

# Azure AI Search の検索インデックス

[アーティクル] • 2024/02/16

Azure AI Search の "検索インデックス" は検索可能なコンテンツであり、検索エンジンでインデックス作成、全文検索、ベクトル検索、ハイブリッド検索、フィルターされたクエリに使用できます。インデックスは、スキーマによって定義され、検索サービスに保存されます。2番目のステップとしてデータのインポートが続けます。このコンテンツは検索サービス内に存在します。これは、最新の検索アプリケーションで想定されるミリ秒単位の応答時間に必要な、プライマリデータストアとは別のことです。インデクサー主導のインデックス作成シナリオを除き、検索サービスがソースデータに接続したり、クエリを実行したりすることはありません。

検索インデックスを作成して管理する場合、この記事は次の点を理解するのに役立ちます。

- コンテンツ (ドキュメントおよびスキーマ)
- 物理データ構造
- 基本操作

今すぐに使いたいですか? 代わりに、[検索インデックスの作成](#)に関する記事を参照してください。

## 検索インデックスのスキーマ

Azure AI Search のインデックスには検索ドキュメントが格納されます。概念的に、ドキュメントはインデックス内で検索可能なデータの1つの単位です。たとえば、小売業者に製品ごとのドキュメントがあり、ニュース組織に記事ごとのドキュメントがある場合、旅行サイトにはホテルと目的地ごとのドキュメントがある場合があります。これらの概念をなじみのあるデータベースの同等のものに対応させるなら、検索インデックスはテーブルと同じで、ドキュメントはテーブルにおける行とほぼ同じです。

次の例に示すように、ドキュメントの構造は "インデックス スキーマ" によって決まります。"フィールド" コレクションは通常、インデックスの最大の部分であり、各フィールドには、名前、[データ型](#)の割り当て、および使用方法を決定する許容される動作を示す属性が設定されます。

JSON

```
{  
  "name": "name_of_index, unique across the service",  
  "fields": [  
    {
```

```

        "name": "name_of_field",
        "type": "Edm.String | Collection(Edm.String) | Collection(Edm.Single)
| Edm.Int32 | Edm.Int64 | Edm.Double | Edm.Boolean | Edm.DateTimeOffset |
Edm.GeographyPoint",
        "searchable": true (default where applicable) | false (only Edm.String
and Collection(Edm.String) fields can be searchable),
        "filterable": true (default) | false,
        "sortable": true (default where applicable) | false
(Collection(Edm.String) fields cannot be sortable),
        "facetable": true (default where applicable) | false
(Edm.GeographyPoint fields cannot be facetable),
        "key": true | false (default, only Edm.String fields can be keys),
        "retrievable": true (default) | false,
        "analyzer": "name_of_analyzer_for_search_and_indexing", (only if
'searchAnalyzer' and 'indexAnalyzer' are not set)
        "searchAnalyzer": "name_of_search_analyzer", (only if 'indexAnalyzer'
is set and 'analyzer' is not set)
        "indexAnalyzer": "name_of_indexing_analyzer", (only if
'searchAnalyzer' is set and 'analyzer' is not set)
        "normalizer": "name_of_normalizer", (applies to fields that are
filterable)
        "synonymMaps": "name_of_synonym_map", (optional, only one synonym map
per field is currently supported)
        "dimensions": "number of dimensions used by an embedding models",
(applies to vector fields only, of type Collection(Edm.Single))
        "vectorSearchProfile": "name_of_vector_profile" (indexes can have many
configurations, a field can use just one)
    }
],
"suggesters": [ ],
"scoringProfiles": [ ],
"analyzers":(optional)[ ... ],
"charFilters":(optional)[ ... ],
"tokenizers":(optional)[ ... ],
"tokenFilters":(optional)[ ... ],
"defaultScoringProfile": (optional) "...",
"corsOptions": (optional) { },
"encryptionKey":(optional){ },
"semantic":(optional){ },
"vectorSearch":(optional){ }
}

```

簡潔にするために他の要素は折りたたまれていますが、次のリンク先で詳細を確認できます。

- `suggesters` は、オートコンプリートのような先行入力クエリをサポートします。
- `scoringProfiles` は関連性のチューニングに使われます。
- `analyzers` は、アナライザーがサポートする言語規則やその他の特性に従って文字列をトークンに処理するために使われます。
- `corsOptions`、つまりクロスオリジンリモートスクリプト (CORS) は、さまざまなドメインから要求を発行するアプリに使われます。

- `encryptionKey` は、インデックス内の機密コンテンツの二重暗号化を構成します。
- `semantic` は、全文検索とハイブリッド検索でのセマンティック再ランク付けを構成します。
- `vectorSearch` は、ベクトルフィールドとクエリを構成します。

## フィールド定義

検索ドキュメントは、[インデックス要求の作成](#)に関する記事の本文の "フィールド" コレクションによって定義されます。ドキュメントの識別のためのフィールド(キー)、検索可能なテキストの格納、フィルター、ファセット、並べ替えをサポートするためのフィールドが必要になります。ユーザーに表示しないデータのフィールドが必要になる場合もあります。たとえば、検索スコアを上げるためにスコアリングプロファイルで使用できる、利益率やマーケティングプロモーションのフィールドが必要になることがあります。

受信データが階層化された性質を持つ場合は、入れ子構造に使われる複合型として、インデックス内でそれを表すことができます。あらかじめ登録されているサンプルデータセットである Hotels は、各ホテルとの一対一のリレーションシップを持つ Address(複数のサブフィールドを含む) と、各ホテルに複数の部屋が関連付けられている複合型コレクションの Rooms を使用した複合型を示しています。

## フィールド属性

フィールド属性は、フィールドがどのように使用されるか(フルテキスト検索、ファセットナビゲーション、並べ替えなどの操作で使用されるかどうか)を決定します。

文字列フィールドは多くの場合、"検索可能" および "取得可能" としてマークされます。検索結果を絞り込むために使用されるフィールドには、"並べ替え可能"、"フィルター可能"、および "ファセット可能" が含まれます。

[+] テーブルを展開する

属性	説明
"検索可能"	全文またはベクトル検索可能。テキストフィールドは、インデックス作成時に単語分割などの字句解析の対象になります。検索可能フィールドを "sunny day" などの値に設定した場合、その値は内部的に個別のトークン "sunny" と "day" に分割されます。 詳細については、「 <a href="#">フルテキスト検索のしくみ</a> 」を参照してください。
"フィルター可能"	\$filter クエリで参照されます。型 <code>Edm.String</code> または <code>Collection(Edm.String)</code> のフィルター可能フィールドは単語分割されないため、比較は完全に一致するかどうかだけになります。たとえば、このようなフィールドを "sunny day" に設定した場合、\$filter=f

属性	<code>eq 'sunny'</code> では一致が見つかりませんが、 <code>\$filter=f eq 'sunny day'</code> では見つかります。
"並べ替え可能"	既定では、システムは結果をスコアで並べ替えますが、ドキュメント内のフィールドに基づいて並べ替えを構成できます。型 <code>Collection(Edm.String)</code> のフィールドを "並べ替え可能" にすることはできません。
"フル可"能"	通常、カテゴリ (たとえば、特定の市にあるホテル) ごとのヒットカウントを含む検索結果のプレゼンテーションで使用されます。このオプションは、型 <code>Edm.GeographyPoint</code> のフィールドでは使用できません。"フィルター可能"、"並べ替え可能"、または "フル可"能" である型 <code>Edm.String</code> のフィールドの長さは、最大 32 キロバイトです。詳細については、「 <a href="#">Create Index (REST API) (インデックスの作成 (REST API))</a> 」を参照してください。
"キー"	インデックス内のドキュメントの一意識別子。キー フィールドとして正確に 1 つのフィールドを選択する必要があります、それは型 <code>Edm.String</code> である必要があります。
"取得可能"	検索結果でこのフィールドを返すことができるかどうかを決定します。これは、あるフィールド ("利幅" など) をフィルター、並べ替え、またはスコア付けのメカニズムとして使用するが、このフィールドをエンドユーザーには表示したくない場合に役立ちます。 <code>true for key</code> である必要があります。

いつでも新しいフィールドを追加できますが、既存のフィールド定義はインデックスの有効期間の間ロックされます。このため、開発者は通常、単純なインデックスを作成したり、アイデアをテストしたり、ポータルページを使用して設定を検索したりするためのポータルを使用します。インデックスを容易に再構築できるようにコードベースのアプローチに従う場合は、インデックス設計を頻繁に反復する方がより効率的です。

### ① 注意

インデックスの作成に使用する API には、さまざまな既定の動作があります。REST API の場合、ほとんどの属性は既定で有効であり (たとえば、文字列フィールドの "searchable" および "retrievable" は `true` です)、無効にする場合は、単にそれらを設定するだけです。.NET SDK の場合は、逆のことが言えます。明示的に設定していないプロパティの場合、既定では、特に有効にしない限り、対応する検索動作は無効にされています。

## 物理的な構造とサイズ

Azure AI Search におけるインデックスの物理的な構造は、主に内部実装です。そのスキーマにアクセスし、そのコンテンツにクエリを実行し、そのサイズを監視し、容量を

管理することができますが、クラスター自体(インデックス、[シャード](#)、その他のファイルとフォルダー)は、Microsoft が内部で管理します。

インデックス サイズを監視するには、Azure portal の [インデックス] タブを使用するか、検索サービスに対して [GET INDEX 要求](#)を発行します。 [サービス統計情報要求](#)を発行し、ストレージ サイズの値を確認することもできます。

インデックスのサイズは、次の条件によって決まります。

- ドキュメントの数量と構成
- 個々のフィールドの属性
- インデックスの構成(具体的には、suggerer を含めるかどうか)

ドキュメントの構成と数量は、インポートに選択した内容によって決まります。検索インデックスには検索可能なコンテンツのみを含める必要があります。ソースデータにバイナリ フィールドが含まれている場合は、AI エンリッチメントを使用してコンテンツを解読して分析し、テキスト検索可能な情報を作成する場合を除き、それらのフィールドは除外してください。

フィールドの属性によって動作が決まります。これらの動作をサポートするために、インデックス作成プロセスによって、必要なデータ構造が作成されます。たとえば、`Edm.String` 型のフィールドの場合、"検索可能" によって、トーカン化された用語の転置インデックスをスキャンする[全文検索](#)が呼び出されます。これに対して、"フィルター可能" または "並べ替え可能" の属性では、未変更の文字列に対する反復処理がサポートされます。次のセクションの例では、選択した属性に基づくインデックス サイズのバリエーションを示しています。

[suggerer](#) は、先行入力またはオートコンプリートのクエリをサポートするコンストラクトです。そのため、suggerer を含めると、インデックス作成プロセスによって、逐語的な文字の照合に必要なデータ構造が作成されます。suggerer はフィールドレベルで実装されるため、先行入力にふさわしいフィールドのみを選択してください。

## 属性と suggerer がストレージに与える影響を示す例

次のスクリーンショットは、属性のさまざまな組み合わせの結果であるインデックス格納パターンを示しています。このインデックスは[不動産サンプル インデックス](#)に基づいています。これは、データのインポート ウィザードと組み込みのサンプル データを使用して簡単に作成できます。インデックスのスキーマは表示されませんが、インデックス名に基づいて属性を推測できます。たとえば、`realestate-searchable` インデックスでは "searchable" 属性が選択されていて他には何もなく、`realestate-retrievable` インデックスでは "retrievable" 属性が選択されていて他には何もなく、以下同様です。

NAME	DOCUMENT COUNT	STORAGE SIZE
realestate-all-attributes-no-suggester	4,959	26.55 MiB
realestate-all-attributes-plus-suggester	4,959	49.4 MiB
realestate-filterable-facetable-sortable	4,959	20.89 MiB
realestate-no-attributes	4,959	4.99 MiB
realestate-retrievable	4,959	5.04 MiB
realestate-searchable	4,959	9.95 MiB

これらのインデックスのバリエーションはやや人為的なものですが、属性がストレージに与える影響の広範な比較のために参照できます。

- "取得可能" は、インデックスのサイズに影響しません。
- "フィルター可能"、"並べ替え可能"、"ファセット可能" は、より多くのストレージを消費します。
- `suggester` では、インデックス サイズが大きくなる可能性が大きいにありますが、スクリーンショットで示すほどではありません。`(suggester` 対応になる可能性のあるすべてのフィールドが選択されていますが、ほとんどのインデックスではこのようなシナリオになる可能性はありません)。

また、上記の表に反映されていない事柄に、[アナライザー](#)の影響があります。

`edgeNgram` トークナイザーを使って逐語的な文字シーケンス (`a`, `ab`, `abc`, `abcd`) を格納した場合、インデックスは、標準アナライザーを使用した場合よりも大きくなります。

## 基本的な操作と相互作用

インデックスの概要を理解したので、このセクションでは、1つのインデックスに接続してセキュリティを保護するなどの、インデックスの実行時操作について説明します。

### ① 注意

インデックスを管理する際、インデックスの移動やコピーに関して、ポータルや API のサポートはないことに注意してください。代わりに、ユーザーは通常、アプリケーション デプロイソリューションを別の検索サービスでポイントする(同じインデックス名を使用している場合)か、名前を変更して現在の検索サービスにコピーを作成してからビルドします。

## インデックスの分離性

Azure AI Search で操作の対象となるインデックスは一度に 1 つです。インデックスに関連したすべての操作は、単一のインデックスが対象となります。関連するインデックスや、インデックス作成またはクエリのための独立したインデックスの結合の概念はありません。

## 継続的に使用可能

インデックスは、最初のドキュメントのインデックスが作成されるとすぐにクエリで使用できますが、すべてのドキュメントのインデックスが作成されるまでは完全には機能しません。内部的には、検索インデックスは [パーティション間で分散され、レプリカ上で実行されます](#)。物理インデックスは内部で管理されます。論理インデックスはユーザーが管理します。

インデックスは継続的に使用可能であり、一時停止したり、オフラインにしたりすることはできません。継続的な操作のために設計されているので、コンテンツの更新やインデックス自体への追加はリアルタイムで行われます。その結果、要求がドキュメントの更新と一致する場合、クエリは一時的に不完全な結果を返す可能性があります。

ドキュメント操作 (更新または削除) や、現在のインデックスの既存の構造と整合性に影響しない変更 (新しいフィールドの追加など) に対しては、クエリの継続性が存在します。構造上の更新 (既存のフィールドの変更) を行う必要がある場合は、通常、開発環境での削除と再構築のワークフローを使用して、または運用サービスでインデックスの新しいバージョンを作成することによってそれらが管理されます。

[インデックスの再構築](#)を避けるため、小規模な変更を行っている一部のお客様は、以前のバージョンと共に新しいものを作成することによって、フィールドの "バージョン管理" を選択しています。これは、時間の経過と共に、特にレプリケートに負荷のかかる運用環境のインデックスで、古いフィールドまたは古いカスタム アナライザー定義の形式の、孤立したコンテンツになります。インデックス ライフサイクル管理の一部として、インデックスの計画更新に関する問題に対処することができます。

## エンドポイントの接続とセキュリティ

すべてのインデックス作成とクエリの要求は、インデックスを対象とします。エンドポイントは、通常、次のいずれかになります。

[+] [テーブルを展開する](#)

エンドポイント	接続とアクセスの制御
<your-service>.search.windows.net/indexes	インデックスのコレクションを対象とします。インデックスを作成、一覧表示、または削除するときに使用します。これらの操作には管理者権限が必要です。管理者 API キーまたは Search 共同作成者ロールを通じて使用できます。
<your-service>.search.windows.net/indexes/<your-index>/docs	1 つのインデックスのドキュメントコレクションを対象とします。インデックスまたはデータ更新に対してクエリを実行するときに使用します。クエリには、読み取り権限で十分であり、クエリ API キーまたはデータ閲覧者ロールを通じて使用できます。データ更新の場合は、管理者権限が必要です。

検索サブスクリバイバー (つまり検索サービスの作成者) は、Azure portal で検索サービスを管理できます。サービスを作成または削除するには、Azure サブスクリプションに共同作成者以上のアクセス許可が必要です。検索サービスに直接接続するには、[Azure portal にサインイン](#)します。

他のクライアントの場合は、接続手順のクイックスタートを確認することをお勧めします。

- [クイックスタート: REST](#)
- [クイックスタート: Azure SDK](#)

## 次のステップ

Azure AI Search のほぼすべてのサンプルまたはチュートリアルを使用して、インデックスを作成する実践的な体験ができます。まず、目次から任意のクイックスタートを選択できます。

ただし、データを使用してインデックスを読み込む方法についても理解しておく必要があります。インデックスの定義とデータのインポートの方法は、連携して定義されます。次の記事では、インデックスの作成および読み込みの詳細について説明します。

- [検索インデックスの作成](#)
- [ベクトルストアを作成する](#)
- [インデックスの別名を作成する](#)
- [データインポートの概要](#)

- インデックスを読み込む

# Azure AI Search でのベクター ストレージ

[アーティクル] • 2024/03/04

Azure AI 検索には、[ベクトル検索とハイブリッド検索用のベクトルストレージ](#)と構成が用意されています。サポートはフィールド レベルで実装されます。つまり、同じ検索コーパスでベクター フィールドと非ベクター フィールドを組み合わせることができます。

ベクターは検索インデックスに格納されます。[Create Index REST API](#) または同等の Azure SDK メソッドを使用して、[ベクター ストアを作成します。](#)

ベクター ストレージの主な考慮事項は以下のとおりです。

- 目的とするベクトル取得パターンに基づいて、ユース ケースに合うスキーマを設計します。
- インデックス サイズを見積もり、検索サービスの容量を確認します。
- ベクトルストアを管理する
- ベクトルストアをセキュリティで保護する

## ベクター取得パターン

Azure AI Search には、検索結果を操作するための 2 つのパターンがあります。

- 生成検索。言語モデルは、Azure AI Search のデータを使用して、ユーザーのクエリに対する応答を作成します。このパターンには、プロンプトを調整し、コンテキストを維持するためのオーケストレーションレイヤーが含まれます。このパターンでは、検索結果はプロンプト フローにフィードされ、GPT や Text-Davinci などのチャット モデルが受け取ります。このアプローチは、検索インデックスによってグラウンド データが提供される[取得拡張生成 \(RAG\)](#) アーキテクチャに基づいています。
- 検索バー、クエリ入力文字列、レンダリングされた結果を使用した従来の検索。検索エンジンによって、ベクトル クエリが受け入れられて実行され、応答が作成されます。ユーザーはそれらの結果をクライアント アプリにレンダリングします。Azure AI Search では、結果はフラット化された行セットで返され、検索結果を含めるフィールドを選ぶことができます。チャット モデルがないため、応答で人間が判読できる非ベクトル コンテンツをベクトルストア (検索インデックス) に設定することが期待されます。検索エンジンはベクトルに一致しますが、非ベクトル値を使用して検索結果を設定する必要があります。[ベクトル クエリとハイ](#)

[ブリッド クエリ](#)は、従来の検索シナリオ用に作成できるクエリ要求の型に対応します。

インデックススキーマには、主なユースケースが反映されている必要があります。次のセクションでは、生成AIと従来の検索用に構築されるソリューションのフィールド構成の違いに着目します。

## ベクトルストアのスキーマ

ベクトルストアのインデックススキーマには、名前、キー フィールド(文字列)、1つ以上のベクトルフィールド、およびベクトル構成が必要です。非ベクター フィールドは、ハイブリッド クエリ、または言語モデルを通過する必要のない、人間が読み取り可能な逐語的なコンテンツを返す場合に推奨されます。ベクター構成の手順については、「[ベクターストアを作成する](#)」を参照してください。

## 基本的なベクター フィールドの構成

ベクトルフィールドは、データ型とベクトル固有のプロパティによって区別されます。フィールドコレクション内のベクトルフィールドの外観を以下に示します。

```
JSON
{
  "name": "content_vector",
  "type": "Collection(Edm.Single)",
  "searchable": true,
  "retrievable": true,
  "dimensions": 1536,
  "vectorSearchProfile": "my-vector-profile"
}
```

ベクトルフィールドの型は `Collection(Edm.Single)` です。

ベクトルフィールドは検索可能で取得可能である必要がありますが、フィルター可能、ファセット可能、並べ替え可能にすることはできません。また、アナライザー、ノーマライザー、シノニムマップの割り当てを持つことはできません。

ベクトルフィールドでは、埋め込みモデルによって生成される埋め込みの数に `dimensions` を設定する必要があります。たとえば、text-embedding-ada-002では、テキストのチャunkごとに1,536個の埋め込みが生成されます。

ベクトルフィールドには、"ベクトル検索プロファイル"によって示されるアルゴリズムを使用してインデックスが作成されます。このプロファイルはインデックス内の他の

場所で定義されているため、例では示されていません。 詳細については、[ベクトル検索の構成](#)に関するページを参照してください。

## 基本的なベクトル ワークロードのフィールド コレクション

ベクトルストアでは、ベクトルフィールド以外にもさらにフィールドが必要です。 たとえば、キー フィールド (この例では `"id"`) はインデックス要件です。

JSON

```
"name": "example-basic-vector-idx",
"fields": [
  { "name": "id", "type": "Edm.String", "searchable": false, "filterable": true, "retrievable": true, "key": true },
  { "name": "content_vector", "type": "Collection(Edm.Single)", "searchable": true, "retrievable": true, "dimensions": 1536, "vectorSearchProfile": null },
  { "name": "content", "type": "Edm.String", "searchable": true, "retrievable": true, "analyzer": null },
  { "name": "metadata", "type": "Edm.String", "searchable": true, "filterable": true, "retrievable": true, "sortable": true, "facetable": true }
]
```

`"content"` フィールドなどの他のフィールドでは、人間が判読できる `"content_vector"` フィールドと同等のものが提供されます。 応答の作成専用の言語モデルを使用している場合は、非ベクトルコンテンツ フィールドを省略できますが、クライアント アプリに検索結果を直接プッシュするソリューションには非ベクトルコンテンツが必要です。

メタデータ フィールドは、特にメタデータにソース ドキュメントに関する配信元情報が含まれている場合に、フィルターに役立ちます。 ベクトルフィールド上で直接フィルター処理を行うことはできませんが、ベクトルクエリの実行前または実行後にフィルター処理を行うプリフィルター モードまたはポストフィルター モードを設定することができます。

## データのインポートとベクター化ウィザードによって生成されるスキーマ

評価と概念実証のテストには、[データのインポートとベクター化ウィザード](#)をお勧めします。 ウィザードによって、このセクションのスキーマ例が生成されます。

このスキーマの偏りは、検索ドキュメントがデータ チャンクを中心に構築されていることです。RAG アプリで一般的なように、言語モデルが応答を作成する場合は、データ チャンクを中心に設計されたスキーマが必要です。

データ チャンクは、言語モデルの入力制限内を維持するために必要ですが、複数の親 ドキュメントからプルされたコンテンツの小さなチャンクに対してクエリを照合できる場合の類似性検索の精度も向上します。最後に、セマンティック ランク付けを使用している場合、セマンティック ランカーにはトークン制限もあります。これは、データ チャンクがアプローチの一部である場合に、より簡単に満たされます。

次の例では、検索ドキュメントごとに 1 つのチャンク ID、チャンク、タイトル、ベクター フィールドがあります。blob メタデータ (パス) の base 64 エンコードを使用して、chunkID と親 ID がウィザードによって設定されます。チャンクとタイトルは、BLOB コンテンツと BLOB 名から派生します。ベクター フィールドのみが完全に生成されます。これは、ベクトル化されたバージョンのチャンク フィールドです。埋め込みは、指定した Azure OpenAI 埋め込みモデルを呼び出すことによって生成されます。

#### JSON

```
"name": "example-index-from-import-wizard",
"fields": [
    { "name": "chunk_id", "type": "Edm.String", "key": true, "searchable": true, "filterable": true, "retrievable": true, "sortable": true, "facetable": true, "analyzer": "keyword"}, 
    { "name": "parent_id", "type": "Edm.String", "searchable": true, "filterable": true, "retrievable": true, "sortable": true}, 
    { "name": "chunk", "type": "Edm.String", "searchable": true, "filterable": false, "retrievable": true, "sortable": false}, 
    { "name": "title", "type": "Edm.String", "searchable": true, "filterable": true, "retrievable": true, "sortable": false}, 
    { "name": "vector", "type": "Collection(Edm.Single)", "searchable": true, "retrievable": true, "dimensions": 1536, "vectorSearchProfile": "vector-1707768500058-profile"}]
```

## RAG アプリとチャットスタイル アプリのスキーマ

生成検索用のストレージを設計する場合は、インデックスを作成してベクトル化した静的コンテンツに対して個別のインデックスを作成し、プロンプト フローで使用できる会話用に 2 つ目のインデックスを作成できます。以下のインデックスは、[chat-with-your-data-solution-accelerator](#) アクセラレータから作成されます。

Home > contosochat-search

contosochat-search | Indexes

Search service

Search management

Add index Refresh Delete

Filter by name...

