custom-skill-interface).

Although the custom skill executes code that is external to the pipeline, in a skills array, it's just another skill. Like the built-in skills, it has a type, context, inputs, and outputs. It also reads and writes to an enrichment tree, just as the built-in skills do. Notice that the context field is set to "/document/orgs/\*" with an asterisk, meaning the enrichment step is called *for each* organization under "/document/orgs".

Output, such as the company description in this example, is generated for each organization that's identified. When referring to the node in a downstream step (for example, in key phrase extraction), you would use the path "/document/orgs/\*/companyDescription" to do so.

A screenshot of a computer

Description automatically generated

**Send output to a destination**

Although skill output can be optionally cached for reuse purposes, it's usually temporary and exists only while skill execution is in progress.

* To send output to a field in a search index, [create an output field mapping](https://learn.microsoft.com/en-us/azure/search/cognitive-search-output-field-mapping) in an indexer.
* To send output to a knowledge store, [create a projection](https://learn.microsoft.com/en-us/azure/search/knowledge-store-projection-overview).
* To send output to a downstream skill, reference the output by its node name, such as "/document/organization", in the downstream skill's input source property. See [Reference an annotation](https://learn.microsoft.com/en-us/azure/search/cognitive-search-concept-annotations-syntax) for examples.

**Tips for a first skillset**

* Try the [Import data wizard](https://learn.microsoft.com/en-us/azure/search/search-get-started-portal) or [Import and vectorize data wizard](https://learn.microsoft.com/en-us/azure/search/search-get-started-portal-import-vectors).

The wizards automate several steps that can be challenging the first time around. It defines the skillset, index, and indexer, including field mappings and output field mappings. It also defines projections in a knowledge store if you're using one. For some skills, such as OCR or image analysis, the wizard adds utility skills that merge the image and text content that was separated during document cracking.

After the wizard runs, you can open each object in the Azure portal to view its JSON definition.

* Try [Debug Sessions](https://learn.microsoft.com/en-us/azure/search/cognitive-search-debug-session) to invoke skillset execution over a target document and inspect the enriched document that the skillset creates. You can view and modify input and output settings and values. This tutorial is a good place to start: [Tutorial: Debug a skillset using Debug Sessions](https://learn.microsoft.com/en-us/azure/search/cognitive-search-tutorial-debug-sessions).