Name	Document Count	Storage Size
conversations	14	434.61 KB
chat-index	191	5.42 MB

Indexes  
Indexers  
Data sources  
Aliases

生成検索エクスペリエンスをサポートするチャットインデックスのフィールド:

JSON

```
"name": "example-index-from-accelerator",
"fields": [
    { "name": "id", "type": "Edm.String", "searchable": false, "filterable": true, "retrievable": true },
    { "name": "content", "type": "Edm.String", "searchable": true, "filterable": false, "retrievable": true },
    { "name": "content_vector", "type": "Collection(Edm.Single)", "searchable": true, "retrievable": true, "dimensions": 1536, "vectorSearchProfile": "my-vector-profile" },
    { "name": "metadata", "type": "Edm.String", "searchable": true, "filterable": false, "retrievable": true },
    { "name": "title", "type": "Edm.String", "searchable": true, "filterable": true, "retrievable": true, "facetable": true },
    { "name": "source", "type": "Edm.String", "searchable": true, "filterable": true, "retrievable": true },
    { "name": "chunk", "type": "Edm.Int32", "searchable": false, "filterable": true, "retrievable": true },
    { "name": "offset", "type": "Edm.Int32", "searchable": false, "filterable": true, "retrievable": true }
]
```

オーケストレーションとチャット履歴をサポートする会話インデックスのフィールド:

JSON

```
"fields": [
    { "name": "id", "type": "Edm.String", "key": true, "searchable": false, "filterable": true, "retrievable": true, "sortable": false, "facetable": false },
    { "name": "conversation_id", "type": "Edm.String", "searchable": false, "filterable": true, "retrievable": true, "sortable": false, "facetable": true },
    { "name": "content", "type": "Edm.String", "searchable": true, "filterable": false, "retrievable": true },
    { "name": "content_vector", "type": "Collection(Edm.Single)", "searchable": true, "retrievable": true, "dimensions": 1536,
```

```

"vectorSearchProfile": "default-profile" },
  { "name": "metadata", "type": "Edm.String", "searchable": true,
"filterable": false, "retrievable": true },
  { "name": "type", "type": "Edm.String", "searchable": false,
"filterable": true, "retrievable": true, "sortable": false, "facetable":
true },
  { "name": "user_id", "type": "Edm.String", "searchable": false,
"filterable": true, "retrievable": true, "sortable": false, "facetable":
true },
  { "name": "sources", "type": "Collection(Edm.String)", "searchable":
false, "filterable": true, "retrievable": true, "sortable": false,
"facetable": true },
  { "name": "created_at", "type": "Edm.DateTimeOffset", "searchable":
false, "filterable": true, "retrievable": true },
  { "name": "updated_at", "type": "Edm.DateTimeOffset", "searchable":
false, "filterable": true, "retrievable": true }
]

```

次に示すのは、[Search Explorer](#)における会話インデックスの検索結果のスクリーンショットです。検索に条件がないため、検索スコアは 1.00 となっています。オーケストレーションとプロンプト フローをサポートするために存在するフィールドに注目してください。会話 ID は、特定のチャットを特定します。"type" は、コンテンツがユーザーとアシスタントのどちらからのものであるかを示します。日付は、古さによって履歴からチャットを削除するために使用されます。

```

Results
18  {
19    "@search.score": 1,
20    "id": "NDImODY4MjgtOMU2ZC00YTY3LTgwNTiTNmI2OWUyYzllZjRj",
21    "conversation_id": "01db26eb-f781-462b-8da3-0ec10e551a35",
22    "content": "The Gulf Stream carries a lot of heat from the Equator toward the far North Atlantic, n",
23    "metadata": "{\"conversation_id\": \"01db26eb-f781-462b-8da3-0ec10e551a35\", \"sources\": [\"doc_26",
24    "type": "assistant",
25    "user_id": null,
26    "sources": [
27      "doc_2005da260b463009d5e230b09a55acedf51fbcd7",
28      "doc_541c34b9a54b97ac888034a21c73fc6910243741"
29    ],
30    "created_at": "2024-01-15T05:19:52Z",
31    "updated_at": "2024-01-15T05:19:52Z"
32  },
33  {
34    "@search.score": 1,
35    "id": "NDM1NjM0MjZGIMZi00ZmQ4LTk3YTgtY2MyMGU3YmRhNTVm",
36    "conversation_id": "01db26eb-f781-462b-8da3-0ec10e551a35",
37    "content": "what can you tell me about winds",
38    "metadata": "{\"type\": \"user\", \"conversation_id\": \"01db26eb-f781-462b-8da3-0ec10e551a35\", \"",
39    "type": "user",
40    "user_id": null,
41    "sources": [],
42    "created_at": "2024-01-15T19:51:12Z",
43    "updated_at": "2024-01-15T19:51:12Z"

```

## 物理的な構造とサイズ

Azure AI Search におけるインデックスの物理的な構造は、主に内部実装です。スキーマへのアクセス、コンテンツの読み込みとクエリ実行、サイズの監視、容量の管理を行うことはできますが、クラスター自体 (逆インデックスとベクトルインデックス、シャ

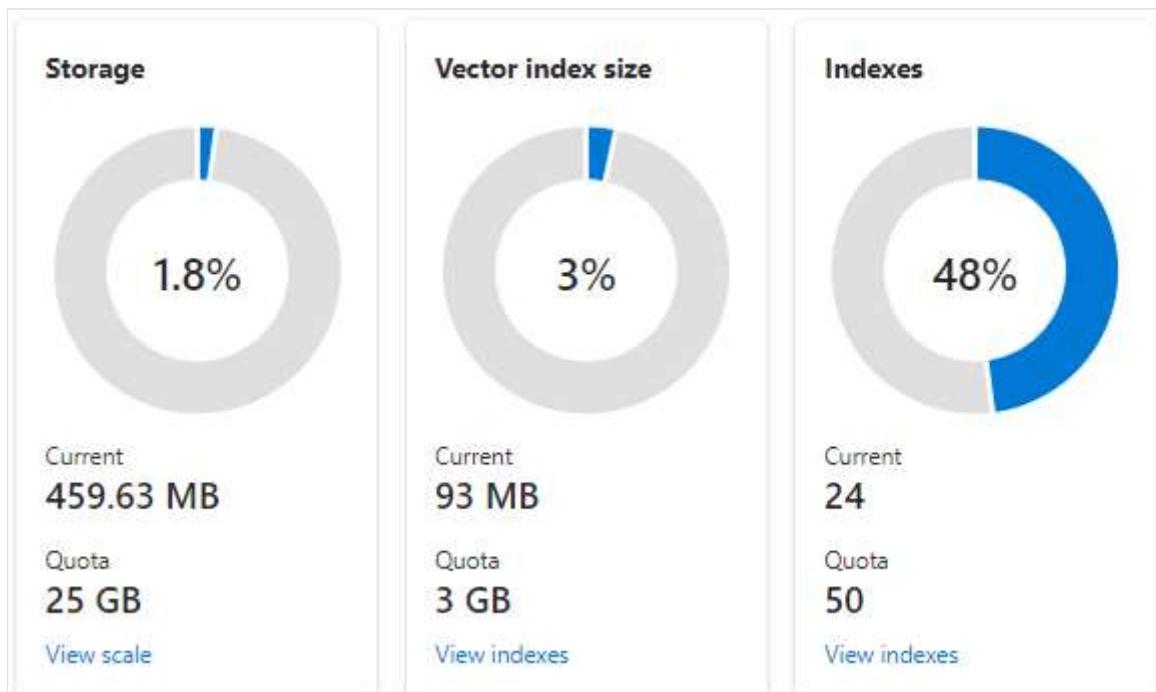
ード、その他のファイルとフォルダー) は、Microsoft によって内部的に管理されます。

インデックスのサイズと内容は、以下によって決まります。

- ドキュメントの数量と構成
- 個々のフィールドの属性。たとえば、フィルター処理可能なフィールドにはより多くのストレージが必要です。
- 類似性検索に HNSW と網羅的 KNN のどちらを選択するかに基づくインデックス構成 (内部的なナビゲーション構造がどのように作成されるかを指定するベクトル構成を含む)。

Azure AI 検索はベクトルストレージに制限を課しており、これは、すべてのワークコードにとってバランスのとれた安定したシステムを維持するのに役立ちます。利用者が制限の範囲内に留まるのを手助けするために、ベクトルの使用状況は Azure portal においてと、サービスとインデックスの統計情報を使用したプログラム的な方法とで個別に追跡され報告されます。

次に示すのは、1つのパーティションと1つのレプリカで構成された S1 サービスのスクリーンショットです。この特定のサービスには、平均して1つのベクトルフィールドを含む24個の小さなインデックスがあり、各フィールドは1536個の埋め込みで構成されています。2番目のタイルが示しているのは、ベクトルインデックスのクオータと使用状況です。ベクトルインデックスは、各ベクトルフィールドに対して作成された内部的なデータ構造です。そのため、ベクトルインデックスのストレージは常に、インデックス全体によって使用されるストレージの一部となります。残りの部分は、その他の非ベクトルフィールドとデータ構造によって使用されます。



ベクトルインデックスの制限と見積もりについては[別の記事](#)でカバーされていますが、前もって強調しておくべき 2 つの点として、ストレージの最大値はサービス レベルによって異なるということと、検索サービスがいつ作成されたかによても異なることがあります。同じレベルでも新しいサービスの方が、かなり多いベクトルインデックスの容量を持ちます。このため、以下のアクションを行ってください。

- [検索サービスのデプロイ日を確認する](#)。2023 年 7 月 1 日より前に作成されている場合は、容量を増やすために新しい検索サービスを作成することを検討してください。
- ベクトルストレージ要件の変動が予想される場合、[スケーラブルなレベルを選択する](#)。Basic レベルは、1 つのパーティションに固定されています。柔軟性を高め、パフォーマンスを向上させるには、Standard 1 (S1) 以上を検討してください。

## 基本的な操作と相互作用

このセクションでは、1 つのインデックスへの接続やセキュリティ保護など、ベクトルの実行時操作について紹介します。

### ① 注意

インデックスを管理する際、インデックスの移動やコピーに関して、ポータルや API のサポートはないことに注意してください。代わりに、ユーザーは通常、アプリケーション デプロイ ソリューションを別の検索サービスでポイントする(同じインデックス名を使用している場合)か、名前を変更して現在の検索サービスにコピーを作成してからビルドします。

## 継続的に使用可能

インデックスは、最初のドキュメントのインデックスが作成されるとすぐにクエリで使用できますが、すべてのドキュメントのインデックスが作成されるまでは完全には機能しません。内部的には、インデックスは[複数のパーティションにわたって分散され、レプリカ上で実行されます](#)。物理インデックスは内部で管理されます。論理インデックスはユーザーが管理します。

インデックスは継続的に使用可能であり、一時停止したり、オフラインにしたりすることはできません。継続的な操作のために設計されているので、コンテンツの更新やインデックス自体への追加はリアルタイムで行われます。その結果、要求がドキュメントの更新と一致する場合、クエリは一時的に不完全な結果を返す可能性があります。

ドキュメント操作 (更新または削除) や、現在のインデックスの既存の構造と整合性に影響しない変更 (新しいフィールドの追加など) に対しては、クエリの継続性が存在します。構造上の更新 (既存のフィールドの変更) を行う必要がある場合は、通常、開発環境での削除と再構築のワークフローを使用して、または運用サービスでインデックスの新しいバージョンを作成することによってそれらが管理されます。

[インデックスの再構築](#)を避けるため、小規模な変更を行っている一部のお客様は、以前のバージョンと共に新しいものを作成することによって、フィールドの "バージョン管理" を選択しています。これは、時間の経過と共に、特にレプリケートに負荷のかかる運用環境のインデックスで、古いフィールドまたは古いカスタム アナライザ一定義の形式の、孤立したコンテンツになります。インデックス ライフサイクル管理の一部として、インデックスの計画更新に関する問題に対処することができます。

## エンドポイント接続

ベクトルのインデックス作成とクエリ要求はすべて、インデックスを対象とします。エンドポイントは、通常、次のいずれかになります。

[+] [テーブルを展開する](#)

エンドポイント	接続とアクセスの制御
<code>&lt;your-service&gt;.search.windows.net/indexes</code>	インデックスのコレクションを対象とします。インデックスを作成、一覧表示、または削除するときに使用します。これらの操作には管理者権限が必要です。管理者 API キーまたは <a href="#">Search 共同作成者ロール</a> を通じて使用できます。
<code>&lt;your-service&gt;.search.windows.net/indexes/&lt;your-index&gt;/docs</code>	1 つのインデックスのドキュメントコレクションを対象とします。インデックスまたはデータ更新に対してクエリを実行するときに使用します。クエリには、読み取り権限で十分であり、クエリ API キーまたはデータ閲覧者ロールを通じて使用できます。データ更新の場合は、管理者権限が必要です。

## Azure AI 検索への接続方法

1. [アクセス許可](#)または [API アクセスキー](#)があることを確認します。既存のインデックスに対してクエリを実行する場合を除き、検索サービスのコンテンツを管理および表示するには、管理者権限または共同作成者ロールの割り当てが必要です。

2. [Azure portal](#) から開始します。検索サービスを作成したユーザーは、[アクセス制御 (IAM)] ページを使用して他のユーザーにアクセスを許可するなど、検索サービスを表示および管理できます。
3. プログラムによるアクセスのために他のクライアントに移動します。最初の手順としては、以下のクイックスタートとサンプルをお勧めします。
  - [クイックスタート: REST](#)
  - [ベクトルのサンプル](#)

## ベクトルデータへのアクセスをセキュリティで保護する

Azure AI 検索では、データ暗号化、インターネットなしのシナリオ用のプライベート接続、Microsoft Entra ID を介した安全なアクセスのためのロールの割り当てが実装されています。エンタープライズセキュリティ機能の全容については、「[Azure AI 検索のセキュリティ](#)」で説明されています。

## ベクトルストアを管理する

Azure には、診断ログとアラートを含む[監視プラットフォーム](#)が用意されています。推奨するベストプラクティスを次に示します。

- [診断ログを有効にする](#)
- [アラートを設定する](#)
- [クエリとインデックスのパフォーマンスを分析する](#)

## 関連項目

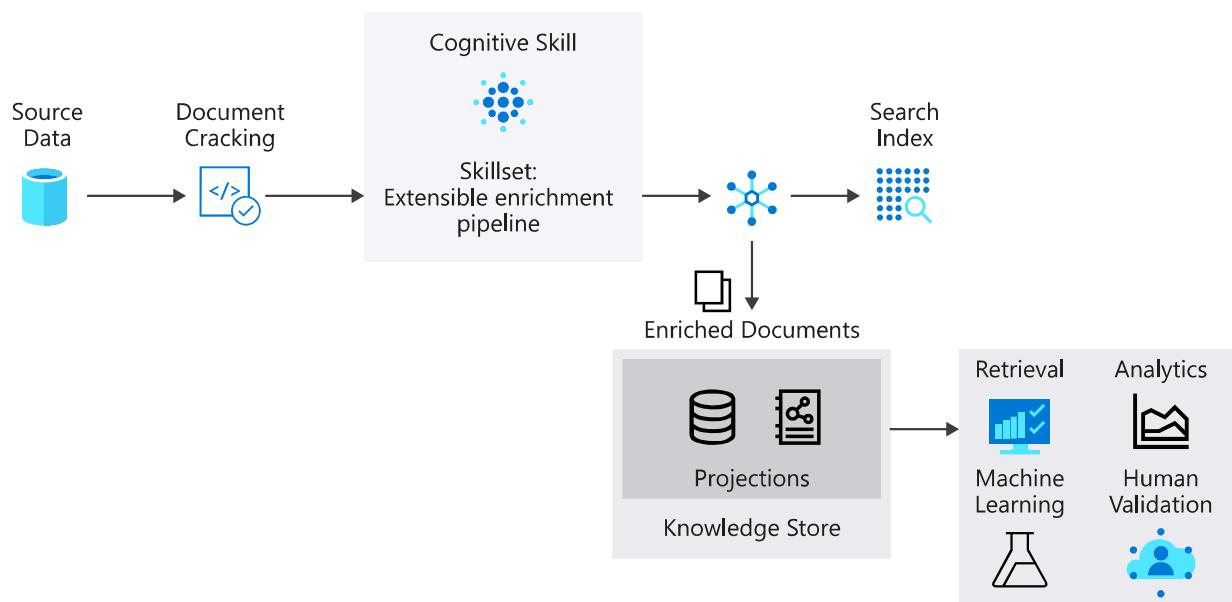
- [REST API を使用してベクトルストアを作成する \(クイックスタート\)](#)
- [ベクトルストアを作成する](#)
- [ベクトルストアにクエリを実行する](#)

# Azure AI Search 内のナレッジストア

[アーティクル] • 2024/01/10

ナレッジストアは、Azure AI Search のスキルセットによって作成された AI エンリッチコンテンツのセカンダリストレージです。 Azure AI Search では、インデックス作成ジョブは常に出力を検索インデックスに送信しますが、インデクサーにスキルセットをアタッチする場合は、必要に応じて、AZURE Storage のコンテナーまたはテーブルに AI エンリッチされた出力を送信することもできます。 ナレッジストアは、ナレッジマイニングなどの検索以外のシナリオで、独立した分析またはダウンストリーム処理に使用できます。

インデックス作成の 2 つの出力 (検索インデックスとナレッジストア) は、同じパイプラインの相互排他的な製品です。 これらは同じ入力から派生し、同じデータを含みますが、その内容は構造化され、格納され、さまざまなアプリケーションで使用されます。



物理的には、ナレッジストアは [Azure Storage](#) です。つまり Azure Table Storage か Azure Blob Storage、またはその両方になります。 Azure Storage に接続できるすべてのツールまたはプロセスは、ナレッジストアのコンテンツを使用できます。

Azure portal を使用して表示すると、ナレッジストアはテーブル、オブジェクト、またはファイルの他のコレクションのようになります。 次のスクリーンショットは、3 つのテーブルで構成されるナレッジストアを示しています。 プレフィックスなどの名前付け規則を `kstore` 採用して、コンテンツをまとめることができます。

The screenshot shows the Azure Storage browser (preview) interface for a storage account named 'demoblobstorage'. The left sidebar includes links for Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, and Storage browser (preview). The main area displays blobstorage, favorites, recently viewed, blob containers, file shares, queues, and tables. The 'Tables' section is highlighted with a red box. It shows a list of tables: kstoreProjectionDemoDocument, kstoreProjectionDemoEntities, kstoreProjectionDemoKeyPhrases, and MsAzSearchIndexerCacheIndex33b0d... . A search bar at the top right says 'Search tables by prefix'.

## ナレッジストアのメリット

ナレッジストアの主な利点は、コンテンツに柔軟にアクセスできることと、データを形成する機能という2つの点にあります。

Azure AI Search のクエリでのみアクセスできる検索インデックスとは異なり、ナレッジストアには、Azure Storageへの接続をサポートする任意のツール、アプリ、またはプロセスからアクセスできます。この柔軟性によって、エンリッチメントパイプラインによって生成された、分析およびエンリッチメントされたコンテンツを消費するための新しいシナリオが開きます。

データをエンリッチする同じスキルセットを、データの形成にも使用できます。Power BIのようなツールは、テーブルの方が適していますが、データサイエンスワークフローには BLOB 形式の複雑なデータ構造が必要になる場合があります。スキルセットに Shaper スキルを追加すると、データのシェイプを制御できるようになります。そして、このシェイプをテーブルや BLOB などのプロジェクトに渡すことで、データの使用目的に沿った物理的なデータ構造を作成することができます。

次のビデオでは、これらの利点の両方について説明します。

<https://www.youtube-nocookie.com/embed/XWzLBP8iWqg?version=3>

## ナレッジストアの定義

ナレッジストアは、スキルセット定義内で定義されており、2つのコンポーネントがあります。

- Azureストレージの接続文字列

- ナレッジストアがテーブル、オブジェクト、ファイルのいずれで構成されているかを決定する[プロジェクト](#)。プロジェクト要素は配列です。1つのナレッジストア内に、テーブル、オブジェクト、ファイルの組み合わせを複数セット作成することができます。

JSON

```
"knowledgeStore": {
    "storageConnectionString": "<YOUR-AZURE-STORAGE-ACCOUNT-CONNECTION-STRING>",
    "projections": [
        {
            "tables": [ ],
            "objects": [ ],
            "files": [ ]
        }
    ]
}
```

この構造体で指定するプロジェクトの種類は、ナレッジストアが使用するストレージの種類を決定しますが、その構造体は決定しません。テーブル、オブジェクト、およびファイルのフィールドは、ナレッジストアをプログラムで作成する場合は Shaper スキルの出力によって決定され、ポータルを使用している場合はデータのインポート ウィザードによって決定されます。

- `tables` は、エンリッチメントされたコンテンツを Table Storage に投影します。分析ツールへの入力のために表形式のレポート構造が必要な場合や、データフレームとして他のデータストアにエクスポートする場合は、テーブルプロジェクトを定義します。同じプロジェクトグループ内の複数の `tables` を指定して、エンリッチメントされたドキュメントのサブセットまたは断面を取得することができます。同じプロジェクトグループ内では、テーブルのリレーションシップが保持されるため、すべてのテーブルを操作できます。

プロジェクトされたコンテンツは集計または正規化されません。次のスクリーンショットは、キー フレーズで並べ替えられたテーブルを示しており、隣接する列に親ドキュメントが示されています。インデックス作成中のデータインジェストとは対照的に、言語分析やコンテンツの集計はありません。複数形と大文字と小文字の違いは、一意のインスタンスと見なされます。

Content.metadata_storage_name	Content.KeyPhrases
Cognitive Services and Content Intelligence.pptx	Computer Vision
10-K-FY16.html	computing device
10-K-FY16.html	computing devices
MSFT_FY17_10K.docx	computing devices
10-K-FY16.html	Computing segment
Cognitive Services and Bots (spanish).pdf	confianza

- `objects` では、JSON ドキュメントを BLOB ストレージに投影します。 `object` の物理的表現は、エンリッチメントされたドキュメントを表す階層型の JSON 構造体です。
- `files` では、イメージファイルを BLOB ストレージに投影します。 `file` は、ドキュメントから抽出され、BLOB ストレージにそのまま転送されるイメージです。 "ファイル" という名前ですが、ファイルストレージではなく Blob Storage に表示されます。

## ナレッジストアの作成

ナレッジストアを作成するには、ポータルまたは API を使用します。

Azure Storage、スキルセット、インデクサーが必要になります。インデクサーには検索インデックスが必要なので、インデックス定義も指定する必要があります。

完成したナレッジストアへの最短ルートとしては、ポータル アプローチを採用してください。または、オブジェクトがどのように定義され、関連しているかをより深く理解するには、REST API を選択します。

Azure Portal

データのインポート ウィザードを使用して、[4つの手順で最初のナレッジストアを作成します。](#)

1. エンリッチするデータを含むデータ ソースを定義します。
2. スキルセットを定義します。スキルセットにより、エンリッチメントステップとナレッジストアが指定されます。
3. インデックススキーマを定義します。これは必要ない場合もありますが、インデクサーでは必要です。このウィザードではインデックスを推測できます。

4. ウィザードの完了。この最後のステップで、抽出、エンリッチメント、ナレッジストアの作成が行われます。

このウィザードを使用すると、いくつかのタスクを自動化できます。具体的には、整形とプロジェクトの両方 (Azure Storage 内の物理データ構造の定義) が作成されます。

## アプリに接続する

エンリッチされたコンテンツがストレージに存在するようになると、Azure Blob に接続する任意のツールまたはテクノロジを使用して、コンテンツを探索、分析、または使用できます。次の一覧が開始点です。

- エンリッチされたドキュメント構造とコンテンツを表示するための Azure portal の [Storage Explorer](#) またはストレージブラウザー (プレビュー)。これは、ナレッジストアのコンテンツを表示するためのベースライン ツールと考えてください。
- レポートと分析のための [Power BI](#)。
- さらに操作するための [Azure Data Factory](#)。

## コンテンツのライフサイクル

インデクサーとスキルセットを実行するたび、スキルセットまたは基になるソース データが変更された場合、ナレッジストアが更新されます。インデクサーによって取得された変更は、エンリッチメントプロセスを通じてナレッジストア内のプロジェクトに反映され、投影されたデータが元のデータ ソース内のコンテンツの現在の表現になります。

### ① 注意

プロジェクト内のデータを編集することができますが、ソース データ内のドキュメントが更新された場合、次のパイプライン呼び出しですべての編集が上書きされます。

## ソース データの変更

変更の追跡をサポートするデータ ソースの場合、インデクサーは新規および変更されたドキュメントを処理し、既に処理されている既存のドキュメントをバイパスします。タイムスタンプ情報はデータ ソースによって異なりますが、BLOB コンテナーでは、イ

インデクサーによって `lastmodified` の日付が確認され、取り込む必要がある BLOB が特定されます。

## スキルセットの変更

スキルセットに変更を加える場合は、[エンリッチメントされたドキュメントのキャッシュを有効にして](#)、可能な限り既存のエンリッチメントを再利用する必要があります。

増分キャッシュを使用しない場合、インデクサーは常に高いウォーター マークの順に逆戻りせずドキュメントを処理します。BLOB の場合、インデクサーは、インデクサーの設定やスキルセットに対する変更に関係なく、`lastModified` で並べ替えた BLOB を処理します。スキルセットを変更した場合、以前に処理されたドキュメントは、新しいスキルセットを反映するように更新されません。スキルセットの変更後に処理されたドキュメントでは新しいスキルセットが使用され、その結果、インデックス ドキュメントには古いスキルセットと新しいスキルセットが混在します。

増分キャッシュを使用する場合、スキルセットの更新後に、インデクサーはスキルセットの変更の影響を受けないエンリッチメントを再利用します。アップストリーム エンリッチメントは、変更されたスキルから独立して分離されたエンリッチメントと同様に、キャッシュからプルされます。

## 削除

インデクサーは、Azure Storage 内の構造とコンテンツを作成および更新しますが、それらを削除しません。インデクサーまたはスキルセットが削除された場合でも、プロジェクトは引き続き存在します。ストレージ アカウントの所有者は、不要になったプロジェクトを削除する必要があります。

## 次のステップ

ナレッジストアは、エンリッチメントされたドキュメントを永続化する手段として、スキルセットを設計する際に役立つほか、Azure Storage アカウントにアクセスする機能を備えた、あらゆるクライアント アプリケーションから利用する新しい構造やコンテンツを作成する際にも役立てることができます。

エンリッチメントされたドキュメントを作成する最も簡単なアプローチは、[ポータルを使用することですが](#)、Postman と REST API を使用する方法もあります。プログラムでオブジェクトがどのように作成され、参照されるのかについて分析情報が必要な場合には、後者の方が便利です。

[Postman と REST を使用してナレッジストアを作成する](#)

# Azure AI Search でのデータインポート

[アーティクル] • 2024/01/19

Azure AI Search では、クエリは、[検索インデックス](#)に読み込まれたユーザー所有のコンテンツに対して実行されます。この記事では、インデックスを作成する 2 つの基本的なワークフローについて説明します。プログラムでデータをインデックスにプッシュするワークフローと、[検索インデクサー](#)を使用してデータをプルするワークフローです。

どちらの方法でも、外部データ ソースからドキュメントが読み込まれます。空のインデックスを作成することもできますが、コンテンツを追加するまでクエリは実行できません。

## ① 注意

AI エンリッチメントがソリューションの要件である場合は、プル モデル(インデクサー)を使用してインデックスを読み込む必要があります。スキルセットはインデクサーにアタッチされ、独立して実行されません。

## インデックスにデータをプッシュする

プッシュ モデルは、API を使用して既存の検索インデックスにドキュメントをアップロードするアプローチです。ドキュメントは、1 バッチあたり最大 1,000 個、またはバッチあたり 16 MB (メガバイト) (どちらか早い方) に個別に、またはバッチでアップロードできます。

主な利点:

- データ ソースの種類に制限はありません。ペイロードは、インデックス スキーマにマップされる JSON ドキュメントで構成されている必要がありますが、データはどこからでもソース化できます。
- 実行頻度に制限はありません。インデックスには、必要に応じて何度も変更をプッシュすることができます。待機時間の要件が低いアプリケーションの場合 (たとえば、インデックスを製品在庫の変動と同期させる必要がある場合など)、プッシュ モデルが唯一のオプションです。
- ドキュメントの接続性と安全な取得は、完全に制御下にあります。これに対し、インデクサー接続は、Azure AI Search で提供されるセキュリティフィーチャーを使用して認証されます。

# Azure AI Search インデックスにデータをプッシュする方法

1つまたは複数のドキュメントをインデックスに読み込むには、次の API を使用します。

- ドキュメントの追加、更新、削除 (REST API)
- `IndexDocumentsAsync` (Azure SDK for .NET) または `SearchIndexingBufferedSender`
- `IndexDocumentsBatch` (Azure SDK for Python) または `SearchIndexingBufferedSender`
- `IndexDocumentsBatch` (Azure SDK for Java) または `SearchIndexingBufferedSender`
- `IndexDocumentsBatch` (Azure SDK for JavaScript) または `SearchIndexingBufferedSender`

Azure portal を使用したデータのプッシュはサポートされていません。

プッシュ API の概要については、以下を参照してください。

- クイックスタート: Azure SDK を使用したフルテキスト検索
- C# チュートリアル: プッシュ API を使用してインデックス作成を最適化する
- REST クイック スタート: PowerShell を使用して Azure AI Search インデックスを作成する

## インデックス作成アクション: `upload`、`merge`、`mergeOrUpload`、`delete`

インデックス作成アクションの種類をドキュメントごとに制御できます。つまり、ドキュメントを全部アップロードするか、既存のドキュメントコンテンツとマージするか、または削除するかを指定できます。

REST API と Azure SDK のどちらを使用する場合でも、データのインポートでは次のドキュメント操作がサポートされます。

- ドキュメントが新しい場合は挿入され、存在する場合は更新または置き換えられる `"upsert"` と同様にアップロードします。インデックスに必要な値がドキュメントにない場合、ドキュメントフィールドの値は `null` に設定されます。
- マージでは、既に存在するドキュメントが更新され、見つからないドキュメントが失敗します。マージは既存の値を置き換えます。そのため、`Collection(Edm.String)` 型のフィールドなど、複数の値を含むコレクションフィールドは必ず確認してください。たとえば、`tags` フィールドの値が `["budget"]` で始まり、値 `["economy", "pool"]` でマージを実行した場合、`tags` フィールドの

最終値は `["economy", "pool"]` になります。 `["budget", "economy", "pool"]` にはなりません。

- `mergeOrUpload`。ドキュメントが存在する場合は `merge` と同様な動作をし、ドキュメントが新しい場合は `upload` の動作をします。
- `delete`。インデックスから指定したドキュメントを削除します。個々のフィールドを削除する場合は、代わりに `merge` を使い、問題のフィールドを `null` に設定します。

## インデックスへのデータのプル

プレモデルでは、サポートされているデータソースに接続するインデクサーが使用され、データがインデックスに自動的にアップロードされます。Microsoft のインデクサーは、次のプラットフォームで利用できます。

- [Azure BLOB Storage](#)
- [Azure Table Storage](#)
- [Azure Data Lake Storage Gen2](#)
- [Azure Files \(プレビュー\)](#)
- [Azure Cosmos DB](#)
- [Azure SQL Database、SQL Managed Instance、および Azure VM 上の SQL Server](#)
- [Microsoft 365 での SharePoint \(プレビュー\)](#)

Microsoft パートナーによって開発およびメインされたサードパーティ製コネクタを使用できます。詳細とリンクについては、「[データソースギャラリー](#)」を参照してください。

インデクサーは、インデックスをデータソース(通常はテーブル、ビュー、または同等の構造体)に接続し、ソースフィールドをインデックスの同等のフィールドにマップします。実行中、行セットが自動的に JSON に変換され、指定したインデックスに読み込まれます。すべてのインデクサーはスケジュールをサポートしているため、データの更新頻度を指定できます。ほとんどのインデクサーは、変更の追跡を提供します(データソースでサポートされている場合)。インデクサーは、新しいドキュメントを認識するだけでなく、既存のドキュメントの変更と削除を追跡するため、インデックス内のデータをアクティブに管理する必要がありません。

## Azure AI Search インデックスにデータをプルする方法

インデクサーベースのインデックス作成には、次のツールと API を使用します。

- [Azure portal のデータのインポート ウィザード](#)

- REST API: [インデクサーの作成 \(REST\)](#)、[データ ソースの作成 \(REST\)](#)、[インデックスの作成 \(REST\)](#)
- Azure SDK for .NET: [SearchIndexer](#)、[SearchIndexerDataSourceConnection](#)、[SearchIndex](#)、
- Azure SDK for Python: [SearchIndexer](#)、[SearchIndexerDataSourceConnection](#)、[SearchIndex](#)、
- Azure SDK for Java: [SearchIndexer](#)、[SearchIndexerDataSourceConnection](#)、[SearchIndex](#)、
- Azure SDK for JavaScript: [SearchIndexer](#)、[SearchIndexerDataSourceConnection](#)、[SearchIndex](#)、

インデクサー機能は、[Azure portal]、[REST API](#)、および[.NET SDK](#)で公開されています。

ポータルを使用する利点は、Azure AI Search では通常、ソース データセットのメタデータを読み取ることでデフォルトのインデックススキーマを生成できることです。

## Search エクスプローラーを使用してデータのインポートを検証する

ドキュメントのアップロード時に事前チェックを実行する簡単な方法は、ポータルで[Search エクスプローラー](#)を使用することです。



エクスプローラーを使用すると、コードを記述することなくインデックスを照会できます。検索エクスペリエンスは、既定の設定 ([単純構文](#)、既定の [searchMode クエリパラメーター](#)など) に基づきます。結果は JSON で返されるため、ドキュメント全体を確認できます。

JSON ビューの検索エクスプローラーで実行できるクエリの例を次に示します。  
"HotelId" は hotels-sample-index のドキュメントキーです。フィルターには特定のドキュメントのドキュメント ID を指定します。

```
JSON
{
  "search": "*",
  "filter": "HotelId eq '50'"
}
```

REST を使っている場合は、この[参照クエリ](#)で同じ目的を達成できます。

## 関連項目

- インデクサーの概要
- ポータルのクイックスタート: インデックスの作成、読み込み、クエリ

# Azure AI Search のインデクサー

[アーティクル] • 2023/12/18

*Azure AI Search のインデクサー*は、クラウド データ ソースからテキスト データを抽出し、ソース データと検索インデックスの間のフィールド間マッピングを使用して検索インデックスを設定するクローラーです。この方法は、インデックスにデータを追加するコードを記述することなく、検索サービスがデータをプルするため、「プル モデル」と呼ばれることもあります。

インデクサーはスキルセットの実行と AI エンリッチメントも推進します。このエンリッチメントでは、インデックスにルーティングされるコンテンツの追加処理を統合するようにスキルを構成できます。画像ファイルの OCR、データ チャンクのテキスト分割スキル、複数の言語のテキスト翻訳など、いくつかの例があります。

インデクサーは、サポートされているデータ ソースを対象とします。インデクサー構成では、データ ソース (配信元) と検索インデックス (宛先) を指定します。Azure Blob Storage など、いくつかのソースには、そのコンテンツの種類に固有の追加の構成プロパティがあります。

インデクサーは、オンデマンドで実行することも、5 分ごとに実行される定期的なデータ更新スケジュールで実行することもできます。より頻繁に更新するには、Azure AI Search と外部データ ソースの両方のデータを同時に更新する "プッシュ モデル" が必要です。

検索サービスは、検索ユニットごとに 1 つのインデクサー ジョブを実行します。同時処理が必要な場合は、十分なレプリカがあることを確認してください。インデクサーはバックグラウンドで実行されないため、サービスに負荷がかかっている場合は、通常よりも多くのクエリ調整が検出される可能性があります。

## インデクサーのシナリオとユース ケース

インデクサーは、データ インジェストの唯一の手段として、または他の手法と組み合わせて使用できます。次の表に主なシナリオをまとめています。

[+] テーブルを展開する

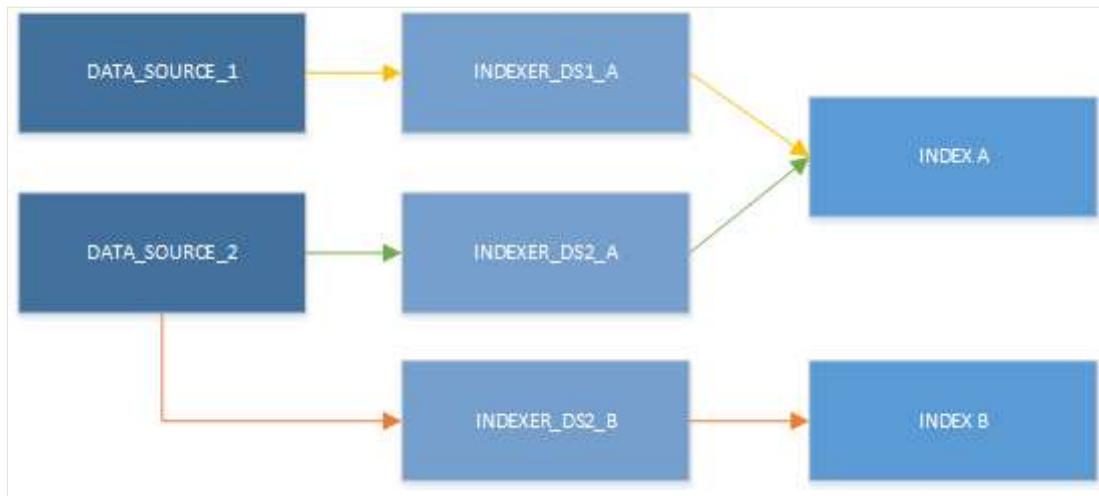
シナリオ	戦略
単一データ	このパターンは最も単純です。1 つのデータ ソースが検索インデックス用の唯一のコンテンツ プロバイダーです。サポートされているほとんどのデータ ソースで何らかの形式

シナリオ	戦略
ソース	の変更が検出されるため、ソースでコンテンツが追加または更新されたときに、後続のインデクサーの実行によって差分が取得されます。
複数のデータソース	インデクサーの指定で使用できるデータソースは1つだけですが、検索インデックス全体は複数のソースからのコンテンツを受け入れることができます。この場合、各インデクサーの実行によって、別のデータ プロバイダーから新しいコンテンツが取り込まれます。各ソースは、完全なドキュメントの共有を投稿したり、各ドキュメントで選択したフィールドを設定したりできます。このシナリオの詳細については、 <a href="#">複数のデータ ソースからのインデックスのチュートリアル</a> を参照してください。
複数のインデクサー	実行時のパラメーター、スケジュール、またはフィールド マッピングを変更する必要がある場合、複数のデータ ソースは通常、複数のインデクサーとペアになります。
—	<a href="#">Azure AI Search のクロスリージョンのスケールアウト</a> がもう1つのシナリオです。同じ検索インデックスのコピーが異なるリージョンに存在する場合があります。検索インデックスのコンテンツを同期するには、同じデータ ソースからプルする複数のインデクサーを作成できます。この場合、各インデクサーのターゲットはリージョンごとに異なる検索インデックスです。非常に大きなデータセットの並列インデックスでも、各インデクサーがデータのサブセットをターゲットとするマルチインデクサー戦略が必要です。
コンテンツの変換	インデクサーは、スキルセットの実行と AI エンリッチメントを推進します。コンテンツの変換は、インデクサーにアタッチする <a href="#">スキルセット</a> で定義されます。スキルを使用して、 <a href="#">データ チャンクとベクター化</a> を組み込むことができます。

ターゲット インデックスとデータ ソースの組み合わせごとにインデクサーを1つ作成するように設計する必要があります。複数のインデクサーが同じインデックスに書き込みできます。複数のインデクサーに同じデータ ソースを再利用できます。ただし、インデクサーが1回に利用できるデータ ソースは1つだけです。そして、書き込めるインデックスは1つだけです。次の図に示すように、1つのデータ ソースが1つのインデクサーに入力を提供し、1つのインデックスを設定します:



一度に使用できるインデクサーは1つだけですが、リソースはさまざまな組み合わせで使用できます。次の図の主なポイントは、データ ソースを複数のインデクサーと組み合わせることができ、複数のインデクサーが同じインデックスに書き込むことができる点です。



## サポートされるデータ ソース

インデクサーは、Azure および Azure 外部でデータストアをクロールします。

- [Azure Blob Storage](#)
- [Azure Cosmos DB](#)
- [Azure Data Lake Storage Gen2](#)
- [Azure SQL Database](#)
- [Azure Table Storage](#)
- [Azure SQL Managed Instance](#)
- [Azure Virtual Machines における SQL Server](#)
- [Azure Files \(プレビューフェーズ\)](#)
- [Azure MySQL \(プレビューフェーズ\)](#)
- [Microsoft 365 での SharePoint \(プレビューフェーズ\)](#)
- [Azure Cosmos DB for MongoDB \(プレビューフェーズ\)](#)
- [Azure Cosmos DB for Apache Gremlin \(プレビューフェーズ\)](#)

Azure Cosmos DB for Cassandra はサポートされていません。

インデクサーは、テーブルやビューなどのフラット化された行セット、またはコンテナーまたはフォルダー内の項目を受け入れます。ほとんどの場合、行、レコード、または項目ごとに 1 つの検索ドキュメントが作成されます。

共有プライベートリンクを使用する場合、リモートデータソースへのインデクサー接続は、標準のインターネット接続 (パブリック) または暗号化されたプライベート接続を使用して行うことができます。また、マネージド ID を使用して認証を行うように、接続を設定することもできます。セキュリティで保護された接続の詳細については、[Azure ネットワークセキュリティ機能によって保護されたコンテンツへのインデクサーのアクセスと、マネージド ID を使用したデータソースへの接続に関するページ](#)を参照してください。

# インデックス作成のステージ

最初の実行時に、インデックスが空の場合、テーブルまたはコンテナーで提供されるすべてのデータがインデクサーによって読み取られます。その後の実行では、通常、変更されたデータのみがインデクサーによって検出され取得されます。BLOB データの場合、変更の検出は自動で行われます。Azure SQL や Azure Cosmos DB などの他のデータソースで、変更の検出を有効にする必要があります。

受信したドキュメントごとに、インデクサーによって、ドキュメントの取得からインデックス付けのための最終的な検索エンジンの "ハンドオフ" までの、複数のステップが実装または調整されます。また、インデクサーを使用すると、スキルセットが定義されている場合に、[スキルセットの実行と出力](#)も促進されます。



## ステージ 1: ドキュメント解析

ドキュメント解析は、ファイルを開いてコンテンツを抽出するプロセスです。テキストベースのコンテンツは、サービスのファイル、テーブルの行、またはコンテナーやコレクションの項目から抽出できます。スキルセットと[画像スキル](#)を追加した場合、ドキュメント解析で画像を抽出し、画像処理のためにキューに登録することもできます。

データソースに応じて、インデックス付けが可能なコンテンツを抽出するために、インデクサーによってさまざまな操作が試行されます。

- ドキュメントが PDF などの画像が埋め込まれたファイルである場合、インデクサーはテキスト、画像、メタデータを抽出します。インデクサーは、[Azure Blob Storage](#)、[Azure Data Lake Storage Gen2](#)、[SharePoint](#) からファイルを開くことができます。
- ドキュメントが [Azure SQL](#) のレコードの場合は、インデクサーによって各レコードの各フィールドからバイナリ以外のコンテンツが抽出されます。
- ドキュメントが [Azure Cosmos DB](#) 内のレコードの場合は、インデクサーによって Azure Cosmos DB ドキュメントのフィールドとサブフィールドからバイナリ以外のコンテンツが抽出されます。

## ステージ 2: フィールド マッピング

インデクサーによって、ソース フィールドからテキストが抽出され、インデックスまたはナレッジ ストアの送信先フィールドにそれが送信されます。フィールド名とデータ型が一致すると、パスは明確になります。ただし、出力には異なる名前または型が必要な場合があります。その場合は、フィールドをマップする方法をインデクサーに指示する必要があります。

[フィールド マッピングを指定する](#)には、インデクサー定義に、ソース フィールドと宛先フィールドを入力します。

フィールド マッピングは、ドキュメント解析の後、変換前に、インデクサーがソース ドキュメントから読み取るときに行われます。フィールド マッピングを定義するときに、ソース フィールドの値は変更されずにそのまま送信先フィールドに送信されます。

## ステージ 3: スキルセットの実行

スキルセットの実行は、組み込みまたはカスタムの AI 処理を呼び出す省略可能なステップです。スキルセットでは、コンテンツがバイナリの場合、光学式文字認識 (OCR) またはその他の形式の画像分析を追加できます。スキルセットで自然言語処理を追加することもできます。たとえば、テキスト翻訳やキー フレーズ抽出を追加できます。

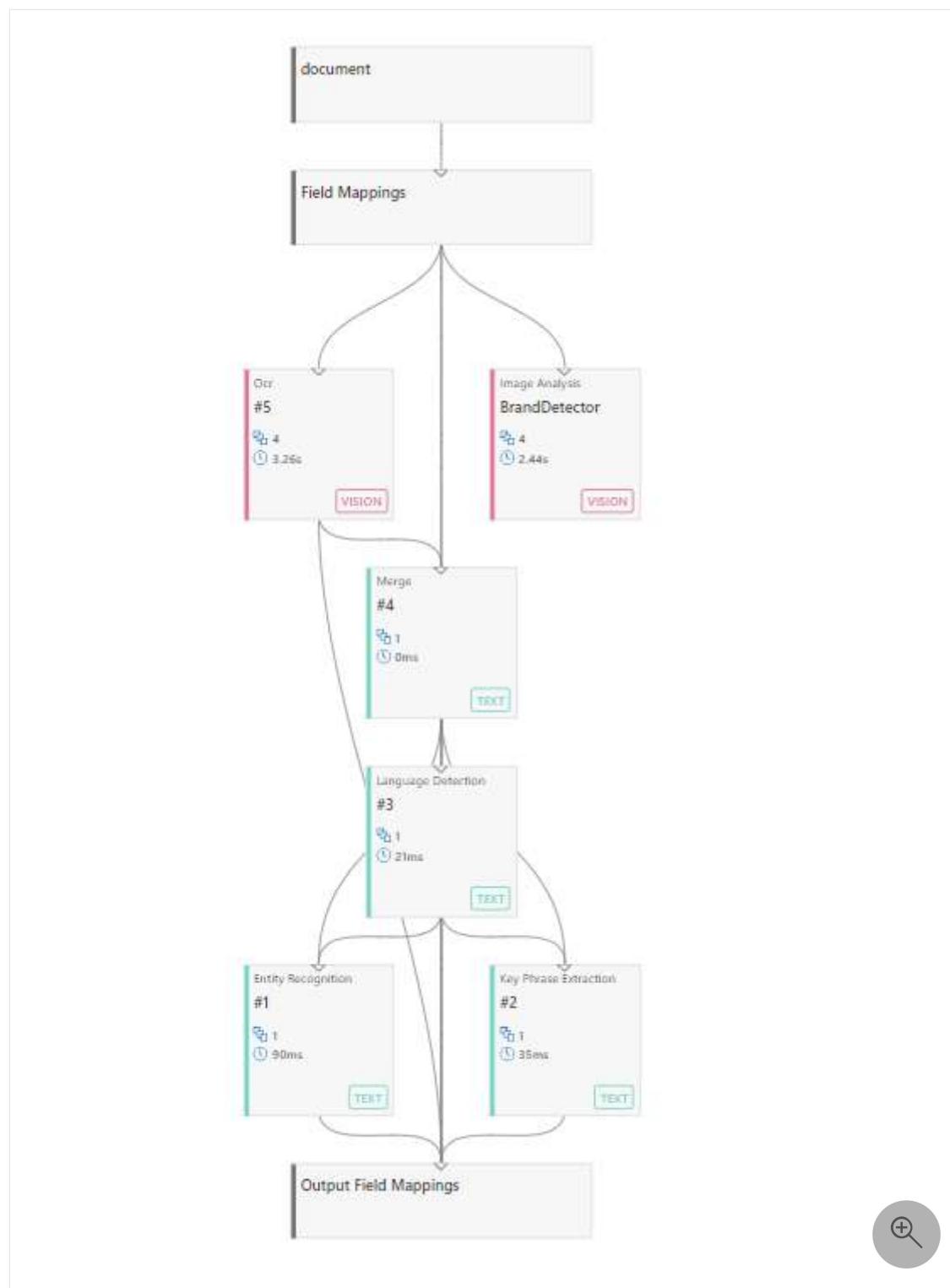
変換が何であれ、スキルセットの実行は、エンリッチメントが発生する場所です。インデクサーがパイプラインの場合、[スキルセット](#)を "パイプライン内のパイプライン" として考えることができます。

## ステージ 4: 出力フィールドマッピング

スキルセットを含める場合は、インデクサー定義で[出力フィールド マッピングを指定する](#)必要があります。スキルセットの出力は、エンリッチされたドキュメントと呼ばれるツリー構造として内部的に示されます。出力フィールド マッピングを使用すると、このツリーの部分を選択してインデックス内のフィールドにマップすることができます。

名前が類似しているにもかかわらず、出力フィールド マッピングとフィールド マッピングは、異なるソースから関連付けを構築します。フィールド マッピングでは、ソース フィールドの内容を検索インデックスの宛先フィールドに関連付けます。出力フィールド マッピングでは、内部のエンリッチされたドキュメント (スキル出力) の内容をインデックス内の宛先フィールドに関連付けます。省略可能と見なされるフィールド マッピングとは異なり、インデックスに存在する必要がある変換されたすべてのコンテンツには、出力フィールド マッピングが必要になります。

次の図は、インデクサーのステージ(ドキュメント解析、フィールドマッピング、スキルセットの実行、出力フィールドマッピング)のサンプルインデクサーのデバッグセッション表現を示しています。



## の基本的なワークフロー

インデクサーで実行できる機能は、データソースごとに異なります。そのためインデクサーとデータソースの構成には、インデクサーの種類ごとに異なる点があります。しかし基本的な成り立ちと要件は、すべてのインデクサーに共通です。以降、すべてのインデクサーに共通の手順について取り上げます。

## 手順 1: データ ソースを作成する

インデクサーには、接続文字列と必要に応じて資格情報を提供する、"データ ソース" オブジェクトが必要です。データ ソースは独立したオブジェクトです。複数のインデクサーは、同じデータ ソース オブジェクトを使用して、一度に複数のインデックスを読み込むことができます。

次のいずれかの方法を使用してデータ ソースを作成できます。

- Azure portal を使用し、検索サービス ページの [データ ソース] タブで [データ ソースの追加] を選択して、データ ソースの定義を指定します。
- Azure portal を使用して、[データのインポート] ウィザードでデータ ソースを出力します。
- REST API を使用して、[データ ソースの作成] を呼び出します。
- Azure SDK for .NET を使用して、`SearchIndexerDataSourceConnection` クラスを呼び出します

## 手順 2: インデックスを作成する

インデクサーは、データ インジェストに関連したいくつかのタスクを自動化しますが通常、そこにはインデックスの作成は含まれていません。前提条件として、お使いの外部データ ソース内のすべてのソース フィールドの対応するターゲット フィールドが含まれた定義済みインデックスが必要になります。各フィールドでは、名前とデータ型が一致する必要があります。そうでない場合は、[フィールド マッピングを定義](#)して関連付けを確立できます。

詳細については、「[インデックスの作成](#)」を参照してください。

## 手順 3: インデクサーを作成して実行 (またはスケジュール) する

インデクサーの定義は、インデクサーを一意に識別するプロパティ、使用するデータ ソースとインデックスを指定するプロパティ、実行時の動作に影響する他の構成オプションを提供するプロパティ (インデクサーをオンデマンドで実行するか、スケジュールに基づき実行するかなど) で構成されます。

データ アクセスまたはスキルセットの検証に関するエラーまたは警告は、インデクサーの実行中に発生します。インデクサーの実行が開始されるまで、データ ソース、インデックス、スキルセットなどの依存オブジェクトは、検索サービスでパッシブになります。

詳細については、「[インデクサーの作成](#)」を参照してください

インデクサーの初回実行の後、[オンデマンドで再実行](#)することや、[スケジュールを設定](#)することができます。

インデクサーの状態は、[ポータル](#)か [Get Indexer Status API](#) を使用して監視できます。また、[インデックスのクエリを実行](#)し、期待した結果が得られるかどうかを確認する必要があります。

インデクサーには専用の処理リソースがありません。これに基づき、インデクサーの状態が(キュー内の他のジョブに応じて)実行される前にアイドル状態として表示され、実行時間が予測できない場合があります。インデクサーのパフォーマンスは、ドキュメントサイズ、ドキュメントの複雑さ、画像分析などのその他の要因によっても定義されます。

## 次のステップ

インデクサーの概要がわかったので、次のステップは、インデクサーのプロパティとパラメーター、スケジュール、およびインデクサーの監視について確認することです。また、[サポートされているデータソース](#)の一覧に戻り、特定のソースの詳細を確認することもできます。

- [インデクサーを作成する](#)
- [インデクサーのリセットと実行](#)
- [インデクサーをスケジュールする](#)
- [フィールドマッピングを定義する](#)
- [インデクサーの状態を監視する](#)

# Azure AI Search での AI エンリッチメント

[アーティクル] • 2024/01/30

Azure AI Search では、AI エンリッチメントとは、未加工の形式で検索できないコンテンツを処理するための Azure AI サービスとの統合を指します。エンリッチメントにより、分析と推論を使用して、以前は存在しなかった検索可能なコンテンツや構造を作成します。

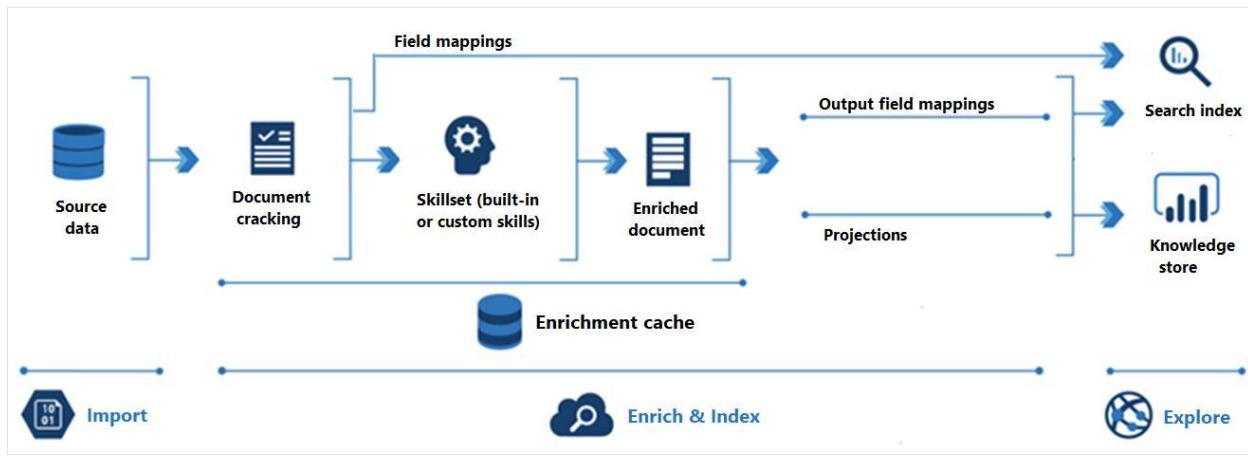
Azure AI Search はテキストおよびベクター検索ソリューションであるため、AI エンリッチメントの目的は、検索関連のシナリオでコンテンツの有用性を向上することです。ソース コンテンツはテキスト形式である必要がありますが（ベクターをエンリッチすることはできません）、エンリッチメント パイプラインによって作成されたコンテンツは、チャネル用の Text Split スキルやエンコード用の AzureOpenAiEmbedding スキルなどのスキルを使用してベクター ストアでベクター化およびインデックス付けできます。

組み込みのスキルは、生のコンテンツに次の変換と処理を適用します。

- 多言語検索の場合の翻訳と言語検出
- テキストの大きなチャネルからユーザー名、場所、およびその他のエンティティを抽出するエンティティ認識
- 重要な用語を識別して出力するためのキー フレーズ抽出
- バイナリ ファイル内の印刷されたテキストと手書きのテキストを認識する光学式文字認識 (OCR)
- 画像の内容を説明し、説明を検索可能なテキスト フィールドとして出力する画像分析

AI エンリッチメントは、Azure データ ソースに接続するインデクサー パイプラインの拡張機能です。エンリッチメント パイプラインには、インデクサー パイプラインのすべてのコンポーネント（インデクサー、データ ソース、インデックス）に加えて、アトミック エンリッチメント ステップを指定するスキルセットが含まれます。

次の図に、AI エンリッチメントの進行を示します。



**インポート**は最初の手順です。ここで、インデクサーがデータソースに接続し、コンテンツ(ドキュメント)を検索サービスにプルします。AIエンリッチメントシナリオで使用される最も一般的なリソースは [Azure Blob Storage](#) ですが、サポートされている任意のデータソースがコンテンツを提供できます。

**エンリッチ&インデックス**は、ほとんどのAIエンリッチメントパイプラインを対象とします。

- エンリッチメントは、インデクサーが "[ドキュメントを解読](#)" し、画像とテキストを抽出したときに開始されます。次に発生する処理の種類は、データと、スキルセットに追加したスキルによって異なります。画像がある場合は、画像処理を実行するスキルに転送できます。テキストコンテンツは、テキストと自然言語の処理のためにキューに入れられます。内部的には、スキルによって、変換が発生したときにそれを収集する "[エンリッチされたドキュメント](#)" が作成されます。
- エンリッチ済みコンテンツは、スキルセットの実行の間に生成され、ユーザーが保存しない限り一時的なものです。後でスキルセットを実行するときに再利用するため、[エンリッチメントキャッシュ](#)で解読されたドキュメントとスキル出力を保持できます。
- 検索インデックスにコンテンツを取り込むため、インデクサーはエンリッチされたコンテンツをターゲットフィールドに送信するためのマッピング情報を保持している必要があります。[フィールドマッピング](#)(明示的または暗黙的)は、ソースデータから検索インデックスへのデータパスを設定します。[出力フィールドマッピング](#)は、エンリッチされたドキュメントからインデックスへのデータパスを設定します。
- インデックス作成は、生のコンテンツとエンリッチされたコンテンツが[検索インデックス](#)(そのファイルとフォルダー)の物理データ構造に取り込まれるプロセスです。このステップで、字句解析とトーカン化が行われます。

**探索**は最後の手順です。出力は常に、クライアントアプリからクエリを実行できる[検索インデックス](#)です。必要に応じて出力を、データ探索ツールまたはダウンストリーム

ムプロセスを介してアクセスされる Azure Storage 内の BLOB とテーブルで構成されるナレッジストアにすることができます。ナレッジストアを作成している場合は、エンリッチされたコンテンツのデータパスが[プロジェクト](#)によって決定されます。インデックスとナレッジストアの両方に、同じエンリッチされたコンテンツを含めることができます。

## どのような場合に AI エンリッチメントを使用するか

生コンテンツが、非構造化テキスト、画像コンテンツ、または言語検出と翻訳を必要とするコンテンツの場合は、エンリッチメントが有用です。"組み込みのコグニティブスキル" を通じて AI を適用すると、フルテキスト検索とデータサイエンスアプリケーションに対してこのコンテンツのロックが解除されます。

[カスタムスキル](#)を作成して、外部処理を提供することもできます。オープンソース、サードパーティ、またはファーストパーティのコードは、カスタムスキルとしてパイプラインに統合できます。さまざまなドキュメントの種類の顕著な特徴を識別する分類モデルはこのカテゴリに分類されますが、コンテンツの価値を高める任意の外部パッケージを使用できます。

## 組み込みスキルのユースケース

組み込みのスキルは、[Azure AI Computer Vision](#) と [Language Service](#) などの Azure AI サービス API に基づいています。コンテンツの入力が少ない場合を除いて、より大きなワークフローを実行するために、[課金対象の Azure AI サービス リソースをアタッチする必要があります。](#)

組み込みのスキルを使用して作られた[スキルセット](#)は、次のアプリケーションシナリオに適しています。

- **画像処理**スキルには、[光学式文字認識 \(OCR\)](#) と、[視覚的特徴](#)の識別が含まれます。後者は、顔検出、画像の解釈、画像の認識(有名な人物やランドマーク)、画像の向きのような属性などの識別です。これらのスキルにより、Azure AI Search でフルテキスト検索用の画像コンテンツのテキスト表現が作成されます。
- **機械翻訳**は、多くの場合、多言語ソリューション用の[言語検出](#)と組み合わせて、[テキスト翻訳](#)スキルによって提供されます。
- **自然言語処理**では、テキストのチャンクが分析されます。このカテゴリのスキルには、[エンティティ認識](#)、[センチメント検出 \(オピニオンマイニングを含む\)](#)、[個人を特定できる情報の検出](#)などがあります。これらのスキルによって、構造化さ

れていないテキストが、インデックスで検索可能およびフィルター可能なフィールドとしてマップされます。

## カスタム スキルのユース ケース

**カスタム スキル**は、指定した外部コードを実行し、**カスタム スキル Web インターフェイス**でラップします。カスタム スキルのいくつかの例は、[azure-search-power-skills](#) の GitHub リポジトリにあります。

カスタム スキルは常に複雑とは限りません。たとえば、パターン マッチングまたはドキュメント分類モデルを提供する既存のパッケージがある場合は、カスタム スキルで完成させることができます。

## 出力を格納する

Azure AI Search では、インデクサーによって作成された出力が保存されます。1回のインデクサー実行で、エンリッチされてインデックス付けされた出力を含む最大3つのデータ構造を作成できます。

[+] [テーブルを展開する](#)

データ ストア	必 須	場所	説明
検索可 能なイ ンデッ クス	必 須	検索サ ービス	フルテキスト検索やその他のクエリ フォームに使用されます。インデックスの指定はインデクサーの要件です。インデックスの内容は、スキルの出力に加えて、インデックス内のフィールドに直接マップされるすべてのソース フィールドから設定されます。
ナレッ ジス トア	オ プ シ ヨ ン	Azure Storage	ナレッジ マイニングやデータ サイエンスなどのダウンストリーム アプリに使用されます。ナレッジストアは、スキルセット内で定義されています。その定義により、エンリッチされたドキュメントが Azure Storage でテーブルとオブジェクト (ファイルと BLOB) のどちらとして投影されるかが決まります。
エンリ ッチメ ント キャッ シュ:	オ プ シ ヨ ン	Azure Storage	後続のスキルセットの実行で再利用するためのエンリッチメントのキャッシュに使用されます。キャッシュには、インポートされた未処理のコンテンツ (解読されたドキュメント) が格納されます。スキルセットの実行中に作成されたエンリッチされたドキュメントも格納されます。キャッシュは、画像解析または OCR を使用していて、画像ファイルの再処理にかかる時間と費用を回避したい場合に役立ちます。

インデックスとナレッジストアは相互に完全に独立しています。インデクサーの要件を満たすにはインデックスをアタッチする必要がありますが、ナレッジストアだけが目的の場合は、作成された後でインデックスを無視してかまいません。

# コンテンツを探索する

検索インデックスまたはナレッジストアを定義して読み込んだら、そのデータを探索できます。

## 検索インデックスに対してクエリを実行する

クエリを実行して、パイプラインによって生成されたエンリッチされたコンテンツにアクセスします。 インデックスは、Azure AI Search 用に作成する他のインデックスと同様です。 カスタム アナライザーを使用してテキスト解析を補完したり、あいまい検索クエリを呼び出したり、フィルターを追加したり、スコアリング プロファイルを実験して検索の関連性を調整したりできます。

## ナレッジストアに対してデータ探索ツールを使用する

この Azure Storage ナレッジストア では、JSON ドキュメントの BLOB コンテナー、イメージオブジェクトの BLOB コンテナー、または Table Storage のテーブルの形式を想定できます。 [Storage Explorer](#)、[Power BI](#)、または Azure Storage に接続する任意のアプリを使用して、コンテンツにアクセスできます。

- BLOB コンテナーは、エンリッチされたドキュメント全体をキャプチャします。これは、他のプロセスにフィードを作成する場合に便利です。
- テーブルは、エンリッチされたドキュメントのスライスが必要な場合、または出力の特定の部分を含めるか除外する場合に便利です。 Power BI での分析には、表がデータの探索や可視化に推奨されるデータソースです。

## 可用性と料金

エンリッチメントは、Azure AI サービスが利用できるリージョンで利用できます。 エンリッチメントをご利用いただけるかどうかは、「[リージョン別の利用可能な Azure 製品](#)」ページをご確認いただけます。

課金は、従量課金制モデルに従います。 スキルセットで複数リージョンの Azure AI サービスキーが指定されている場合、組み込みスキルを使用するためのコストが転嫁されます。 Azure AI Search によって測定される画像抽出に関するコストもあります。ただし、テキスト抽出とユーティリティのスキルは請求されません。 詳細については、「[Azure AI Search の課金方法](#)」を参照してください。

## 一般的なワークフローのチェックリスト

エンリッチメントパイプラインは、"スキルセット" を含む "インデクサー" で構成されます。インデックス作成の後で、インデックスのクエリを実行して結果を検証できます。

サポートされるデータソース内のデータのサブセットから始めます。インデクサーとスキルセットの設計は、反復的なプロセスです。代表的なデータセットが小さいと作業が速くなります。

1. データへの接続が指定されているデータソースを作成します。
2. スキルセットを作成します。プロジェクトが小さい場合を除き、Azure AI マルチサービスリソースをアタッチする必要があります。ナレッジストアを作成する場合は、スキルセット内でそれを定義します。
3. 検索インデックスを定義するインデックススキーマを作成します。
4. インデクサーを作成して実行し、上記のすべてのコンポーネントをまとめます。この手順では、データの取得、スキルセットの実行、インデックスの読み込みを行います。

インデクサーでは、検索インデックスへのデータパスを設定するフィールドマッピングと出力フィールドマッピングも指定します。

必要に応じて、インデクサーの構成でエンリッチメントキャッシュを有効にします。この手順により、後で既存のエンリッチメントを再利用できるようになります。

5. クエリを実行して結果を評価するか、デバッグセッションを開始してスキルセットの問題を解決します。

上記の手順を繰り返す場合は、インデクサーを実行する前にリセットします。または、実行ごとにオブジェクトを削除して再作成します (Free レベルを使用している場合は推奨)。インデクサーのキャッシュを有効にした場合、ソースでデータが変更されていない場合、およびパイプラインに対する編集によってキャッシュが無効にされない場合は、インデクサーがキャッシュからプルされます。

## 次のステップ

- クイックスタート: AI エンリッチメントのスキルセットを作成する
- チュートリアル: AI エンリッチメント REST API について学習する
- スキルセットの概念
- ナレッジストアの概念
- スキルセットを作成する
- ナレッジストアを作成する

# Azure AI Search での増分 エンリッチメントとキャッシュ

[アーティクル] • 2024/02/18

## ① 重要

この機能はパブリック プレビュー段階にあり、[追加使用条件](#) の下で提供されます。この機能は、[プレビュー REST API](#) でサポートされます。

"インクリメンタル エンリッチメント" とは、[スキルセットの実行](#)中にキャッシュされたエンリッチメントを使うことを指します。こうすることで、新規および変更されたスキルとドキュメントにのみ、Azure AI サービスへの API 呼び出しの従量課金処理料金が発生します。キャッシュには、[ドキュメント解析](#)の出力に加え、ドキュメントごとの各スキルの出力が格納されます。キャッシュは課金対象 (Azure Storage を使用) です。ただし、ストレージのコストは画像抽出や AI 処理よりも低いため、強化全体コストは削減されます。

キャッシュを有効にすると、インデクサーによって更新内容が評価され、既にあるエンリッチメントをキャッシュから取得できるかどうかが判断されます。ドキュメント解析フェーズから出力される画像やテキスト コンテンツに加え、編集の上流のスキル出力や、編集に対して直交するスキル出力は再利用できる可能性が高くなります。

スキルセットの処理が完了した後、更新された結果は、キャッシュだけでなく、検索インデックスまたはナレッジ ストアにも書き戻されます。

## 制限事項

### ⊗ 注意事項

[SharePoint Online インデクサー \(プレビュー\)](#) を使用している場合は、増分エンリッチメントを避ける必要があります。特定の状況では、キャッシュが無効になり、キャッシュを再ロードする場合は、[インデクサーをリセットして実行](#)する必要があります。

## キャッシュの構成

物理的には、ご使用の Azure ストレージ アカウントの BLOB コンテナーに、インデクサーごとに 1 つキャッシュが格納されます。 それぞれのインデクサーには、使っているコンテナーに対応する一意で不变のキャッシュ識別子が割り当てられます。

キャッシュは、"cache" プロパティを指定してインデクサーを実行すると作成されます。 キャッシュできるのは、エンリッチされたコンテンツのみです。 インデクサーにスキルセットがアタッチされていない場合、キャッシュは適用されません。

次の例は、キャッシュが有効になっているインデクサーを示しています。 詳細な手順については、[エンリッチメント キャッシュの有効化](#)に関するページを参照してください。 cache プロパティを追加するときの要求では、[プレビュー API バージョン \(2020-06-30-Preview 以降\)](#) を使うことに注意してください。

#### JSON

```
POST https://[search service name].search.windows.net/indexers?api-version=2020-06-30-Preview
{
    "name": "myIndexerName",
    "targetIndexName": "myIndex",
    "dataSourceName": "myDatasource",
    "skillsetName": "mySkillset",
    "cache": {
        "storageConnectionString": "<Your storage account connection string>",
        "enableReprocessing": true
    },
    "fieldMappings": [],
    "outputFieldMappings": [],
    "parameters": []
}
```

## キャッシュ管理

キャッシュのライフサイクルは、インデクサーによって管理されます。 インデクサーが削除されると、そのキャッシュも削除されます。 インデクサーの `cache` プロパティが `null` に設定されるか、接続文字列が変更された場合、既存のキャッシュはインデクサーの次回の実行時に削除されます。

インクリメンタル エンリッチメントはユーザーの介入なしに変更を検出して対応するように設計されています。 その一方で、特定の動作を呼び出すために使用できるパラメーターが用意されています。

- 新しいドキュメントを優先する
- スキルセット チェックをバイパスする

- データ ソース チェックをバイパスする
- スキルセットの評価を強制的に実行する

## 新しいドキュメントを優先する

cache プロパティには、`enableReprocessing` パラメータが含まれます。キャッシュ内で既に表現されている受信ドキュメントの処理を制御する目的で使用されます。true (既定値) の場合、インデクサーを再実行すると、キャッシュ内に既にあるドキュメントが再処理されます。これは、スキルの更新によってそのドキュメントが影響を受けると仮定されるためです。

false の場合、既存のドキュメントは再処理されず、実質的に新しい受信コンテンツが既存のコンテンツより優先されます。`enableReprocessing` を false に設定するのは、あくまで一時的な措置としてください。ほとんどの場合は `enableReprocessing` を true に設定しておくと、新規と既存両方のすべてのドキュメントが、最新のスキルセット定義に従って確実に有効になります。

## スキルセットの評価をバイパスする

通常、スキルの変更とそのスキルの再処理は連動します。ただし、スキルを変更しても再処理が行われてはならない場合がいくつかあります (たとえば、新しい場所に、または新しいアクセス キーを使って、カスタム スキルをデプロイする場合)。ほとんどの場合、これらは、スキル出力の実体そのものに実際の影響を与えない末梢的な変更です。

スキルに対する変更が実際は表面的なものであることがわかっている場合は、`disableCacheReprocessingChangeDetection` パラメータを true に設定して、スキルの評価をオーバーライドする必要があります。

1. [スキルセットの更新](#)を呼び出し、スキルセット定義を変更します。
2. 要求に "disableCacheReprocessingChangeDetection=true" パラメーターを追加します。
3. 変更を送信します。

このパラメーターを設定すると、スキルセット定義に対する更新だけがコミットされ、変更が既存のキャッシュに及ぼす影響は評価されません。プレビューの API バージョンである 2020-06-30-Preview 以降を使用してください。

### HTTP

```
PUT https://[servicename].search.windows.net/skillsets/[skillset name]?api-version=2020-06-30-Preview&disableCacheReprocessingChangeDetection
```

## データ ソースの検証チェックをバイパスする

ほとんどの場合、データ ソース定義を変更すると、キャッシングが無効になります。ただし、接続文字列の変更やストレージ アカウントのキーのローテーションなど、変更によってキャッシングが無効になってはならないことがわかっているシナリオでは、[データ ソースの更新](#)で `ignoreResetRequirement` パラメータを追加します。このパラメーターを `true` に設定すると、リセット条件 (結果的にすべてのオブジェクトが最初から再構築されて設定される条件) をトリガーすることなく、コミットを実行できます。

HTTP

```
PUT https://[search service].search.windows.net/datasources/[data source name]?api-version=2020-06-30-Preview&ignoreResetRequirement
```

## スキルセットの評価を強制的に実行する

キャッシングの目的は不必要的処理を回避することにあります。ここで、インデクサーによって検出されない変更 (たとえば、カスタム スキルなどの外部コード内の変更) をスキルに加えるケースを考えてみましょう。

この場合は、[スキルのリセット](#)を使用して、特定のスキル (そのスキルの出力に依存するダウンストリームのスキルも含まれます) を強制的に再処理することができます。この API は、無効にして再処理用にマークする必要があるスキルのリストが含まれた POST 要求を受け取ります。スキルのリセット後、[インデクサー実行](#)要求を行ってパイプライン処理を呼び出します。

## 特定のドキュメントを再キャッシングする

[インデクサーのリセット](#)を使用すると、検索コープス内のすべてのドキュメントが再処理されます。再処理を必要とするドキュメントがごく少数のシナリオでは、[ドキュメントのリセット \(レビュー\)](#)を使用して、特定のドキュメントを強制的に再処理します。ドキュメントがリセットされると、インデクサーによってそのドキュメントのキャッシングが無効にされ、ドキュメントはその後データ ソースから読み取ることによって再処理されます。詳細については、[インデクサー、スキル、ドキュメントの実行またはリセット](#)に関するページを参照してください。

特定のドキュメントをリセットする場合は、検索インデックスから読み取るドキュメントキーのリストを要求で指定します。キーが外部データソースのフィールドにマップ

されている場合、指定する値は検索インデックスで使用されているものであることが必要です。

API の呼び出し方法に応じて、要求ではキー リストを追加、上書き、またはキューに登録します。

- 異なるキーを使用して API を複数回呼び出すと、ドキュメントキーのリセットの一覧に新しいキーが追加されます。
- "overwrite" クエリ文字列パラメーターを true に設定して API を呼び出すと、リセットされるドキュメントキーの現在のリストが要求のペイロードで上書きされます。
- API を呼び出しても、ドキュメントキーはインデクサーによって実行される処理のキューに追加されるだけです。スケジュールに従ってまたはオンデマンドでインデクサーが次に呼び出されると、データ ソースからの他の変更よりも、リセットされたドキュメントキーの処理が優先されます。

次の例は、ドキュメントのリセット要求を示しています。

HTTP

```
POST https://[search service name].search.windows.net/indexers/[indexer
name]/resetdocs?api-version=2020-06-30-Preview
{
    "documentKeys" : [
        "key1",
        "key2",
        "key3"
    ]
}
```

## キャッシュが無効になる変更

キャッシュを有効にすると、インデクサーによってパイプライン構成の変更が評価され、再利用できるコンテンツと再処理が必要なコンテンツが特定されます。このセクションでは、キャッシュが完全に無効になる変更を示した後、増分処理がトリガーされる変更を示します。

無効化につながる変更とは、キャッシュ全体の有効性が失われる変更をいいます。たとえばデータ ソースの更新は、無効化につながる変更です。次に示すのは、インデクサー パイプラインのいずれかの部分に対する変更で、キャッシュが無効になるすべてのものの一覧です。

- データ ソースの種類の変更

- データ ソース コンテナーの変更
- データ ソースの資格情報の変更
- データ ソースの変更検出ポリシーの変更
- データ ソースの削除検出ポリシーの変更
- インデクサーのフィールドのマッピングの変更
- インデクサーのパラメーターの変更:
  - 解析モード
  - ファイル名拡張子を除外
  - ファイル名拡張子のインデックスを作成
  - サイズの大きいドキュメントのストレージ メタデータのみのインデックスを作成
  - 区切りテキストのヘッダー
  - 区切りテキストの区切り記号
  - ドキュメントのルート
  - 画像操作 (画像の抽出方法に対する変更)

## 増分処理がトリガーされる変更

増分処理では、対象のスキルセット定義が評価された後、再実行する必要があるスキルが特定され、ドキュメントツリーの影響を受ける部分が選択的に更新されます。結果的にインクリメンタル エンリッチメントが発生する変更の完全な一覧を次に示します。

- スキルの種類を変更する (スキルの OData 型が更新された)。
- スキルに固有のパラメーター (URL、defaults など) が更新された。
- スキルの出力が変更 (スキルから返される出力が追加または変更) された。
- 先祖の変更を伴うスキルの入力の変更があった (スキルのチェーンが変更された)。
- アップストリームのスキルが無効化された (このスキルへの入力となっているスキルが更新された場合)。
- ドキュメントの再プロジェクトを伴う更新がナレッジストアのプロジェクト場所に生じた。
- ドキュメントの再プロジェクトを伴う変更がナレッジストアのプロジェクトに生じた。
- インデックスに対するドキュメントの再プロジェクトを伴う変更が、インデクサーの出力フィールドのマッピングに生じた。

## キャッシュに使用される API

REST API バージョン 2020-06-30-Preview 以降では、インデクサーの追加のプロパティを使用して、インクリメンタルエンリッチメントが提供されます。スキルセットとデータソースは、一般公開されているバージョンを使用できます。操作の順序について詳しくは、リファレンスドキュメントに加えて、[インクリメンタルエンリッチメント用のキャッシュの構成](#)に関するページを参照してください。

- インデクサーの作成または更新 (api-version=2020-06-30-Preview)
- スキルセットの更新 (api-version=2020-06-30) (要求での新しい URI パラメーター)
- スキルのリセット (api-version=2020-06-30)
- データソースの更新: プレビュー API バージョンで呼び出されるときは、"ignoreResetRequirement" という名前の新しいパラメータを指定し、更新アクションによってキャッシュが無効になってはならないときはそれを true に設定する必要があります。"ignoreResetRequirement" は、簡単に検出できない不整合が意図せずデータに発生する可能性があるため、慎重に使ってください。

## 次のステップ<sup>°</sup>

インクリメンタルエンリッチメントは、変更の追跡をスキルセットと AI エンリッチメントに拡張する強力な機能です。インクリメンタルエンリッチメントを使用すると、スキルセットの設計を反復処理する際に、既存の処理済みコンテンツを再利用できます。次のステップで、インデクサーのキャッシュを有効にします。

[インクリメンタルエンリッチメントのキャッシュを有効にする](#)

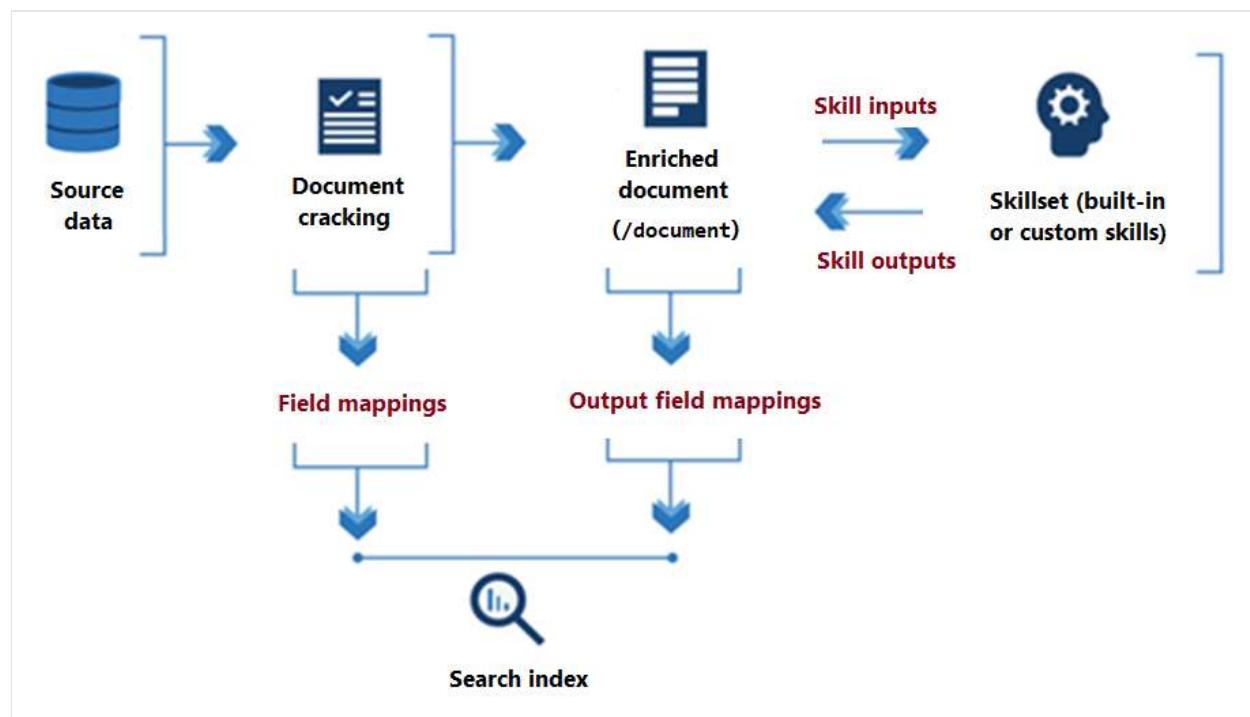
# Azure Cognitive Search のスキルセットの概念

[アーティクル] • 2023/08/08

この記事は、スキルセットの概念と構成について理解を深める必要がある開発者を対象としています。ここでは、AI エンリッチメントの高度な概念に精通していることを前提としています。

スキルセットは、インデクサーにアタッチされている Azure Cognitive Search の再利用可能なリソースです。これには、外部のデータ ソースから取得したドキュメントに対して、組み込みの AI や外部カスタム処理を呼び出す 1 つ以上のスキルが含まれています。

次の図は、スキルセット実行の基本的なデータ フローを示しています。



スキルによって行われるのは、スキルセット処理の開始から終了まで、"エンリッチされたドキュメント"に対する読み取りと書き込みです。最初は、エンリッチされたドキュメントは、データ ソースから抽出された生コンテンツだけです ("`/document`" ルートノードとして明確に表現されています)。スキルが実行されるたびに、スキルによってその出力がグラフのノードとして書き込まれるため、エンリッチされたドキュメントは、構造と実質的な内容を取得していきます。

スキルセットの実行が完了すると、エンリッチされたドキュメントの出力は、"出力フィールド マッピング" を介してインデックスに組み込まれます。ソースからインデッ

クスにそのまま転送する生コンテンツは、"フィールド マッピング" によって定義されます。

エンリッチメントを構成するには、スキルセットとインデクサーで設定を指定します。

## スキルセットの定義

スキルセットとは、画像ファイル上のテキストや OCR の翻訳など、エンリッチメントを実行する 1 つ以上の "スキル" の配列です。スキルは、Microsoft の組み込みスキル、または外部でホストするロジックを処理するためのカスタムスキルのいずれかになります。スキルセットにより、インデックスを付けるときに使用される、またはナレッジストアにプロジェクトされるエンリッチされたドキュメントが生成されます。

スキルには、コンテキスト、入力、出力があります。



- **コンテキスト**は、ドキュメントごとに 1 回、またはコレクション内のアイテムごとに 1 回の操作のスコープを指します。
- 入力はエンリッチされたドキュメント内のノードから発信されます。ここで、"source" と "name" は特定のノードを識別します。
- 出力は、エンリッチされたドキュメントに新しいノードとして返送されます。値は、ノードの "name" とノードの内容です。ノード名が重複している場合は、あいまいさを解消するためにターゲット名を設定できます。

# スキル コンテキスト

各スキルにはコンテキストがあります。コンテキストはドキュメント全体 (`/document`) であったり、ツリー内の下位のノード (`/document/countries/*`) であったりします。コンテキストによって次のことが決まります。

- 1つの値に対してスキルが実行される回数 (フィールドごと、ドキュメントごとに1回)。型コレクションのコンテキスト値の場合、`/*` を追加すると、コレクション内のインスタンスごとにスキルが1回呼び出されます。
- 出力の宣言。スキルの出力が追加される強化ツリー内の場所。出力は、常にコンテキストノードの子としてツリーに追加されます。
- 入力の形状。複数レベルのコレクションの場合、コンテキストを親コレクションに設定すると、スキル入力の形状に影響します。たとえば、国または地域のリストが含まれる強化ツリーがあり、それぞれが郵便番号のリストを含む州のリストでエンリッチメント処理されている場合、コンテキストをどのように設定するかによって、入力がどのように解釈されるかが決まります。

Context	入力	入力の形状	スキルの呼び出し
<code>/document/countries/*</code>	<code>/document/countries/*/states/*/zipcodes/*</code>	国または地域ごとにすべての郵便番号のリ	1回

Context	入力	入力の形状	スキルの呼び出し
		スト	
/document/countries/*/states/*	/document/countries/*/states/*/zipcodes/*	州内の郵便番号のリスト	国または地域と州の組み合わせごとに1回

## スキルの依存関係

スキルは、独立して並列に実行することも、あるスキルの出力を別のスキルにフィードする場合は順番に実行することもできます。次の例は、順番に実行される 2 つの組み込みスキルを示しています。

- スキル #1 は、"reviews\_text" ソース フィールドのコンテンツを入力として受け取り、そのコンテンツを出力として 5,000 文字の "pages" に分割する[テキスト分割スキル](#)です。大きなテキストを小さなチャunkに分割すると、センチメント検出などのスキルの結果が向上する可能性があります。
- スキル #2 は、"pages" を入力として受け入れ、センチメント分析の結果が含まれる "Sentiment" という名前の新しいフィールドを作成する[センチメント検出スキル](#)です。

最初のスキル ("pages") の出力が感情分析でどのように使用されているかに注目してください。ここで、"/document/reviews\_text/pages/\*" はコンテキストと入力の両方です。パスの構文の詳細については、[スキルセットで注釈を参照する方法](#)に関するページを参照してください。

```
JSON

{
  "skills": [
    {
      "@odata.type": "#Microsoft.Skills.Text.SplitSkill",
      "name": "#1",
      "description": null,
      "context": "/document/reviews_text",
      "defaultLanguageCode": "en",
      "textSplitMode": "pages",
      "maximumPageLength": 5000,
      "inputs": [
        {
          "name": "text",
          "source": "/document/reviews_text"
        }
      ],
      "outputs": [
        {
          "name": "textItems",
          "targetName": "pages"
        }
      ]
    },
    {
      "@odata.type": "#Microsoft.Skills.Text.SentimentSkill",
      "name": "#2",
      "description": null,
      "context": "/document/reviews_text/pages/*",
      "defaultLanguageCode": "en",
      "inputs": [
        {
          "name": "text",
          "source": "/document/reviews_text/pages/*"
        }
      ],
      "outputs": [
        {
          "name": "sentiment",
          "targetName": "sentiment"
        },
        {
          "name": "confidenceScores",
          "targetName": "confidenceScores"
        },
        {
          "name": "sentences",
          "targetName": "sentences"
        }
      ]
    }
  ]
}
```

```
        "targetName": "sentences"
    }
]
}
...
]
}
```

## 強化ツリー

エンリッチされたドキュメントは、スキルセットの実行中に作成された一時的なツリー状のデータ構造で、スキルを通じて行われたすべての変更を収集します。集合的に、エンリッチメントはアドレス指定可能なノードの階層として表されます。ノードには、外部データ ソースから逐語的に渡されたエンリッチされていないフィールドも含まれます。

エンリッチされたドキュメントが存在するのは、スキルセットの実行中ですが、[キャッシュ](#)したり、[ナレッジストア](#)に送信したりできます。

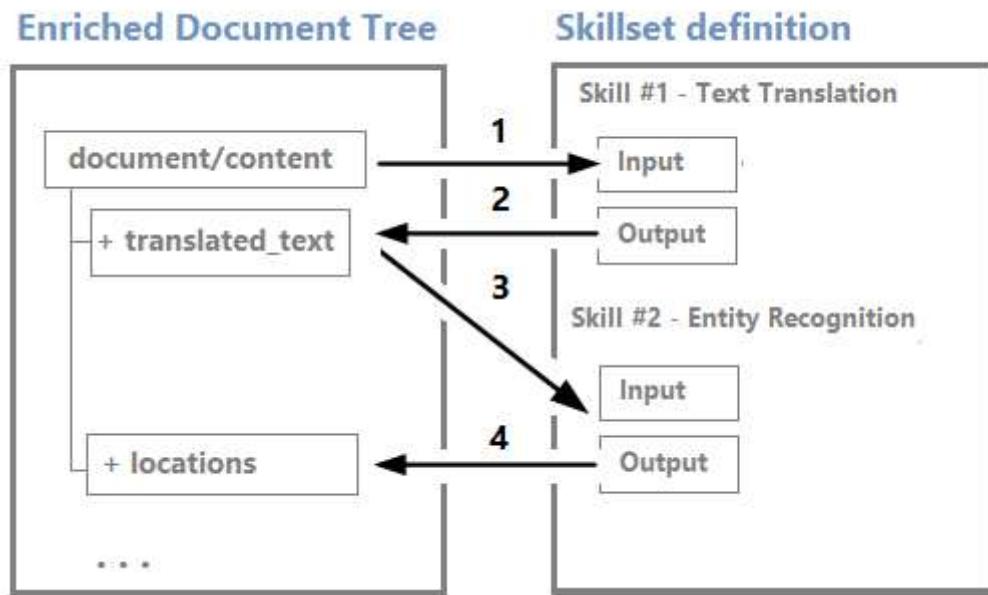
エンリッチされたドキュメントは、最初は、"ドキュメント解析" 中にデータ ソースから抽出されたコンテンツに過ぎません。ここでテキストやイメージがソースから抽出され、言語解析やイメージ解析に利用できるようになります。

最初のコンテンツはメタデータと "ルート ノード" (`document/content`) です。ルート ノードは、通常、ドキュメント全体であるか、ドキュメント解析中にデータ ソースから抽出された正規化されたイメージです。強化ツリーでの表現方法は、データ ソースの種類によって異なります。次の表は、サポートされているいくつかのデータ ソースについて、エンリッチメントパイプラインに入ったドキュメントの状態を示しています。

データ ソース/解析モード	Default	JSON、JSON 行、および CSV
Blob Storage	/document/content /document/normalized_images/* ...	/document/{key1} /document/{key2} ...
Azure SQL	/document/{column1} /document/{column2} ...	該当なし
Azure Cosmos DB	/document/{key1} /document/{key2} ...	該当なし

スキルが実行されると、出力が新しいノードとしてエンリッチメントツリーに追加されます。スキルの実行がドキュメント全体に対して行われる場合、ノードはルートの下の最初のレベルに追加されます。

ノードは、ダウンストリームスキルの入力として使用できます。たとえば、翻訳された文字列などのコンテンツを作成するスキルは、エンティティを認識したり、キーフレーズを抽出したりするスキルの入力になる可能性があります。



デバッグセッションのビジュアルエディターを使用してエンリッチメントツリーを視覚化して操作できますが、ほとんどは内部構造です。

エンリッチメントは変更できません。ノードは一度作成されたら編集できません。スキルセットや強化ツリーの複雑さが増しても、強化ツリー内のすべてのノードをインデックスやナレッジストアにする必要はありません。

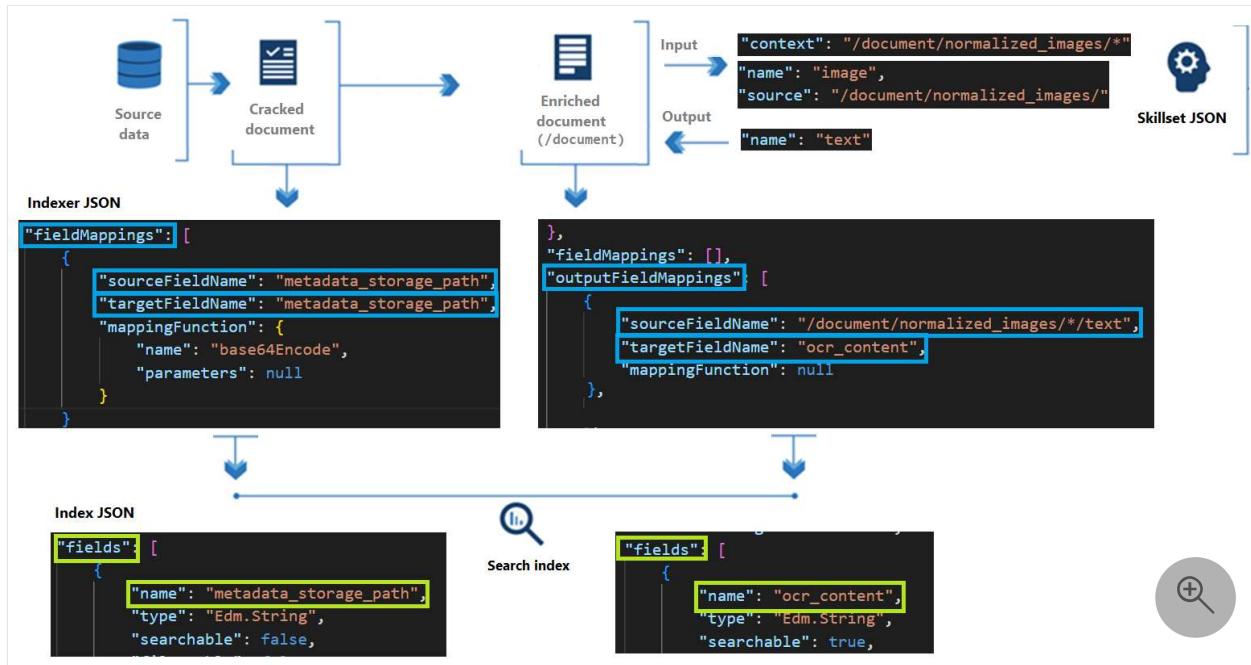
エンリッチメント出力のサブセットのみを選択的に保持して、使用する情報のみを保持することができます。インデクサー定義の[出力フィールドマッピング](#)によって、検索インデックスに実際に取り込まれるコンテンツが決まります。同様に、ナレッジストアを作成する場合は、プロジェクトに割り当てられている[シェイプ](#)に、出力をマップできます。

### ① 注意

強化ツリー形式により、エンリッチメントパイプラインでは、メタデータをプリミティブデータ型にもアタッチできます。メタデータは有効な JSON オブジェクトになりませんが、ナレッジストア内のプロジェクト定義で有効な JSON 形式にプロジェクトできます。詳細については、[Shaper スキル](#)に関するページをご覧ください。

# インデクサーの定義

インデクサーには、インデクサーの実行を構成するために使用されるプロパティとパラメーターがあります。これらのプロパティの中には、検索インデックス内のフィールドにデータ パスを設定するマッピングがあります。



マッピングには次の 2 つのセットがあります。

- "fieldMappings" は、ソース フィールドを検索 フィールドにマップします。
- "outputFieldMappings" は、エンリッチされたドキュメント内のノードを検索 フィールドにマップします。

"sourceFieldName" プロパティは、データ ソース内のフィールドまたはエンリッチメントツリー内のノードのいずれかを指定します。 "targetFieldName" プロパティは、コンテンツを受け取るインデックス内の検索 フィールドを指定します。

## エンリッチメントの例

この例では、[ホテル レビュースキルセット](#)を参照ポイントとして使用し、スキルの実行によって強化ツリーがどのように発展するかを概念図を使って説明します。

また、この例では以下もわかります。

- スキルの実行回数の決定に対して、スキルのコンテキストと入力がどのように働くか
- コンテキストに基づく入力の形状

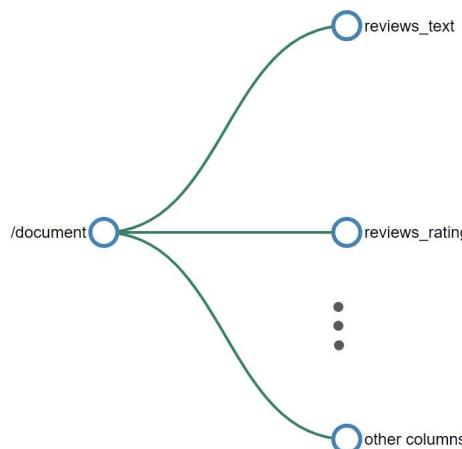
この例では、CSV ファイルのソース フィールドに、ホテルに関する顧客レビュー ("reviews\_text") と評価 ("reviews\_rating") が含まれています。インデクサーによって、BLOB ストレージからメタデータ フィールドが追加され、スキルによって、翻訳されたテキスト、センチメント スコア、キーフレーズ検出が追加されます。

ホテルレビューの例で、エンリッチメントプロセス内の "ドキュメント" は 1 つのホテルレビューを表します。

### 💡 ヒント

このデータの検索インデックスやナレッジ ストアを作成するには、[Azure portal](#)、[Postman](#)、または [REST API](#) を使用します。 **デバッグ セッション**を使用して、スキルセットの構成、依存関係、強化ツリーへの影響の分析情報を確認することもできます。この記事のイメージは、デバッグ セッションからプルされます。

概念的には、最初の強化ツリーは次のようにになります。



すべての強化のルート ノードは `"/document"` です。 BLOB インデクサーを使用すると、`"/document"` ノードに子ノード `"/document/normalized_images"` と `"/document/content"` ができます。この例のように、データが CSV の場合、列名は `"/document"` の下のノードにマップされます。

## スキル #1: 分割スキル

ソース コンテンツが大量のテキストで構成されている場合は、言語、センチメント、キーフレーズ検出の精度を高めるために、より小さいコンポーネントに分割すると便利

です。 使用できるグレインは、ページと文の 2 つです。 ページは約 5,000 文字で構成されます。

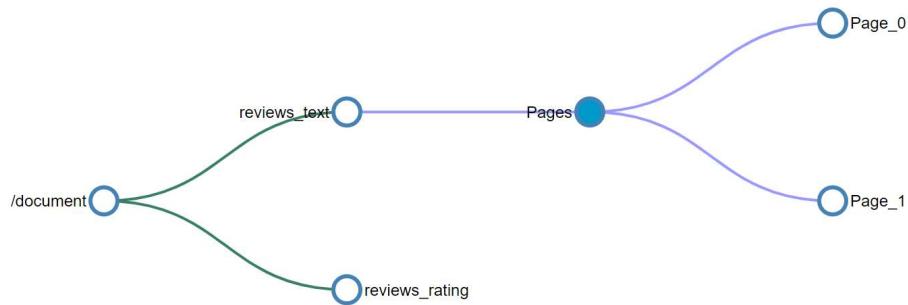
通常、テキスト分割スキルは、スキルセットにおいて最初に選ばれます。

#### JSON

```
"@odata.type": "#Microsoft.Skills.Text.SplitSkill",
"name": "#1",
"description": null,
"context": "/document/reviews_text",
"defaultLanguageCode": "en",
"textSplitMode": "pages",
"maximumPageLength": 5000,
"inputs": [
{
    "name": "text",
    "source": "/document/reviews_text"
},
],
"outputs": [
{
    "name": "textItems",
    "targetName": "pages"
}]
```

"/document/reviews\_text" のスキルコンテキストでは、この分割スキルは reviews\_text に対して 1 回実行されます。 スキルの出力はリストであり、 reviews\_text が 5000 文字のセグメントに分割されています。 分割スキルからの出力は、 pages という名前が付けられ、エンリッチメントツリーに追加されます。 targetName 機能を使用することで、強化ツリーに追加される前に、スキル出力の名前を変更できます。

これで、強化ツリーのスキルのコンテキストの下に、新しいノードが配置されました。 このノードは、任意のスキル、プロジェクト、または出力フィールド マッピングで使用できます。

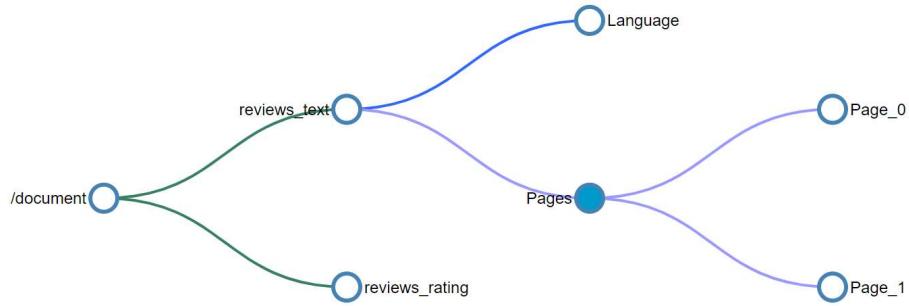


スキルによってノードに追加されたいずれかの強化にアクセスするには、強化の完全なパスが必要です。たとえば、`pages` ノードのテキストを別のスキルへの入力として使用する場合は、`"/document/reviews_text/pages/*"` として指定する必要があります。パスの詳細については、[注釈の参照](#)に関するページをご覧ください。

## スキル #2: 言語検出

ホテルレビュー ドキュメントには、複数の言語で表される顧客フィードバックが含まれています。言語検出スキルによって、どの言語が使用されているか判断されます。結果は、キー フレーズ抽出とセンチメント検出に渡され(非表示)、センチメントと語句を検出するときに言語が考慮されます。

言語検出スキルは、スキルセットで定義されている 3 番目のスキル(スキル #3)ですが、次に実行されるスキルです。入力は必要ないので、前のスキルと並行して実行されます。先行する分割スキルと同様に、言語検出スキルもドキュメントごとに 1 回呼び出されます。強化ツリーには言語用の新しいノードが追加されています。

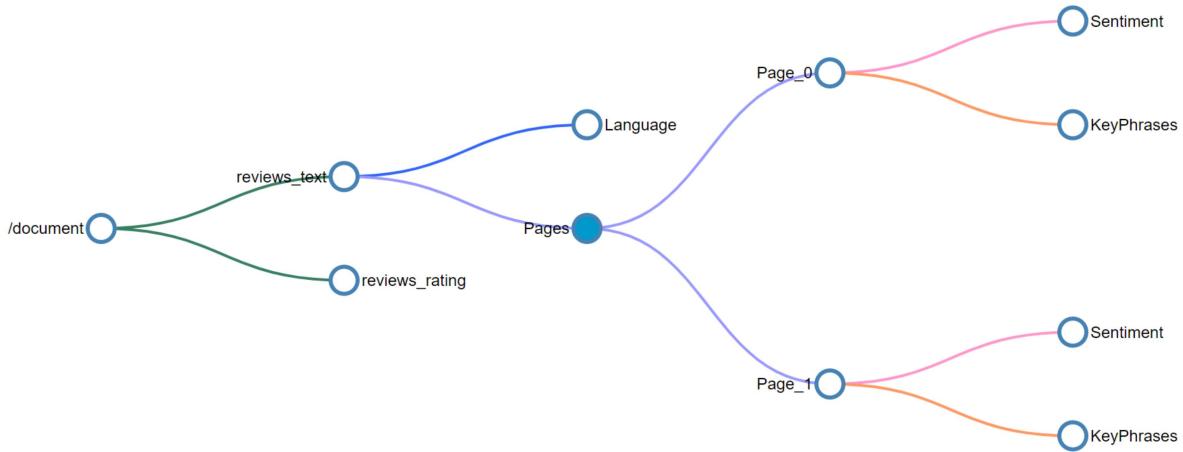


## スキル #3 と #4 (感情分析とキー フレーズ検出)

お客様からのフィードバックには、ポジティブなものからネガティブなものまで、さまざまなエクスペリエンスが反映されています。フィードバックは、感情分析スキルによって分析され、負数から正数の範囲内で一連のスコアが割り当てられます。センチメントがはっきりしない場合は、中立が割り当てられます。感情分析と同時に、結果的に重要であると思われる単語と短いフレーズが、キー フレーズ検出によって特定され、抽出されます。

コンテキストとして `/document/reviews_text/pages/*` が指定された感情分析スキルとキー フレーズ スキルは両方とも、`pages` コレクション内の項目ごとに 1 回ずつ呼び出されます。スキルからの出力は、関連付けられている `page` 要素の下のノードになります。

スキルセットに含まれる残りのスキルを表示し、各スキルの実行によって強化のツリーがどのように成長し続けるかを視覚化できるようになるはずです。マージスキルや Shaper スキルなどの一部のスキルでも新しいノードが作成されますが、既存のノードのデータのみが使用され、新しい強化は作成されません。



上のツリーのコネクタの色は、エンリッチメントが異なるスキルで作成されたことを示しています。ノードは個別に指定する必要があり、親ノードを選択したときに返されるオブジェクトの一部にはなりません。

## スキル #5: Shaper スキル

出力に[ナレッジストア](#)が含まれる場合は、最後のステップとして [Shaper スキル](#)を追加します。 Shaper スキルによって、強化ツリー内のノードからデータ シエイプが作成されます。 たとえば、複数のノードを 1 つのシェイプに統合することができます。 その後、このシェイプをテーブルとして射影し(ノードはテーブル内の列になります)、名前によってシェイプをテーブル プロジェクションに渡します。

整形が 1 つのスキルにまとめられているため、Shaper スキルは簡単に操作できます。また、個々のプロジェクト内でインライン整形を選ぶこともできます。 Shaper スキルによってエンリッチメントツリーの加減が行われることはありません。したがって、これは視覚化されません。代わりに、Shaper スキルは、既にあるエンリッチメントツリーを再現する手段と考えることができます。 概念的には、これはデータベース内のテーブルからビューを作成するのと似ています。

```

JSON

{
    "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",
    "name": "#5",
    "description": null,
    "context": "/document",
    "inputs": [
        {
            "name": "name",
            "source": "/document/name"
        },
        {
            "name": "text",
            "source": "/document/reviews/text"
        }
    ],
    "outputs": [
        {
            "name": "sentiment",
            "target": "Sentiment"
        },
        {
            "name": "keyphrases",
            "target": "KeyPhrases"
        }
    ]
}

```

```
{
  "name": "reviews_date",
  "source": "/document/reviews_date"
},
{
  "name": "reviews_rating",
  "source": "/document/reviews_rating"
},
{
  "name": "reviews_text",
  "source": "/document/reviews_text"
},
{
  "name": "reviews_title",
  "source": "/document/reviews_title"
},
{
  "name": "AzureSearch_DocumentKey",
  "source": "/document/AzureSearch_DocumentKey"
},
{
  "name": "pages",
  "sourceContext": "/document/reviews_text/pages/*",
  "inputs": [
    {
      "name": "Sentiment",
      "source": "/document/reviews_text/pages/*/Sentiment"
    },
    {
      "name": "LanguageCode",
      "source": "/document/Language"
    },
    {
      "name": "Page",
      "source": "/document/reviews_text/pages/*"
    },
    {
      "name": "keyphrase",
      "sourceContext": "/document/reviews_text/pages/*/Keyphrases/*",
      "inputs": [
        {
          "name": "Keyphrases",
          "source": "/document/reviews_text/pages/*/Keyphrases/*"
        }
      ]
    }
  ],
  "outputs": [
    {
      "name": "output",
      "targetName": "tableprojection"
    }
  ]
}
```

]  
}

## 次のステップ°

概要と例を参考にしながら、[組み込みスキル](#)を使用して、最初のスキルセットを作成してみてください。

[最初のスキルセットを作成する](#)

# Azure AI Search 内の統合データのチャンキングと埋め込み

[アーティクル] • 2024/03/27

## ① 重要

この機能はパブリック プレビュー段階にあり、[追加使用条件](#) の下で提供されます。この機能は、[2023-10-01-Preview REST API](#) でサポートされます。

統合ベクター化によって、データ チャンキングとテキスト-to-ベクター埋め込みがインデクサーベース インデックス作成のスキルに追加されます。テキスト-to-ベクター変換もクエリに追加されます。

この機能は、プレビューのみ段階です。一般提供バージョンの[ベクトル検索](#)と以前のプレビュー バージョンでは、データのチャンキングとベクター化が外部のチャンキングとベクターのコンポーネントに依存しており、アプリケーションコードが各手順を操作し、調整する必要があります。このプレビュー版では、チャンキングとベクター化がスキルおよびインデクサーを通じてインデックス作成に組み込まれています。テキスト分割スキルを使用してデータをチャunkするスキルセットをセットアップして、それから AzureOpenAIEmbedding スキルまたはカスタム スキルを使用して埋め込みモデルを呼び出すことができます。インデックス作成時に使用されるあらゆるベクトル化を、テキストをベクターに変換するクエリで呼び出すこともできます。

インデックス作成の場合、統合ベクター化では以下が必要です。

- サポートされるデータ ソースからデータを取得する[インデクサー](#)。
- テキスト分割スキルを呼び出してデータをチャunkする[スキルセット](#)と、[AzureOpenAIEmbedding スキル](#)とデータをベクター化するための[カスタム スキル](#)のいずれか。
- チャunkおよびベクター化した内容を受け取るための[1 つ以上のインデックス](#)。

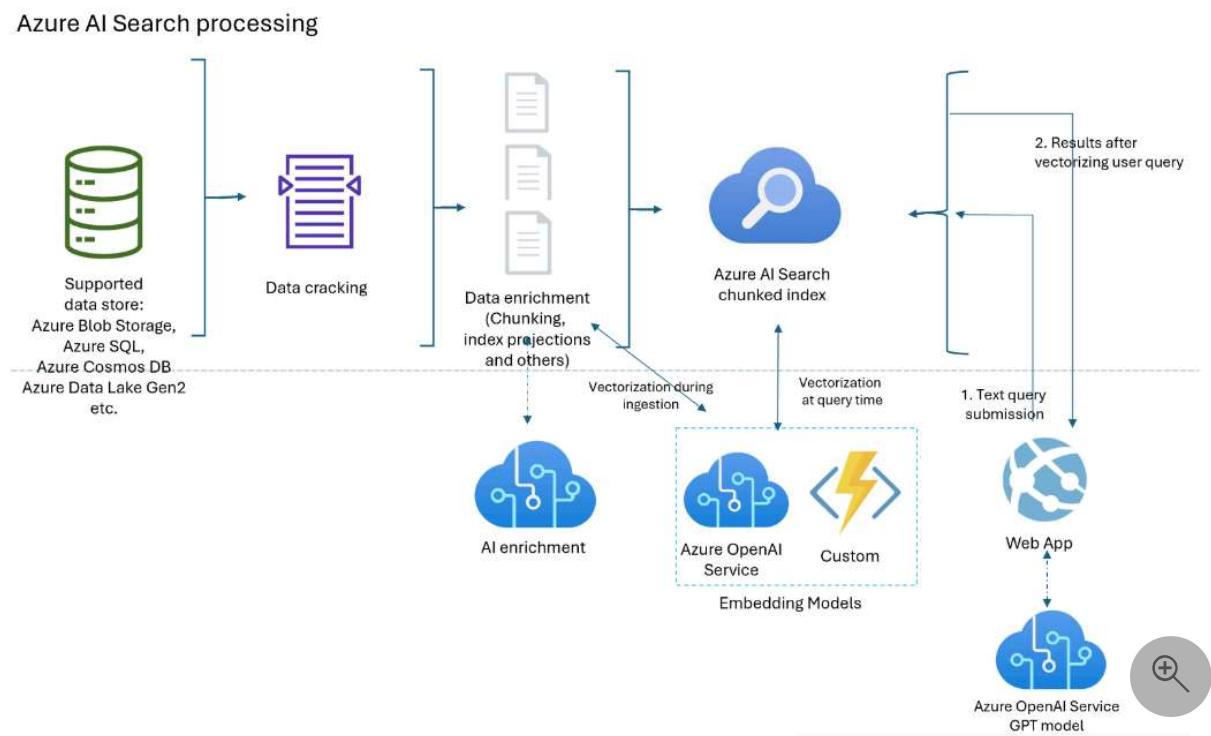
クエリの場合:

- インデックススキーマで定義され、ベクター フィールドに割り当てられて、自動的にクエリ時に使用されてテキスト クエリをベクターに変換する[ベクター化](#)。

ベクター変換は、テキスト-to-ベクターの一方向です。クエリと結果にはベクター-to-テキスト変換がありません (たとえば、ベクター結果を人間が読み取り可能な文字列に変換することはできません)。

# コンポーネント図

次の図は、統合ベクター化の構成要素を示しています。



こちらが統合ベクター化のための構成要素のチェックリストです。

- インデクサーベースのインデックス作成でサポートされているデータソース。
- ベクター フィールドを指定するインデックスと、ベクター フィールドに割り当てられたベクター化定義。
- データ チャンキングのためのテキスト分割スキルを提供するスキルセットと、ベクター化のスキル (AzureOpenAiEmbedding スキルと、外部埋め込みモデルをポイントするカスタム スキルのいずれか)。
- オプションとして、チャンクしたデータをセカンダリ インデックスにプッシュするインデックス プロジェクション (スキルセットにも定義される)
- 埋め込みモデル (Azure OpenAI でデプロイされているか、HTTP エンドポイントを通じて提供される)。
- プロセスをエンドツーエンドで進めるためのインデクサー。インデクサーでは、変更検出のスケジュール、フィールド マッピング、優先度も指定されます。

このチェックリストは統合ベクター化に重点を置いていますが、お持ちのソリューションはこのリストに限定されません。AI エンリッチメントのためのスキルを増やし、ナレッジストアを作成し、セマンティック ランク付けを追加し、関連性チューニングや他のクエリ機能を追加することができます。

## 可用性と料金

統合ベクター化の可用性は、埋め込みモデルに基づきます。 Azure OpenAI を使用している場合は、「[リージョン別の提供状況](#)」を確認してください。

カスタム スキルと Azure ホスティング メカニズム (Azure 関数アプリ、Azure Web アプリ、Azure Kubernetes など) を使用している場合は、[リージョン別の製品ページ](#)で機能の可用性について確認してください。

データ チャンкиング (テキスト分割スキル) は無料で、すべての地域のすべての Azure AI サービスでご利用になれます。

### ① 注意

2019 年 1 月 1 日より前に作成された一部の古い検索サービスは、ベクトル ワークロードをサポートしないインフラストラクチャにデプロイされています。ベクトル フィールドをスキーマに追加しようとしてエラーが表示された場合、それはサービスが古いためです。このような場合は、ベクトル機能を試すために新しい検索サービスを作成する必要があります。

## 統合ベクター化をサポートできるのはどんなシナリオですか？

- 大きなドキュメントをチャunkに再分割すると、ベクターおよび非ベクターシナリオに便利です。ベクターの場合、埋め込みモデルの入力制約に合わせるのにチャunkが役立ちます。非ベクター シナリオの場合、チャットスタイルの検索アプリで GPT がインデックス作成したチャunkからの応答をアセンブルしています。ベクトル化 (または非ベクトル化)されたチャunkをチャットスタイルの検索に使用できます。
- フィールドのすべてがベクター フィールドであり、ドキュメント ID (検索インデックスに必要) が唯一の文字列フィールドであるベクター ストアを構築します。ベクター ストアにクエリを実行してドキュメント ID を取得し、ドキュメントのベクター フィールドを別のモデルに送信します。
- ベクターおよびテキスト フィールドを組み合わせて、セマンティック ランク付けを使用した (または使用しない) ハイブリッド検索にします。統合ベクター化によってベクター検索でサポートされるシナリオのすべてが簡略化されます。

## 統合ベクター化を使用するのはどのようなときか

組み込み統合ベクター化サポートの Azure AI Studio を使用することをお勧めします。この方法でお客様のニーズが満たされない場合は、Azure AI Search のプログラマティックインターフェイスを使用して統合ベクター化を呼び出すインデクサーとスキルセットを作成することができます。

## 統合ベクター化の使用方法

クエリ専用ベクター化の場合:

1. インデックスにベクター化を追加します。インデックスにベクターを生成するために使用したのと同じ埋め込みモデルになるはずです。
2. ベクター プロファイルにベクター化を割り当て、それからベクター プロファイルをベクター フィールドに割り当てます。
3. ベクター化するテキスト文字列を指定するベクター クエリを作成します。

より一般的なシナリオ - インデックス作成時のデータのチャンкиングとベクター化:

1. インデクサーベースのインデックス作成でサポートされているデータ ソースへのデータ ソース接続を作成します。
2. チャンкиング用のテキスト分割スキルと、AzureOpenAIEmbeddingModel またはチャンクをベクター化するカスタムスキルを呼び出すスキルセットを作成します。
3. クエリ時のベクター化を指定し、それをベクター フィールドに割り当てるインデックスを作成します。
4. データの取得からスキルセット実行まで、インデックス作成を通してすべてを進めるためのインデクサーを作成します。

オプションとして、チャンクしたコンテンツが一方のインデックス上にあり、チャンクされていないコンテンツが別のインデックスにある高度なシナリオのためのセカンダリインデックスを作成します。チャンクしたインデックス(セカンダリインデックス)は RAG アプリで役立ちます。

### 💡 ヒント

Azure portal で新しい [データのインポートとベクトル化] ウィザードを試して、コードを記述する前に統合ベクター化を探索します。

あるいは、同じワークフローを実行するための Jupyter ノートブックをセルごとに構成して、各手順がどう機能するかを調べます。

## 制限事項

Azure OpenAI の埋め込みモデルのクオータと制限について理解します。 Azure AI Search には再試行ポリシーがありますが、クオータを使い果たすと、再試行が失敗します。

Azure OpenAI の 1 分あたりトーカンの制限は、モデルごと、サブスクリプションごとに設けられています。埋め込みモデルをクエリとインデックス作成の両ワークフローで使用している場合は、このことを覚えておいてください。可能であれば、[ベストプラクティスに従ってください](#)。ワークフローごとに埋め込みモデルを用意して、それらを別々のサブスクリプションでデプロイするようにしてください。

Azure AI Search では、[サービスの制限](#)がレベルおよびワークフロー別にあることを忘れないでください。

最後に、次の機能は現在サポートされていません。

- [カスタマー マネージド暗号化キー](#)
- ベクター化への[共有プライベート リンク接続](#)
- 現在は、統合型データ チャンкиングおよびベクター化のためのバッチ処理がありません

## 統合ベクター化のメリット

統合ベクター化の重要なメリットのいくつかを紹介します。

- データ チャンкиングとベクター化の分離したパイプラインがありません。コードの書き込みと維持がより簡単です。
- エンドツー エンドのインデックス作成を自動化します。ソース (Azure Storage、Azure SQL、Cosmos DB など) でデータが変更されると、インデクサーはこれらの更新を、パイプライン全体 (取得からドキュメントの解読まで) で、オプションの AI エンリッチメント、データ チャンкиング、ベクター化、インデックス作成を通じて進めることができます。
- チャンクしたコンテンツをセカンダリ インデックスに射影します。セカンダリ インデックスは他の検索インデックス (フィールドや他のコンストラクトを持つスキーマ) のように作成されますが、インデクサーによりプライマリ インデックスと並行して作成されます。各ソース ドキュメントのコンテンツが、同じインデックス作成実行中に、プライマリおよびセカンダリ インデックスのフィールドへ流れていきます。

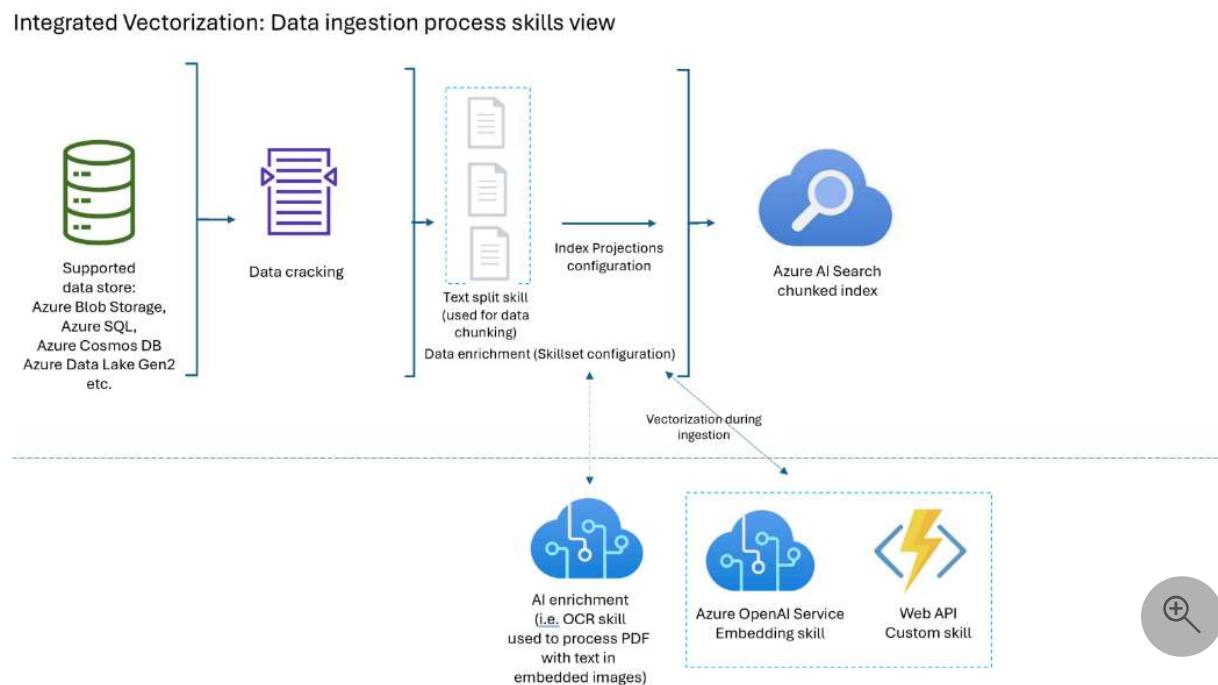
セカンダリ インデックスの目的は、データ チャンкиングおよび取得拡張生成 (RAG) アプリです。サイズの大きな PDF をソース ドキュメントとして想定すると、プライマリ インデックスには基本情報 (タイトル、日付、作成者、説明) があ

り、セカンダリ インデックスにはコンテンツのチャンクがあります。データ チャンク レベルのベクター化によって、関連する情報を見つけて (各チャンクが検索可能である) 関連する応答を返すのが、特にチャットスタイルの検索アプリでは簡単になります。

## チャンク後のインデックス

チャンキングとは、コンテンツをより小さな管理可能部分 (チャンク) に分割することで、それらを別々に処理できるようにするプロセスです。チャンキングが必要になるのは最大入力サイズの埋め込みモデルや大型言語モデルでソース ドキュメントが大きすぎるけれども、それによって [RAG パターン](#) やチャットスタイル検索でインデックス構造がよくなると考えられる場合です。

次の図は、チャンク後インデックス作成の構成要素を示しています。



## 次のステップ<sup>°</sup>

- 検索インデックスにベクター化を構成する
- スキルセットにインデックス投影を構成する

# Azure Cognitive Search でのフルテキスト検索

[アーティクル] • 2023/09/27

この記事は、Azure Cognitive Search におけるフルテキスト検索のしくみについて理解を深める必要がある開発者を対象としています。テキストクエリに関して、Azure Cognitive Search はほとんどの状況で速やかに適切な結果を返します。しかし一見、間違っているのではないか、と思うような結果が返されることも皆無ではありません。このような状況では、Lucene による 4 段階から成るクエリ実行（クエリ解析、字句解析、文書のマッチング、スコア付け）についての背景知識があると、具体的にどのような変更をクエリパラメーターやインデックス構成に加えれば目的の結果が生成されるかが特定しやすくなります。

## ① 注意

Azure Cognitive Search では、フルテキスト検索に Apache Lucene<sup>1</sup> が使われていますが、Lucene の機能がそのままの形で統合されているわけではありません。

Microsoft は、Azure Cognitive Search にとって重要なシナリオを実現する Lucene の機能を選んで公開、拡張しています。

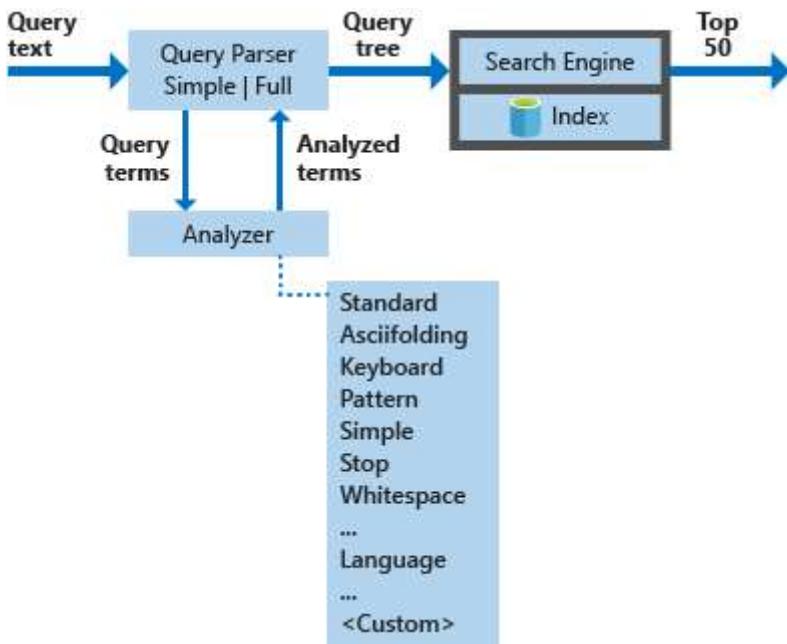
## アーキテクチャの概要と図

クエリの実行には次の 4 つの段階があります。

1. クエリ解析
2. 字句解析
3. 文書検索
4. ポイントの計算

フルテキスト検索クエリは、クエリテキストを解析して検索語と演算子を抽出することから始まります。2つのパーサーがあり、速度と複雑さを選択できます。次に分析段階では、個々の検索語がしばしば分解され、新しい形に再構築されます。この手順では、できるだけ広い範囲から "一致と見なす候補" が得られます。検索エンジンは、インデックスをスキャンして、一致する語句を持つドキュメントを検索し、各一致をスコア付けします。その後、一致する個々の文書に割り当てられた関連度スコアに基づいて結果セットが並べ替えられます。このようにランク順に並んだリストの先頭の結果が、呼び出し元のアプリケーションに返されます。

以下の図は、検索要求の処理に使用されるコンポーネントを示しています。



## 主要コンポーネント 機能の説明

<b>クエリ パーサー</b>	クエリ演算子から検索語を切り離し、検索エンジンに送るクエリ構造(クエリツリー)を作成します。
<b>アナライザー</b>	検索語に対する字句解析を実行します。このプロセスには、検索語の変換、削除、拡大が伴う場合があります。
<b>インデックス</b>	索引付けの対象文書から抽出した検索可能な語句を効率よく体系的に格納するデータ構造です。
<b>検索エンジン</b>	転置インデックスの内容に基づいて、一致する文書を検索してスコア付けします。

## 検索要求の構造

検索要求は、結果セットで返すべき内容を詳細に規定した仕様です。最も単純な形式は、どのような種類の条件も含まれていない空のクエリです。しかしそれより現実的な例では、パラメーターや複数の検索語を伴うのが一般的です。場合によっては、検索範囲を特定のフィールドに限定したり、フィルター式や並べ替え規則が使われたりすることもあります。

次の例は、REST API を使用して Azure Cognitive Search に送信できる検索要求です。

```
POST /indexes/hotels/docs/search?api-version=2020-06-30
{
    "search": "Spacious, air-condition* +\"Ocean view\",
    "searchFields": "description, title",
```

```
    "searchMode": "any",
    "filter": "price ge 60 and price lt 300",
    "orderby": "geo.distance(location, geography'POINT(-159.476235
22.227659)'),",
    "queryType": "full"
}
```

この要求に対して、検索エンジンは次の操作を実行します。

1. 値格が \$60 以上 \$300 未満の文書を検索します。
2. クエリを実行します。この例では、検索クエリが `"Spacious, air-condition*`  
`+\"Ocean view\""` という語句で構成されています (通常はユーザーが句読点を入力することはありませんが、この例では、アナライザーによる処理を説明するためにあえて含めています)。

このクエリの場合、検索エンジンは、"searchFields" に指定された説明フィールドとタイトルフィールドをスキャンして、`"Ocean view"` を含む文書、さらに `"spacious"` という語句、または `"air-condition"` というプレフィックスで始まる語句を探します。明示的に必須指定 (+) されていない語句に関して、マッチング対象を任意 (既定) とするか、すべてとするかが、"searchModel" パラメーターで指定されています。

3. 結果として得られた一連のホテルを特定の地理的位置に近い順に並べ替え、結果を呼び出し元のアプリケーションに返します。

この記事では主に、検索クエリ: `"Spacious, air-condition* +\"Ocean view\""` の処理について取り上げています。フィルター処理と並べ替えについては取り上げません。詳細については、[Search API のリファレンス ドキュメント](#)を参照してください。

## 第 1 段階: クエリ解析

前出のとおり、検索要求の最初の行がクエリ文字列です。

```
"search": "Spacious, air-condition* +\"Ocean view\"",
```

クエリ パーサーは、検索語から演算子 (この例の `*` や `+`) を切り離し、検索クエリを "サブクエリ" に分解します。次の種類のサブクエリがサポートされています。

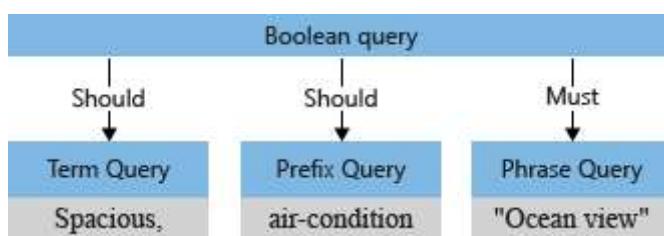
- "単語検索": 独立した語を検索 (`spacious` など)
- "フレーズ検索": 引用符で囲まれた語句を検索 (`ocean view` など)

- ・ "プレフィックス検索": 語句 + プレフィックス演算子 `*` を検索 (air-condition など)

サポートされているクエリの種類すべての一覧については、[Lucene のクエリ構文](#)に関するページを参照してください

文書が一致していると見なされるために検索条件との一致が "必須 (must)" であるのか "勧告 (should be)" であるのかは、サブクエリに関連付けられている演算子によって決まります。たとえば、`+ "Ocean view"` は `+` 演算子が指定されているので検索条件との一致が "必須" となります。

クエリパーサーは、サブクエリを "クエリツリー" (クエリを表す内部的な構造) に再構築して検索エンジンに渡します。クエリ解析の第 1 段階で、クエリツリーは次のようにになります。



## サポートされるパーサー: Simple と Full Lucene

Azure Cognitive Search では、`simple` (既定) と `full` の 2 種類のクエリ言語が使用されます。どちらのクエリ言語を使うかは、検索要求で `queryType` パラメーターの設定で指定します。その指定に基づいて、クエリパーサーが演算子と構文を解釈します。

- ・ [Simple クエリ言語](#)は直感的で安定しており、多くの場合、クライアント側の処理を行わなくてもユーザー入力をそのまま解釈するのに適しています。Web 検索エンジンで多く使われているクエリ演算子がサポートされます。
- ・ [Full Lucene クエリ言語](#)は、`queryType=full` を設定することによって利用できます。より多くの演算子やクエリの種類 (ワイルドカード、あいまい一致、正規表現、フィールド指定検索など) がサポートされることで、既定の Simple クエリ言語が拡張されます。たとえば、Simple クエリ構文で送信された正規表現は、式としてではなくクエリ文字列と解釈されます。この記事で紹介している要求の例では、Full Lucene クエリ言語を使用しています。

## searchMode がパーサーに及ぼす影響

解析方法に作用する検索要求パラメーターとしては、他にも "searchMode" パラメーターがあります。これは、ブールクエリの既定の演算子、つまり `any` (既定) または `all` を制御します。

"searchMode=any"とした場合(既定)、spaciousとair-conditionとの間に区切り記号として置かれた空白はOR(||)の働きをするため、サンプルクエリテキストは次のクエリテキストに相当します。

```
Spacious,||air-condition*+"Ocean view"
```

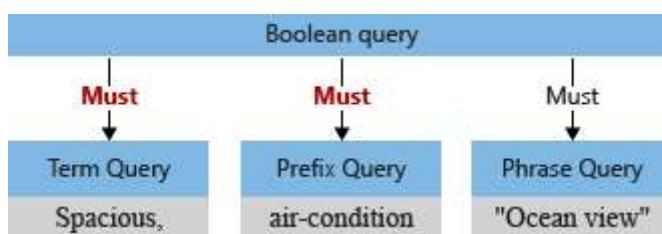
ブルクエリの構造において、明示的な演算子(+ "Ocean view" の+など)の意味ははつきりしています。つまり検索条件との一致は"必須(must)"です。それに比べて、残りの語句(spaciousとair-condition)の解釈はあいまいです。検索エンジンが探すべきなのは、ocean viewとspaciousとair-conditionの"すべて"との一致でしょうか。それとも、ocean viewに加えて、残りの2つの語句のうち、"どちらか一方"のみが含まれていればよいのでしょうか。

既定("searchMode=any")では、検索エンジンはより広い解釈を想定します。どちらか一方のフィールドが一致していればよい(should)の意味、つまり"or"のセマンティクスで解釈されます。先ほど例に挙げた、2つの"should"演算を含んだクエリツリーは既定の動作を示しています。

では、"searchMode=all"と設定したらどうなるでしょうか。この場合は、空白文字が"and"演算と解釈されます。残りの2つの語句が両方とも文書に存在したときに初めて一致と見なされます。最終的にサンプルクエリは次のように解釈されます。

```
+Spacious,+air-condition*+"Ocean view"
```

変更後のクエリツリーは次のようになり、一致する文書は3つすべてのサブクエリの積集合となります。



## ① 注意

"searchMode=all"より"searchMode=any"を選ぶ場合は、代表的なクエリを実行したうえで判断することをお勧めします。普段から演算子を指定するユーザー(ドキュメントストアを検索するときなど)は、"searchMode=all"で得られるブルクエリの構造の方が直感的にわかりやすいかもしれません。 "searchMode"と演算

子の相互作用について詳しくは、「Simple クエリ構文」に関するページをご覧ください。

## 第 2 段階: 字句解析

クエリツリーが構築された後、"単語検索" と "フレーズ検索" のクエリがアナライザーによって加工されます。アナライザーは、パーサーから渡されたテキスト入力を受け取ってそのテキストを加工してから、トークン化した語句をクエリツリーに組み入れます。

最も一般的な字句解析は、特定の言語に固有の規則に従って検索語を変換する \*言語分析です。

- 検索語を単語の原形にします。
- **ストップワード**、つまり検索上の重要性がさほど高くない単語 (英語であれば "the" や "and") を削除します
- 複合語をその構成要素に分解します。
- 単語の大文字を小文字に変換します。

通常はこれらの操作をひととおり適用することで、ユーザーによって入力されたテキストと、インデックスに格納されている語句との相違点が取り除かれます。こうした操作はテキスト処理の範囲を超えており、言語そのものに対する深い知識が必要となります。この言語知識のレイヤーを追加するために、Azure Cognitive Search は、Lucene と Microsoft から提供されているさまざまな言語アナライザーに対応しています。

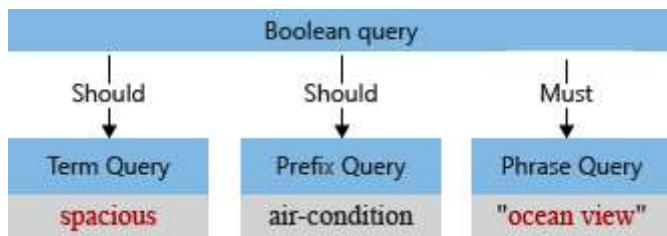
### ① 注意

解析要件は、実際のシナリオによって大きく異なります。ごく最低限で済む場合もあれば、膨大な作業が必要となる場合もあります。字句解析の難易度は、あらかじめ定義されているいづれかのアナライザーを選択するか、**カスタム アナライザー**を独自に作成するかによって決まります。アナライザーは、検索可能なフィールドにその適用対象が限定されており、フィールド定義の一環として指定されます。これにより、フィールドごとに多様な字句解析を行うことができます。指定されなかった場合は、"標準" の Lucene アナライザーが使用されます。

この例では、解析前の最初のクエリツリーに "Spacious," という語句があり、大文字の "S" とコンマはクエリパーサーによって検索語の一部として解釈されています (コンマはクエリ言語の演算子とは見なされません)。

既定のアナライザーは、この語句を加工する際、"ocean view" と "spacious" に含まれている大文字を小文字に変換し、さらにコンマを削除します。変更後のクエリツリー

は次のようにになります。



## アナライザーの動作テスト

アナライザーの動作は、[Analyze API](#) を使ってテストすることができます。解析したいテキストを入力すると、指定したアナライザーからどのような語句が生成されるかを確認できます。たとえば、標準アナライザーで "air-condition" というテキストがどのように加工されるかを確認するには、次のように要求します。

```
JSON

{
  "text": "air-condition",
  "analyzer": "standard"
}
```

標準アナライザーは、入力テキストを次の 2 つのトークンに分解します。その際、開始オフセットと終了オフセット (一致部分の強調表示に使用される) やその位置 (フレーズの照合に使用される) など、各種の属性を使ってそれらに注釈を付けます。

```
JSON

{
  "tokens": [
    {
      "token": "air",
      "startOffset": 0,
      "endOffset": 3,
      "position": 0
    },
    {
      "token": "condition",
      "startOffset": 4,
      "endOffset": 13,
      "position": 1
    }
  ]
}
```

## 字句解析の例外

字句解析が適用されるのは、語句全体を必要とする種類の検索(単語検索とフレーズ検索)だけです。語句全体を必要としない種類の検索(プレフィックス検索、ワイルドカード検索、正規表現検索など)やあいまい検索には適用されません。前出の例の `air-condition*` という語を使ったプレフィックス検索も含め、こうした種類の検索は、解析段階を経ずに直接クエリツリーに追加されます。こうした種類の検索語に対して適用される変換は、大文字から小文字への変換だけです。

## 第3段階: 文書検索

ここでいう文書検索とは、一致する語句がインデックスに存在する文書を見つけることです。この段階は、例を使用するとよくわかります。まず、次のような単純なスキマを使用した `hotels` というインデックスを考えてみましょう。

JSON

```
{  
    "name": "hotels",  
    "fields": [  
        { "name": "id", "type": "Edm.String", "key": true, "searchable"::  
false },  
        { "name": "title", "type": "Edm.String", "searchable": true },  
        { "name": "description", "type": "Edm.String", "searchable": true }  
    ]  
}
```

さらに、このインデックスには、以下の4つの文書が追加されているとします。

JSON

```
{  
    "value": [  
        {  
            "id": "1",  
            "title": "Hotel Atman",  
            "description": "Spacious rooms, ocean view, walking distance to  
the beach."  
        },  
        {  
            "id": "2",  
            "title": "Beach Resort",  
            "description": "Located on the north shore of the island of  
Kauai. Ocean view."  
        },  
        {  
            "id": "3",  
            "title": "Playa Hotel",  
            "description": "Comfortable, air-conditioned rooms with ocean  
view."  
        }  
    ]  
}
```

```
    },
    {
        "id": "4",
        "title": "Ocean Retreat",
        "description": "Quiet and secluded"
    }
]
```

## 語句のインデックス作成方法

検索を理解するには、インデックス作成の基本をいくつかを把握しておくと役立ちます。保存の単位は転置インデックスで、検索可能なフィールドごとに 1 つ存在します。転置インデックス内には、全文書から抽出されたすべての語句を並べ替えたリストが存在します。それぞれの語句は、それが出現する一連の文書に対応付けられています（以下の例を参照）。

転置インデックスに含める語句を得るために、検索エンジンは、クエリの加工時と同様の字句解析を文書の内容に対して実行します。

1. "テキスト入力" はアナライザーに渡され、小文字への変換や句読点の削除など、アナライザーの構成に応じた処理が行われます。
2. "トークン" は字句解析の出力です。
3. "用語" はインデックスに追加されます。

検索語の体裁をインデックスに登録されている語句の体裁と合わせるために、通常は検索操作とインデックス作成操作に同じアナライザーが使用されますが、必ずしも同じである必要はありません。

### ① 注意

Azure Cognitive Search では、追加の `indexAnalyzer` および `searchAnalyzer` フィールド パラメーターを使用して、インデックス作成と検索に別々のアナライザーを指定することができます。指定しなかった場合、`analyzer` プロパティで設定されたアナライザーが、インデックス作成と検索の両方に使用されます。

## 文書サンプルの転置インデックス

もう一度先ほどの例を見てみましょう。`title` フィールドの転置インデックスは、次のようにになります。

任期	ドキュメント リスト
atman	1

任期	ドキュメント リスト
beach	2
hotel	1、3
ocean	4
playa	3
resort	3
retreat	4

**title** フィールドの場合、*hotel*だけが 2 つの文書 (1 と 3) に出現します。

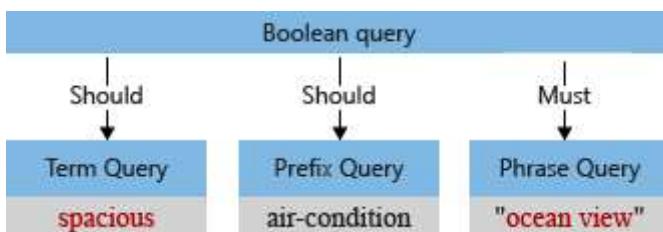
**description** フィールドのインデックスは次のようにになっています。

任期	ドキュメント リスト
air	3
and	4
beach	1
conditioned	3
comfortable	3
distance	1
island	2
kaua'i	2
located	2
north	2
ocean	1, 2, 3
of	2
on	2
通知の停止	4
会議室	1、3
secluded	4

任期	ドキュメント リスト
shore	2
spacious	1
the	1, 2
to	1
表示	1, 2, 3
walking	1
with	3

### インデックスが作成された語句に対する検索語の照合

上記の転置インデックスを踏まえてサンプル クエリに戻り、一致する文書がどのように検索されるかを見ていきましょう。最終的なクエリツリーは、次のようになっていましたことを思い出してください。



クエリの実行中、検索可能なフィールドに対して個々の検索が別々に実行されます。

- 単語検索 "spacious" と一致する文書は 1 件です (Hotel Atman)。
- プレフィックス検索 "air-condition\*" と一致する文書はありません。

これは、場合によっては開発者の混乱を招く動作です。 "air-conditioned" という語句はこの文書内に存在しますが、既定のアナライザーによって 2 つの単語に分割されています。部分的な語句を含むプレフィックス クエリは、解析されないことに注意してください。そのため、転置インデックスで、"air-condition" というプレフィックスを含む語句を検索しても見つかりません。

- フレーズ検索 "ocean view" では、"ocean" と "view" という 2 つの語句が検索され、元の文書内で両者が近いかどうかがチェックされます。文書 1、2、3 の description フィールドに、この検索との一致が見つかります。文書 4 の title には ocean という語が存在しますが、一致とは見なされないことに注意してください。検索対象は "ocean view" というフレーズであって、個々の単語ではありません。

## ① 注意

特定のフィールドを `searchFields` パラメーター (検索要求の例を参照) で指定しない限り、検索クエリは、Azure Cognitive Search インデックス内の検索可能なすべてのフィールドに対して個別に実行されます。選択したフィールドのいずれかで一致する文書が返されます。

全体として、この例のクエリの場合、一致する文書は 1、2、3 です。

## 第 4 段階: スコア付け

検索結果セット内のすべての文書には、関連度スコアが割り当てられます。関連度スコアの機能は、ユーザーからの問い合わせ (検索クエリ) に対して最適解となる文書に対し、相対的に高いランクを与えることです。このスコアは、一致した語句の統計学的な特性に基づいて計算されます。スコア付けの式の核となるのは [TF/IDF \(Term Frequency-Inverse Document Frequency\)](#) です。TF/IDF は、出現頻度の低い語句と高い語句を含んだ検索において、出現頻度の低い語句を含んだ結果に、より高いランクを与えます。たとえば、Wikipedia の記事をすべて含んだ架空のインデックスでは、*the president* というクエリに一致した文書のうち、*president* で一致した文書の方が *the* で一致した文書よりも関連性が高いと見なされます。

## スコア付けの例

冒頭のサンプル クエリで見つかった 3 つの文書を思い出してください。

```
search=Spacious, air-condition* +"Ocean view"
```

JSON

```
{
  "value": [
    {
      "@search.score": 0.25610128,
      "id": "1",
      "title": "Hotel Atman",
      "description": "Spacious rooms, ocean view, walking distance to the beach."
    },
    {
      "@search.score": 0.08951007,
      "id": "3",
      "title": "Ocean View Hotel",
      "description": "Spacious rooms, ocean view, walking distance to the beach."
    }
  ]
}
```

```
        "title": "Playa Hotel",
        "description": "Comfortable, air-conditioned rooms with ocean view."
    },
    {
        "@search.score": 0.05967338,
        "id": "2",
        "title": "Ocean Resort",
        "description": "Located on a cliff on the north shore of the island of Kauai. Ocean view."
    }
]
```

文書 1 は、クエリに対して最も高い関連度で一致しています。なぜなら、*spacious* という単語と *ocean view* という必須のフレーズの両方が *description* フィールドに出現するためです。その他の 2 つの文書は、*ocean view* しか一致していません。しかし文書 2 と文書 3 は、クエリに対して同じように一致しているにもかかわらず、関連度スコアが異なるのはなぜでしょうか。これは、スコア付けの式の構成要素が TF/IDF だけではないためです。この場合、文書 3 の方が、*description* が短いために、少しだけ高いスコアが割り当てられています。フィールドの長さやその他の要因が関連度スコアに与える影響については、[Lucene の実際に役立つスコア付けの式](#)に関するページを参照してください。

一部の検索の種類 (ワイルドカード、プレフィックス、正規表現) は、文書全体のスコアに対して常に一定のスコアをもたらします。これによって、ランクには影響を与えずに、クエリ拡張によって見つかった一致を結果に反映することができます。

このことが重要である理由を例で説明します。ワイルドカード検索 (プレフィックス検索を含む) は、本質的に解釈があいまいです。入力されるのは文字列の一部であり、まったく異なる、膨大な数の語句と一致する可能性があるからです ("tour\*" と入力した場合、"tours"、"tourettes"、"tourmaline" などとの一致が検出されます)。こうした結果の性質上、用語の相対的な重みを適切に推測することができません。そのため、ワイルドカード検索、プレフィックス検索、正規表現検索では、結果のスコア付けを行う際に、語句の出現頻度が無視されます。部分的な語句と完全な語句とを含んだ複数の構成要素から成る検索要求では、予期しない一致が偏重されないよう、部分的な入力から得られた結果については、定数スコアを割り当てたうえで反映されます。

## 関連性のチューニング

Azure Cognitive Search の関連度スコアは、次の 2 とおりの方法でチューニングできます。

1. **スコアリングプロファイル**: ランク付けされた結果リストにおいて、一連のルールに基づく重みを文書に与えます。このページの例では、*title* フィールドに一致

の見つかった文書の方が、`description` フィールドに一致の見つかった文書よりも関連性が高いと見なすことができます。加えて、仮にホテルごとの料金フィールドをインデックスに含めた場合、料金の低い方の文書に高い重みを与えることもできます。[スコアリング プロファイルを検索インデックスに追加する方法](#) の詳細について学習します。

2. **項目ブースト** (Full Lucene クエリ構文のみで使用可能): クエリツリーの任意の構成要素に適用できるブースト演算子 `^` が用意されています。このページの例では、`air-condition^*` というプレフィックスで検索する代わりに、`air-condition^2||air-condition^*` のように単語検索にブーストを適用することもできます。そうすれば、`air-condition` で完全一致する語句と、プレフィックスで一致する語句との両方を検索したうえで、完全一致の語句で一致した文書の方に、より高いランクを与えることができます。[クエリでの語句ブースト](#) の詳細について学習します。

## 分散されたインデックスにおけるスコア付け

Azure Cognitive Search のすべてのインデックスは自動的に複数のシャードに分割されます。これにより、Microsoft は、サービスのスケールアップまたはスケールダウンの間に、複数のノードにインデックスをすばやく分散することができます。検索要求は送信されると、各シャードに対して別々に送られます。その後、各シャードから得られた結果がマージされ、スコア順に並べ替えられます (他に並べ替えが定義されていない場合)。スコアリング関数は、シャード内のすべてのドキュメントで逆ドキュメント頻度に対して検索語頻度を重み付けするものであり、すべてのシャードで重み付けするものではないことを知ることが重要です。

つまり、まったく同じ文書でも、それらが異なるシャードに存在していれば、関連度スコアが異なる "可能性がある" ということです。さいわい、インデックス内の文書数が増えるにつれて語句の分布が平準化され、そのような差異は総じて消失します。文書がどのシャードに配置されるかを推測することは不可能です。しかし文書は、そのキーが変化しなければ、常に同じシャードに割り当てられます。

一般に、順序の不变性が重要な場合、文書を並べ替えるための特性として、文書のスコアはあまり適していません。たとえば、まったく同じスコアの文書が 2 つあるとして、同一クエリを繰り返し実行したときに、どちらの文書が上位にランク付けされるかを毎回確実に予測することができません。文書スコアは、検索結果群における特定の文書の相対的な関連度の高さを測る、おおよその目安としてのみ使用してください。

## まとめ

商用検索エンジンが多くの人々の支持を得たことで、プライベートデータに対するフルテキスト検索への期待が高まってきました。ほぼすべての検索について言えることですが、語句の綴りが間違っていたり不完全であったりしても自分の意図がきちんと反映されるほどの利便性を、私たちは検索エンジンに求めるようになっています。指定してもいいない同義語やほぼ同等の語句に基づいた検索結果が得られることさえ、当然のように感じてしまうほどです。

技術的な観点でいえばフルテキスト検索はきわめて複雑で、洗練された言語分析と検索語の加工(関連性の高い結果を得るために検索語を抽出、展開、変換する処理)への秩序立ったアプローチとが要求されます。こうした本質的な複雑さもあって、検索結果はさまざまな要因によって左右されます。フルテキスト検索のメカニズムをしっかりと理解しておけば、予期しない結果に対処しようとする際に、はっきりとその効果を実感できるでしょう。

この記事では、Azure Cognitive Search の観点からフルテキスト検索について詳しく見てきました。ここで身に付けた知識が、検索時に遭遇しやすい問題の原因や解決策を判断するうえでの一助となればさいわいです。

## 次のステップ

- サンプルインデックスを構築し、さまざまな検索を試してその結果を確認します。詳しい手順については、[ポータルでのインデックスの構築と照会](#)に関するページを参照してください。
- [Search Documents](#) の例に関するセクションや[単純なクエリ構文](#)で紹介されている他のクエリ構文をポータルの検索エクスプローラーで試します。
- 検索アプリケーションにおけるランク付けをチューニングする方法については、[スコアリング プロファイル](#)に関するページを参照してください。
- [言語に固有の字句解析器](#)を適用する方法について書かれた記事を参照します。
- 特定のフィールドに対して最小限の処理または特殊な処理を適用するための[カスタム アナライザー](#)を構成します。

## 関連項目

[Search Documents REST API](#)

[単純なクエリ構文](#)

[Full Lucene クエリ構文](#)

## 検索結果の処理方法

# Azure AI Search のベクター

[アーティクル] • 2024/04/09

ベクトル検索とは、コンテンツの数値表現に対するインデックス付けとクエリの実行をサポートする情報取得のアプローチです。コンテンツはプレーンテキストではなく数値であるため、照合はクエリ ベクトルに最も類似したベクトルに基づいて行われます。これにより、次のシナリオでの照合が可能になります。

- 意味的または概念的な類似性 ("dog" と "canine" は概念的には似ているが言語的には異なる)
- 多言語コンテンツ (英語では "dog"、ドイツ語では "hund")
- 複数のコンテンツ タイプ (プレーンテキストの "dog" と画像ファイル内の犬の写真)

この記事では、Azure AI 検索でのベクトルの概要について説明します。また、他の Azure サービスとの統合についても説明し、ベクトル検索の開発に関する用語と概念についても説明します。

最初にこの記事を読んで基礎知識を得ることをお勧めしますが、それはかまわないのですぐに使いたいという方は、これらの手順を実行してください。

- ✓ インデックス用の埋め込みを提供するか、インデクサー パイプラインで埋め込み (プレビュー) を生成する
- ✓ ベクトルインデックスを作成する
- ✓ ベクトルクエリの実行

ベクトルクイックスタートまたは GitHub のコード サンプル[を](#)使用して開始することもできます。

## ベクトル検索をサポートできるシナリオ

ベクトル検索には次のようなシナリオがあります。

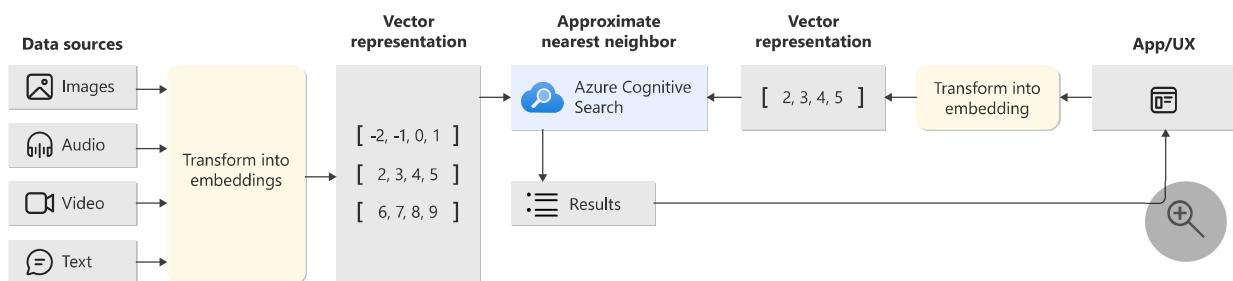
- 類似性検索。OpenAI 埋め込みなどの埋め込みモデルや SBERT などのオープンソース モデルを使用してテキストをエンコードし、やはりベクトルとしてエンコードされたクエリを使用してドキュメントを取得します。
- さまざまなコンテンツ タイプ (マルチモーダル) で検索します。画像とテキストをマルチモーダル埋め込み (たとえば、Azure OpenAI の OpenAI CLIP[や GPT-4 Turbo with Vision](#) を使用) を使用してエンコードし、両方のコンテンツ タイプのベクトルで構成される埋め込みスペースをクエリします。

- **ハイブリッド検索。** Azure AI 検索のハイブリッド検索では、同じ要求でベクトルおよびキーワード クエリの実行を参照します。ベクトルサポートはフィールドレベルで実装されており、ベクトルフィールドと検索可能なテキストフィールドの両方を含むインデックスがあります。クエリは並列で実行され、結果は1つの応答にマージされます。必要に応じて、[セマンティックランク付け](#)を追加して、Bing を動作させているのと同じ言語モデルを使用して、L2 の再ランク付けによって精度をさらに高めます。
- **多言語検索。** 複数の言語でトレーニングされたモデルとチャット モデルを埋め込んで、ユーザーの母国語での検索エクスペリエンスを提供できます。翻訳をより詳細に制御する必要がある場合は、ハイブリッド検索シナリオで Azure AI Search が非ベクトルコンテンツに対して提供する[多言語機能](#)を追加できます。
- **フィルター選択されたベクトル検索。** クエリ要求にはベクトルクエリと[フィルタ一式](#)を含めることができます。フィルターはテキストフィールドと数値フィールドに適用され、メタデータフィルターに役立ち、フィルター条件に基づいて検索結果を含めたり除外したりするのに役立ちます。ベクトルフィールド自体はフィルター処理できませんが、フィルター可能なテキストフィールドまたは数値フィールドを設定できます。検索エンジンは、ベクトルクエリの実行前または実行後にフィルターを処理できます。
- **ベクトルデータベース。** Azure AI Search には、クエリを実行するデータが格納されます。長期メモリやナレッジベース、あるいは[取得拡張生成 \(RAG\) アーキテクチャ](#)やベクトルを使用するあらゆるアプリケーションの基礎データが必要な場合は、[純粋なベクトルストア](#)として使用します。

## Azure AI Searchでのベクトル検索のしくみ

ベクトルのサポートには、検索インデックスからのベクトル埋め込みのインデックス作成、格納、クエリが含まれます。

次の図は、ベクトル検索のインデックス作成とクエリのワークフローを示しています。



インデックス作成側では、Azure AI Search はベクトル埋め込みを受け取り、[ニアレストネイバーアルゴリズム](#)を使用して、同様のベクトルをインデックス内の近い場所に

配置します。 内部では、各ベクトルフィールドのベクトルインデックスが作成されます。

ソースコンテンツから埋め込みを Azure AI Search に取り込む方法は、アプローチとプレビュー機能が使用できるかどうかによって異なります。 OpenAI、Azure OpenAI、および任意の数のプロバイダーのモデルを使用して、テキスト、画像、モデルでサポートされているその他のコンテンツ タイプなど、さまざまなソースコンテンツに対して埋め込みをベクトル化または生成できます。その後、事前にベクトル化したコンテンツをベクトルストアのベクトルフィールドにプッシュすることができます。これが一般公開されているアプローチです。プレビュー機能を使用できる場合、Azure AI Search はインデクサー パイプラインで統合されたデータ チャンクとベクトル化を提供します。リソース (エンドポイントと Azure OpenAI への接続情報) は引き続き提供されますが、Azure AI Search はすべての呼び出しを行い、移行を処理します。

クエリ側では、クライアントアプリケーションで、通常はプロンプトワークフローを使用して、ユーザーからクエリ入力を収集します。その後、入力をベクトルに変換するエンコード手順を追加し、Azure AI Search 上のインデックスにベクトル クエリを送信して類似性検索を行うことができます。インデックス作成と同様に、統合ベクトル化 (プレビュー) をデプロイして、質問をベクトルに変換できます。どちらの方法でも、Azure AI Search では、要求された `k` ニアレスト ネイバー (kNN) を含むドキュメントが結果に返されます。

Azure AI Search は、ベクトル検索とキーワード検索を並行して実行するハイブリッド シナリオをサポートしており、統合された結果セットを返しますが、これはしばしば、ベクトル検索やキーワード検索のみよりも優れた結果を提供します。ハイブリッドの場合、ベクトルコンテンツと非ベクトルコンテンツは、並列して実行されるクエリに対して、同じインデックスに取り込まれます。

## 可用性と料金

ベクトル検索は、すべてのリージョンのすべての Azure AI Search レベルの一部として追加料金なしで利用できます。

2024 年 4 月 3 日以降に作成された新しいサービスでは、ベクトルインデックスのためにより高いクォータをサポートしています。

ベクトル検索は次で利用できます。

- データのインポートとベクトル化ウィザードを使用した Azure portal
- Azure REST API、バージョン 2023-11-01
- .NET、Python、JavaScript 用の Azure SDK
- Azure AI Studio や Azure OpenAI Studio などその他の Azure オファリング。

## ① 注意

2019年1月1日より前に作成された一部の古い検索サービスは、ベクトルワーカロードをサポートしないインフラストラクチャにデプロイされています。ベクトルフィールドをスキーマに追加しようとしてエラーが表示された場合、それはサービスが古いためです。このような場合は、ベクトル機能を試すために新しい検索サービスを作成する必要があります。

## Azure の統合と関連サービス

Azure AI Search は、Azure AI プラットフォーム全体で深く統合されています。次の表に、ベクトルワーカロードで役立ついくつかの要素を示します。

[+] テーブルを展開する

Product	統合
Azure OpenAI Studio	データプレイグラウンドとのチャットで、[独自のデータを追加する] は、データの基盤と会話型検索のために Azure AI Search を使用します。これは、データとチャットするための最も簡単かつ高速なアプローチです。
Azure OpenAI	Azure OpenAI には埋め込みモデルとチャットモデルが用意されています。デモとサンプルでは、 <a href="#">text-embedding-ada-002</a> を対象とします。テキスト用の埋め込みを生成するには、Azure OpenAI をお勧めします。
Azure AI Services	<a href="#">Image Retrieval Vectorize Image API (プレビュー)</a> では、画像コンテンツのベクトル化がサポートされます。画像用の埋め込みを生成するには、この API をお勧めします。
Azure データプラットフォーム: Azure Blob Storage、Azure Cosmos DB	インデクサーを使用してデータインジェストを自動化し、 <a href="#">統合ベクトル化 (プレビュー)</a> を使用して埋め込みを生成できます。Azure AI Search では、Azure Blob インデクサーと <a href="#">Azure Cosmos DB for NoSQL インデクサー</a> の 2 つのデータソースからベクトルデータのインデックスを自動的に作成できます。詳細については、「 <a href="#">検索インデックスにベクトルフィールドを追加する</a> 」を参照してください。

これは、[LangChain](#) などのオープンソースフレームワークでも一般的に使用されています。

## ベクトル検索の概念

ベクトルを初めて使用する場合、このセクションではいくつかの主要な概念について説明します。

# ベクトル検索について

ベクトル検索は、ドキュメントとクエリがプレーン テキストではなくベクトルとして表現される場合の情報取得の方法です。ベクトル検索では、機械学習モデルがソース入力（テキスト、画像、その他のコンテンツ）のベクトル表現を生成します。コンテンツの数学表現を使用することによって、検索シナリオの共通基盤が提供されます。すべてがベクトルであれば、関連する元のコンテンツがクエリとは異なるメディアや言語であっても、クエリはベクトル空間で一致するものを見つけることができます。

## ベクトル検索を使用する理由

検索可能なコンテンツがベクトルとして表されると、クエリは類似するコンテンツ内の近い一致を見つけることができます。ベクトル生成に使用される埋め込みモデルは、どの単語と概念が類似しているかを認識し、結果のベクトルを埋め込み空間内で近くに配置します。たとえば、"クラウド" と "霧" に関するベクトル化されたソース ドキュメントは、意味的に類似しているため、構文上的一致ではない場合も "霧" に関するクエリで表示される可能性が高くなります。

## 埋め込みベクトル化

"埋め込み" は、テキストのセマンティックな意味や画像などの他のコンテンツの表現を読み取る機械学習モデルによって作成された、コンテンツまたはクエリの特定の種類のベクトル表現です。自然言語機械学習モデルは、単語間のパターンや関係を識別するために、大量のデータでトレーニングされます。トレーニング中に、"エンコーダー" と呼ばれる中間ステップで、入力を実数のベクトルとして表現する方法を学習します。トレーニングが完了すると、中間ベクトル表現がモデルの出力になるように、これらの言語モデルを変更できます。結果として得られる埋め込みは高次元ベクトルであり、[埋め込みの概要 \(Azure OpenAI\)](#) に関する記事で説明されているように、同じような意味を持つ単語がベクトル空間で互いに近くになります。

関連する情報の取得におけるベクトル検索の有効性は、ドキュメントとクエリの意味を結果のベクトルに抽出する埋め込みモデルの有効性に依存します。最適なモデルは、それらが代表するデータの種類によって適切にトレーニングされています。Azure OpenAI text-embedding-ada-002 などの既存のモデルを評価したり、問題領域で直接トレーニングされた独自のモデルを使用したり、汎用モデルを微調整したりできます。Azure AI Search では、選ぶモデルに制約が課されないため、データに最適なものを選んでください。

ベクトル検索に対して効果的な埋め込みを作成するには、入力サイズの制限を考慮することが重要です。埋め込みを生成する前に、[データをチャンクするためのガイドライン](#)

ンに従うことをお勧めします。このベストプラクティスのおかげで、埋め込みによって関連情報が正確に読み取られ、より効率的なベクトル検索が可能になります。

## 埋め込み空間とは

"埋め込み空間"は、ベクトルクエリのコーパスです。検索インデックス内では、埋め込み空間は、同じ埋め込みモデルからの埋め込み値が設定されているすべてのベクトルフィールドです。機械学習モデルでは、個々の単語、語句、またはドキュメント(自然言語処理の場合)、画像、またはその他の形式のデータを、高次元空間の座標を表す実数のベクトルで構成される表現にマッピングすることで、埋め込み空間を作成します。この埋め込みスペースでは、類似項目は近くに配置され、異なる項目は離れた場所に配置されます。

たとえば、さまざまな種類の犬について説明するドキュメントは、埋め込み空間で互いに近くに集められます。猫に関するドキュメントは互いに近くに集まりますが、犬のクラスターから遠く離れており、それでも動物としては近くになります。クラウドコンピューティングなどの異なる概念は、はるかに遠く離れています。実際には、これらの埋め込み空間は抽象的で、人間が解釈できる明確に定義された意味はありませんが、中核となる概念は同じです。

## ニアレストネイバー検索

ベクトル検索では、検索エンジンは埋め込みスペース内のベクトルをスキャンして、クエリベクトルに最も近いベクトルを識別します。この手法は"ニアレストネイバー検索"と呼ばれます。[ニアレストネイバー](#)は、項目間の類似性を定量化するのに役立ちます。ベクトルの類似性が高い場合は、元のデータも同様であることを示します。高速なニアレストネイバー検索を容易にするために、検索エンジンでは最適化を実行するか、データ構造およびデータパーティション分割を使用して検索領域を削減します。各ベクトル検索アルゴリズムは、最小待機時間、最大スループット、再現率、メモリを最適化する際に、ニアレストネイバーの問題をさまざまな方法で解決します。類似性を計算するために、類似性メトリックでは距離を計算するためのメカニズムを提供します。

Azure AI Search では現在、次のアルゴリズムがサポートされています。

- Hierarchical Navigable Small World (HNSW): HNSW は、データ分散が不明であるか、頻繁に変更される可能性がある、高いリコールと待機時間の短い用途に最適化された主要な ANN アルゴリズムです。高次元のデータ ポイントを階層グラフ構造に整理することで、高速でスケーラブルな類似性検索を可能にしながら、検索精度と計算コストのトレードオフを調整できます。このアルゴリズムでは、高速ランダムアクセスのためにすべてのデータ ポイントがメモリ内に存在する必要

があるため、このアルゴリズムではベクトルインデックス サイズのクオータが使用されます。

- 完全な K ニアレストネイバー (KNN): クエリベクトルとすべてのデータ ポイントの間の距離を計算します。計算負荷が高いので、小規模なデータセットに最適です。このアルゴリズムではデータ ポイントの高速ランダム アクセスが不要なため、このアルゴリズムではベクトルインデックス サイズのクオータが使用されません。ただし、このアルゴリズムではニアレストネイバーのグローバルセットが提供されます。

インデックス定義内で 1 つ以上のアルゴリズムを指定し、ベクトルフィールドごとに使用するアルゴリズムを指定できます。

- インデックスとフィールドにアルゴリズムを指定するベクトルストアを作成します。
- 完全な KNN の場合、[2023-11-01](#)、[2023-10-01-Preview](#)、または REST API バージョンを対象とする Azure SDK ベータ ライブラリを使用します。

インデックスの作成時にインデックスの初期化に使用されるアルゴリズム パラメーターは不变であり、インデックスの作成後に変更することはできません。ただし、クエリ時間の特性 (`efSearch`) に影響を与えるパラメーターは変更することができます。

さらに、HNSW アルゴリズムを指定するフィールドは、クエリ要求パラメーター `"exhaustive": true` を使用した完全な KNN 検索もサポートします。ただし、その逆は当てはまりません。`exhaustiveKnn` に対してフィールドがインデックス付けされている場合、効率的な検索'を可能にする追加のデータ構造が存在しないため、クエリで HNSW を使用することはできません。

## 近似ニアレストネイバー

近似ニアレストネイバー検索 (ANN) は、ベクトル空間で一致を検索するためのアルゴリズムの種類です。この種類のアルゴリズムでは、検索空間を大幅に削減してクエリ処理を高速化するため、さまざまなデータ構造またはデータパーティション分割方法が採用されます。

ANN アルゴリズムでは、精度がいくらか犠牲になりますが、近似ニアレストネイバーをスケーラブルかつ迅速に取得できるため、最新の情報取得の用途で効率と精度のバランスを取るのに最適です。アルゴリズムのパラメーターを調整して、検索用途のリコール、待機時間、メモリ、ディスク フットプリントの要件を微調整できます。

Azure AI Search では、ANN アルゴリズムに HNSW が使用されます。

## 次のステップ

- クイックスタートを試す
- ベクトルインデックス作成の詳細を確認する
- ベクトルクエリの詳細を確認する
- Azure Cognitive Search と LangChain: 拡張ベクトル検索機能のシームレスな統合 ↗

# Azure AI Search でベクトルやフルテキストを使用したハイブリッド検索

[アーティクル] • 2024/01/31

ハイブリッド検索は、フルテキストとベクトル クエリを組み合わせたものです。検索可能なプレーンテキストコンテンツと生成された埋め込み両方を含む検索インデックスに対してクエリを実行します。 クエリの目的として、ハイブリッド検索とは次のようなものです。

- `search` および `vectors` クエリ パラメータ両方を含む単一のクエリ要求です。
- 並行して実行されます。
- クエリ応答のマージされた結果の場合、[Reciprocal Rank Fusion \(RRF\)](#) を使用してスコアリングされます。

この記事では、ハイブリッド検索のコンセプト、利点、制限について説明します。 この[埋め込みビデオ](#)で、ハイブリッド検索が高品質のチャットスタイルアプリやコパイロットアプリにどのように貢献するかの説明と短いデモをご覧ください。

## ハイブリッド検索が機能するしくみ

Azure AI Search では、埋め込みを含むベクター フィールドをテキスト フィールドと数値フィールドと共に使用できるため、並列で実行されるハイブリッド クエリを作成できます。ハイブリッド クエリでは、単一の検索要求で、フィルター処理やファセット処理、並べ替え、スコアリング プロファイル、[セマンティック ランク付け](#)といった既存の機能を利用できます。

ハイブリッド検索では、BM25 や HNSW などの異なるランク付け機能を利用し、フルテキストおよびベクトル クエリの両方の結果を組み合わせます。[逆ランク 融合 \(RRF\)](#) アルゴリズムによって結果がマージされます。 クエリ応答は、RRF を使用して各クエリから最も関連性の高い一致を選択する結果セットを 1 つだけ提供します。

## ハイブリッド クエリの構造

ハイブリッド検索は、プレーン テキストと数値、地理空間検索の地理座標、テキストのチャンクの数学的表現のためのベクトルなど、さまざまな[データ型](#)のフィールドを含む検索インデックスを持つことを前提とします。 ベクトル クエリにより、オートコンプリートや検索候補などのクライアント側のインタラクションを除き、Azure AI Search のほぼすべてのクエリ機能を使用できます。

典型的なハイブリッド クエリは、次のようにになります(簡潔にするためベクトルを省略しています)。

#### HTTP

```
POST https://{{searchServiceName}}.search.windows.net/indexes/hotels-vector-quickstart/docs/search?api-version=2023-11-01
    content-type: application/JSON
{
    "count": true,
    "search": "historic hotel walk to restaurants and shopping",
    "select": "HotelId, HotelName, Category, Description, Address/City, Address/StateProvince",
    "filter": "geo.distance(Location, geography'POINT(-77.03241 38.90166)') le 300",
    "facets": [ "Address/StateProvince"],
    "vectors": [
        {
            "value": [ <array of embeddings> ]
            "k": 7,
            "fields": "DescriptionVector"
        },
        {
            "value": [ <array of embeddings> ]
            "k": 7,
            "fields": "Description_frVector"
        }
    ],
    "queryType": "semantic",
    "queryLanguage": "en-us",
    "semanticConfiguration": "my-semantic-config"
}
```

重要なポイントは次のとおりです。

- `search` はフルテキスト検索クエリを指定します。
- `vectors` はベクトルクエリです。複数設定して複数のベクトルフィールドを対象とすることができます。埋め込みスペースに多言語コンテンツが含まれる場合、言語アナライザーや翻訳を介さずに、ベクトルクエリで一致を検出できます。
- `select` は結果で返すフィールドを指定します。人間が判読できるテキストフィールドを指定することもできます。
- `filters` は地理空間の検索、またはその他の包含条件や除外条件(駐車場を含めるかどうかなど)を指定できます。この例における地理空間のクエリでは、ワシントン D.C. から半径 300 キロ以内にあるホテルを検出します。
- `facets` はハイブリッド クエリで返された結果のファセットバケットの計算に使用できます。

- `queryType=semantic` はセマンティック ランク付けを呼び出します。機械読解を適用し、より関連性の高い検索結果を表示させます。

フィルターとファセットは、フルテキスト検索に使用した逆インデックスおよびベクトル検索に使用したベクトルインデックスとは異なるインデックス内のデータ構造を対象とします。そのため、フィルターとファセット処理が実行されると、検索エンジンの応答では、ハイブリッド検索結果に操作上の結果が適用されます。

このクエリには `orderby` がないことがわかります。明示的な並べ替え順序は、関連性でランク付けされた結果をオーバーライドするため、類似性と BM25 の関連性が必要な場合は、クエリで並べ替えを省略します。

上記のクエリに対する応答は、この例のようになります。

HTTP

```
{  
    "@odata.count": 3,  
    "@search.facets": {  
        "Address/StateProvince": [  
            {  
                "count": 1,  
                "value": "NY"  
            },  
            {  
                "count": 1,  
                "value": "VA"  
            }  
        ]  
    },  
    "value": [  
        {  
            "@search.score": 0.03333333507180214,  
            "@search.rerankerScore": 2.5229012966156006,  
            "HotelId": "49",  
            "HotelName": "Old Carrabelle Hotel",  
            "Description": "Spacious rooms, glamorous suites and residences, rooftop pool, walking access to shopping, dining, entertainment and the city center.",  
            "Category": "Luxury",  
            "Address": {  
                "City": "Arlington",  
                "StateProvince": "VA"  
            }  
        },  
        {  
            "@search.score": 0.032522473484277725,  
            "@search.rerankerScore": 2.111117362976074,  
            "HotelId": "48",  
            "HotelName": "Nordick's Motel",  
            "Description": "Only 90 miles (about 2 hours) from the nation's
```

```
capital and nearby most everything the historic valley has to offer.  
Hiking? Wine Tasting? Exploring the caverns? It's all nearby and we have  
specially priced packages to help make our B&B your home base for fun while  
visiting the valley.",  
        "Category": "Boutique",  
        "Address": {  
            "City": "Washington D.C.",  
            "StateProvince": null  
        }  
    }  
}  
]
```

## ハイブリッド検索を選択する理由

ハイブリッド検索は、ベクトル検索とキーワード検索の長所を組み合わせたものです。ベクトル検索のメリットは、逆インデックスにキーワードの一致がない場合でも、検索クエリと概念的に似た情報が検索されることです。キーワード検索やフルテキスト検索のメリットは、精度の高さと、最初の結果の品質を向上させるセマンティックランク付けを適用できる機能です。製品コードや、極めて特殊な専門用語、日付、人の名前に対するクエリなど、一部のシナリオでは、完全一致を識別できるため、キーワード検索を使用すると、より良いパフォーマンスを発揮します。

実際のデータセットとベンチマークデータセットに対するベンチマークテストでは、セマンティックランク付けを使用したハイブリッド検索の場合、検索の関連性に大きな利点があることが示されています。

次のビデオでは、有用な AI 応答を生成するための最適なグラウンディングデータが、ハイブリッド検索でどのように提供されるかについて説明します。

<https://www.youtube-nocookie.com/embed/Xwx1DJ0OqCk> ↗

## 関連項目

[ハイブリッド検索とランク付けでベクトル検索の性能を上回る \(技術ブログ\)](#) ↗

# Azure AI Search での取得拡張生成 (RAG)

[アーティクル] • 2024/04/22

取得拡張生成 (RAG) は、グラウンディングデータを提供する情報取得システムを追加することで、ChatGPT などの大規模言語モデル (LLM) の機能を拡張するアーキテクチャです。情報取得システムを追加すると、応答を作成するときに LLM によって使用されるグラウンディングデータを制御できます。エンタープライズソリューションの場合、RAG アーキテクチャは、ベクトル化されたドキュメントや画像、およびそのコンテンツの埋め込みモデルがある場合は、その他のデータ形式から取得された "エンタープライズコンテンツ" に生成 AI を制限できることを意味します。

どの情報取得システムを使用するかによって LLM への入力が決定されるため、この決定は重要です。情報取得システムは、次の情報を提供する必要があります。

- 必要な頻度で、すべてのコンテンツに対して、大規模に読み込んで更新するインデックス作成戦略。
- クエリ機能と関連性のチューニング。システムは、関連する結果を、LLM 入力のトークンの長さの要件を満たすのに必要な短い形式で返す必要があります。
- データと操作の両方のセキュリティ、グローバル展開、信頼性。
- インデックス作成用の埋め込みモデル、および取得のためのチャット モデルまたは言語理解モデルとの統合。

Azure AI Search は、RAG アーキテクチャにおける [情報取得のための実証済みのソリューション](#) です。Azure クラウドのインフラストラクチャとセキュリティを備えたインデックス作成とクエリ機能を提供します。コードやその他のコンポーネントを使用して、財産的価値のあるコンテンツに対する生成 AI のすべての要素を含む包括的な RAG ソリューションを設計できます。

## ① 注意

Copilot と RAG の概念は初めてですか? 「[ベクトル検索と、生成 AI アプリの最新の取得](#)」をご覧ください。

## Azure AI Search を使用した RAG へのアプローチ

Microsoft は、RAG ソリューションで Azure AI Search を使用するためのいくつかの組み込み実装を用意しています。

- Azure AI Studio。ベクトルインデックスと取得拡張を使用します。
- Azure OpenAI Studio。ベクトルの有無にかかわらず、検索インデックスを使用します。
- Azure Machine Learning。プロンプト フローでベクトルストアとして検索インデックスを使用します。

キュレーションされたアプローチを使用すると、簡単に作業を開始できますが、アーキテクチャをより詳細に制御するには、カスタム ソリューションが必要です。これらのテンプレートでは、以下でエンドツー エンドのソリューションが作成されます。

- Python ↗
- .NET ↗
- JavaScript ↗
- Java ↗

この記事の残りの部分では、Azure AI Search がカスタム RAG ソリューションにどのように適合するかについて説明します。

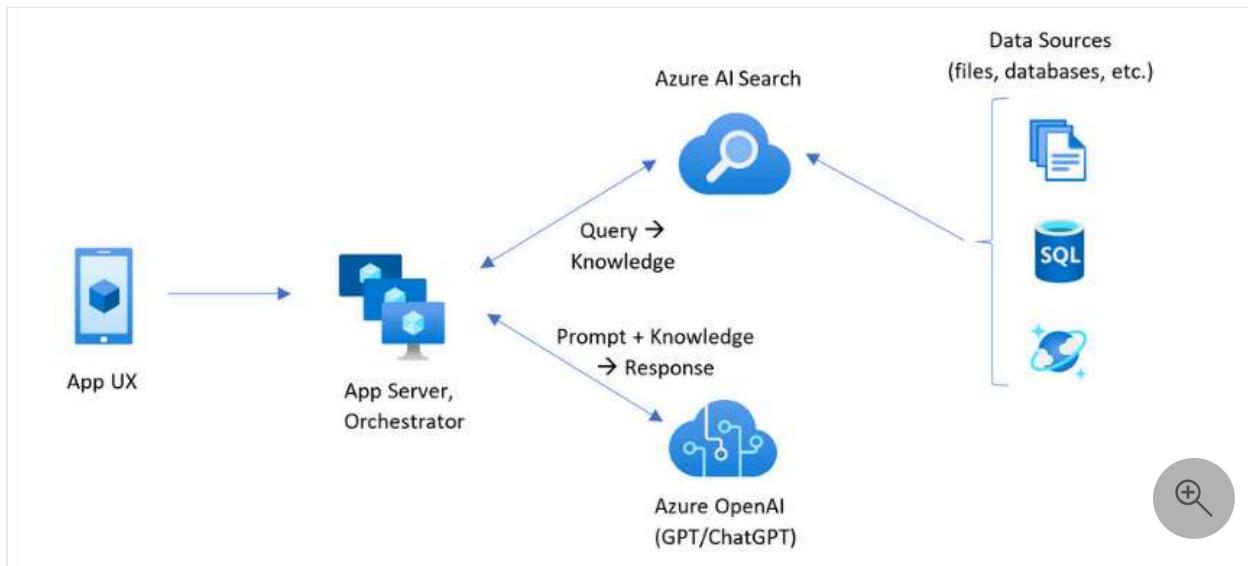
## Azure AI Search のカスタム RAG パターン

パターンの大まかな概要は次のとおりです。

- ユーザーの質問または要求 (プロンプト) から始めます。
- Azure AI Search に送信して、関連情報を見つけます。
- 上位の検索結果を LLM に送信します。
- LLM の自然言語理解と推論機能を使用して、最初のプロンプトに対する応答を生成します。

Azure AI Search が LLM プロンプトに入力を提供しますが、モデルのトレーニングはしません。RAG アーキテクチャでは、追加のトレーニングはありません。LLM はパブリックデータを使用して事前トレーニングされますが、取得コンポーネントからの情報によって拡張された応答を生成します。

Azure AI Search を含む RAG パターンには、次の図に示す要素があります。



- ユーザー エクスペリエンスのためのアプリ UX (Web アプリ)
- アプリ サーバーまたはオーケストレーター (統合と調整レイヤー)
- Azure AI Search (情報取得システム)
- Azure OpenAI (生成 AI 用の LLM)

Web アプリはユーザー エクスペリエンスを提供し、プレゼンテーション、コンテキスト、ユーザー操作を提供します。ユーザーからの質問またはプロンプトは、ここから始まります。入力は統合レイヤーを通過します。最初に情報を取得して検索結果を取得しますが、さらに LLM に移動してコンテキストと意図を設定します。

アプリ サーバーまたはオーケストレーターは、情報の取得と LLM の間のハンドオフを調整する統合コードです。1つのオプションは、[LangChain](#) を使用してワークフローを調整することです。LangChain は [Azure AI Search](#) と統合されるため、Azure AI Search を [取得コンポーネント](#) としてワークフローに簡単に含めることができます。セマンティック カーネルも別のオプションです。

情報取得システムは、検索可能なインデックス、クエリ ロジック、ペイロード (クエリ 応答) を提供します。検索インデックスには、ベクトルまたはベクトル以外のコンテンツを含めることができます。ほとんどのサンプルとデモにはベクトル フィールドが含まれていますが、必須ではありません。クエリは、キーワード (または用語) とベクトル クエリを処理できる Azure AI Search の既存の検索エンジンを使用して実行されます。インデックスは、定義したスキーマに基づいて事前に作成され、ファイル、データベース、またはストレージからソース化されたコンテンツと共に読み込まれます。

LLM は、元のプロンプトに加えて、Azure AI Search からの結果を受け取ります。LLM は結果を分析し、応答を作成します。LLM が ChatGPT の場合、ユーザーの対話は会話のやり取りである可能性があります。Davinci を使用している場合、プロンプトは完全に構成された回答である可能性があります。Azure ソリューションでは Azure OpenAI が使用される可能性が最も高いですが、この特定のサービスに対するハードな依存関係はありません。

Azure AI 検索では、プロンプト フローやチャットの保持のためのネイティブ LLM 統合は提供されないため、オーケストレーションと状態を処理するコードを記述する必要があります。完全なソリューションに必要なブループリントについては、デモ ソース ([Azure-Samples/azure-search-openai-demo](#)) を確認できます。また、LLM と統合する RAG ベースの Azure AI 検索ソリューションを作成するには、Azure AI Studio または Azure OpenAI Studio を使用することをお勧めします。

## Azure AI Search の検索可能なコンテンツ

Azure AI Search では、検索可能なすべてのコンテンツは、検索サービスでホストされている検索インデックスに格納されます。検索インデックスは、応答時間がミリ秒レベルの高速なクエリを実現するために設計されているため、内部データ構造はその目標をサポートするために存在します。そのため、検索インデックスにはインデックス付きコンテンツが保存されます。コンテンツ ファイル全体 (PDF 全体や画像など) は保存されません。内部では、データ構造には [トークン化されたテキスト](#) の逆インデックス、埋め込み用のベクトルインデックス、逐語的一致が必要な場合 (フィルター、あいまい検索、正規表現クエリなど) の変更されていないテキストが含まれます。

RAG ソリューションのデータを設定するときは、Azure AI Search でインデックスを作成して読み込む機能を使用します。インデックスには、ソース コンテンツを複製または表すフィールドが含まれます。インデックス フィールドは単純な転送 (ソース ドキュメントのタイトルまたは説明が検索インデックスのタイトルまたは説明になる) であるか、画像の表現またはテキストの説明を生成するベクトル化やスキル処理などの外部プロセスの出力を含む場合があります。

検索するコンテンツの種類をご存知のことと思われる所以、各コンテンツ タイプに適用できるインデックス作成機能を検討します。

 [テーブルを開く](#)

コンテンツ タイプ	付けられたインデックス	機能
text	トークン、変更されていないテキスト	インデクサーは、Azure Storage や Cosmos DB などの他の Azure リソースからプレーンテキストをプルできます。インデックスに <a href="#">任意の JSON コンテンツをプッシュ</a> することもできます。処理中のテキストを変更するには、 <a href="#">アナライザー</a> と <a href="#">ノーマライザー</a> を使用して、インデックス作成中に字句処理を追加します。 <a href="#">同意語マップ</a> は、クエリで使用される可能性のある用語がソース ドキュメントにない場合に便利です。
text	ベクトル <sup>1</sup>	テキストをチャンクして外部でベクトル化し、インデックスに <a href="#">ベクトルフィールド</a> としてインデックスを付けることができます。

コンテンツ	付けられたインデックス	機能
image	トークン、変更されていないテキスト <sup>2</sup>	OCR と画像解析のスキルでは、テキスト認識やイメージ特性のために画像を処理できます。 画像情報は検索可能なテキストに変換され、インデックスに追加されます。 エンティティにはインデクサーの要件があります。
image	ベクトル <sup>1</sup>	画像は、画像コンテンツを数学的に表現するために外部でベクトル化し、インデックスにベクトルフィールドとしてインデックスを付けることができます。 OpenAI CLIP <sup>3</sup> などのオープンソースモデルを使用して、同じ埋め込み空間内のテキストと画像をベクトル化できます。

<sup>1</sup>ベクトルサポートの一般提供機能では、データ チャンクとベクトル化のために他のライブラリまたはモデルを呼び出す必要があります。しかし、垂直統合 (プレビュー) にはこれらの手順が埋め込まれています。両方のアプローチを示すコード サンプルについては、[azure-search-vector リポジトリ](#) を参照してください。

<sup>2</sup>スキルは AI エンリッチメントの組み込みサポートです。 OCR と画像分析の場合、インデックス作成パイプラインは Azure AI Vision API の内部呼び出しを行います。これらのスキルは、抽出された画像を処理のために Azure AI に渡し、Azure AI Search によってインデックス付けされたテキストとして出力を受け取ります。

ベクトルは、異なるコンテンツ (複数のファイル形式と言語) に最適な設備を提供します。これは、コンテンツが数学表現で汎用的に表現されるためです。また、ベクトルでは類似性検索もサポートされています。つまり、ベクトルクエリに最も似た座標で照合します。トークン化された用語で照合するキーワード検索 (または用語検索) と比較すると、類似性検索の方が微妙です。コンテンツまたはクエリにあいまいな点や解釈の要件がある場合は、より適切な選択肢です。

## Azure AI Search でのコンテンツの取得

データが検索インデックスに格納されたら、Azure AI Search のクエリ機能を使用してコンテンツを取得します。

RAG 以外のパターンでは、クエリは検索クライアントからラウンド トリップを行います。クエリが送信され、検索エンジンで実行され、応答がクライアント アプリケーションに返されます。応答 (検索結果) は、インデックス内で見つかった逐語的なコンテンツのみで構成されます。

RAG パターンでは、検索エンジンと LLM の間でクエリと応答が調整されます。ユーザーの質問またはクエリは、検索エンジンと LLM の両方にプロンプトとして転送されま

す。検索結果は検索エンジンから戻り、LLM にリダイレクトされます。ユーザーに返される応答は生成 AI で、LLM からの合計または回答のどちらかです。

Azure AI Search には、新しい回答を構成するクエリの種類はありません (セマンティック検索やベクトル検索でさえも)。LLM だけが生成 AI を提供します。クエリの作成に使用される Azure AI Search の機能を次に示します。

#### テーブルを展開する

クエリ機能	目的	使用する理由
単純または完全な Lucene 構文	テキストと非ベクトル数値コンテンツに対するクエリ実行	フルテキスト検索は、類似一致ではなく、完全一致に最適です。フルテキスト検索クエリは、BM25 アルゴリズムを使用してランク付けされ、スコアリングプロファイルによる関連性チューニングをサポートします。また、フィルターとファセットもサポートされています。
フィルターとファセット	テキストまたは数値 (非ベクトル) フィールドにのみ適用されます。包含条件または除外条件に基づいて検索対象領域を減らします。	クエリに精度を追加します。
セマンティックランク付け	セマンティックモデルを使用して BM25 結果セットを再ランク付けします。LLM 入力として役立つ短い形式のキャプションと回答を生成します。	スコアリングプロファイルよりも簡単で、コンテンツによっては、関連性チューニングのためのより信頼性の高い手法です。
ベクトル検索	クエリ文字列が 1 つ以上のベクトルである類似性検索のベクトルフィールドに対するクエリ実行。	ベクトルは、あらゆる種類のコンテンツを任意の言語で表すことができます。
ハイブリッド検索	上記のクエリ手法の一部またはすべてを組み合わせます。ベクトルおよび非ベクトルクエリは並列で実行され、統合された結果セットで返されます。	ハイブリッド クエリを使用した場合、精度とリコールにおけるメリットが最も多くなります。

## クエリ応答を構造化する

クエリの応答は LLM に入力を提供するため、検索結果の品質は成功に不可欠です。結果は表形式の行セットです。結果の構成や構造は次に依存します。

- 応答に含まれるインデックスの部分を決定するフィールド。
- インデックスにおける一致を示す行。

フィールドは、属性が "取得可能" である場合に検索結果に表示されます。インデックススキーマ内のフィールド定義には属性があり、これがフィールドが応答で使用されるかどうかを決定します。"取得可能" フィールドだけがフルテキストクエリまたはベクトルクエリ結果で返されます。既定では、すべての "取得可能" フィールドが返されますが、"選択" を使用してサブセットを指定できます。"取得可能" 以外に、フィールドに制限はありません。フィールドには、任意の長さまたは型を指定できます。長さについて、Azure AI Search にはフィールド長の上限はありませんが、[API 要求のサイズ](#)には制限があります。

行ではクエリとの一致が、関連性、類似性、またはその両方でランク付けされます。既定では、結果はフルテキスト検索の場合は上位 50 件、ベクトル検索の場合は K ニアレストネイバーに制限されます。既定値を変更して制限を (最大 1000 ドキュメントまで) 増減できます。top および skip ページング パラメーターを使用して、結果を一連のページングされた結果として取得することもできます。

## 関連性でランク付けする

複雑なプロセスや大量のデータを処理する場合や、ミリ秒レベルの応答が期待されている場合、各ステップで価値を高め、最終的な結果の品質を向上させることが重要です。情報取得側において、関連性のチューニングは、LLM に送信される結果の品質を向上させるアクティビティです。結果には、最も関連性の高い、または最も類似した一致するドキュメントだけを含める必要があります。

関連性は、キーワード (非ベクトル) 検索とハイブリッド クエリ (非ベクトル フィールドに対する) に適用されます。Azure AI Search では、類似性検索とベクトルクエリの関連性のチューニングはありません。[BM25 ランク付け](#)は、フルテキスト検索のランク付けアルゴリズムです。

関連性のチューニングは、BM25 ランク付けを強化する機能によってサポートされます。これらのアプローチには、次が含まれます。

- [スコアリング プロファイル](#)。これは、特定の検索フィールドまたはその他の条件で一致が見つかった場合に検索スコアを向上させます。
- [セマンティックランク付け](#)。これは、Bing のセマンティック モデルを使用して結果を並べ替え、元のクエリに合わせてセマンティックに適合するように BM25 結果セットを再ランク付けします。

比較テストおよびベンチマーク テストでは、テキストフィールドとベクトル フィールドを含むハイブリッド クエリに、BM25 ランクの結果に対するセマンティック ランク付けを補足することで、最も関連性の高い結果が生成されます。

## RAG シナリオの Azure AI Search クエリのコード例

次のコードは、デモ サイトからの  [retrievethenread.py](#) ファイルからコピーされます。ハイブリッド クエリの検索結果から LLM の `content` が生成されます。より簡単なクエリを記述できますが、この例では、セマンティック再ランク付けとスペルチェックを使用したベクトル検索とキーワード検索が含まれています。デモでは、このクエリを使用して初期コンテンツを取得します。

Python

```
# Use semantic ranker if requested and if retrieval mode is text or hybrid
(vectors + text)
if overrides.get("semantic_ranker") and has_text:
    r = await self.search_client.search(query_text,
                                         filter=filter,
                                         query_type=QueryType.SEMANTIC,
                                         query_language="en-us",
                                         query_speller="lexicon",
                                         semantic_configuration_name="default",
                                         top=top,
                                         query_caption="extractive|highlight-false")
if use_semantic_captions else None,
                                         vector=query_vector,
                                         top_k=50 if query_vector else None,
                                         vector_fields="embedding" if query_vector
else None)
else:
    r = await self.search_client.search(query_text,
                                         filter=filter,
                                         top=top,
                                         vector=query_vector,
                                         top_k=50 if query_vector else None,
                                         vector_fields="embedding" if query_vector
else None)
if use_semantic_captions:
    results = [doc[self.sourcepage_field] + ": " + nonewlines(" .
    ".join([c.text for c in doc['@search.captions']])) async for doc in r]
else:
    results = [doc[self.sourcepage_field] + ": " +
nonewlines(doc[self.content_field]) async for doc in r]
content = "\n".join(results)
```

## 統合コードと LLM

Azure AI Search を含む RAG ソリューションでは、完全なソリューションを作成するために、他のコンポーネントとコードが必要です。前のセクションでは、Azure AI Search を使用した情報の取得と、検索可能なコンテンツの作成とクエリに使用される

機能について説明しましたが、このセクションでは LLM の統合と相互作用について説明します。

デモ リポジトリ内のノートブックは、LLM に検索結果を渡すためのパターンを示しているため、出発点として最適です。RAG ソリューションのほとんどのコードは LLM の呼び出しで構成されているため、この記事の範囲外であるこれらの API のしくみを理解する必要があります。

[chat-read-retrieve-read.ipynb](#) ノートブックの次のセルブロックは、チャットセッションのコンテキストでの検索呼び出しを示しています。

```
Python

# Execute this cell multiple times updating user_input to accumulate chat
history
user_input = "Does my plan cover annual eye exams?"

# Exclude category, to simulate scenarios where there's a set of docs you
can't see
exclude_category = None

if len(history) > 0:
    completion = openai.Completion.create(
        engine=AZURE_OPENAI_GPT_DEPLOYMENT,
        prompt=summary_prompt_template.format(summary="\n".join(history),
question=user_input),
        temperature=0.7,
        max_tokens=32,
        stop=["\n"])
    search = completion.choices[0].text
else:
    search = user_input

# Alternatively simply use search_client.search(q, top=3) if not using
semantic ranking
print("Searching:", search)
print("-----")
filter = "category ne '{}'".format(exclude_category.replace("'", ""))
if exclude_category else None
r = search_client.search(search,
                        filter=filter,
                        query_type=QueryType.SEMANTIC,
                        query_language="en-us",
                        query_speller="lexicon",
                        semantic_configuration_name="default",
                        top=3)
results = [doc[KB_FIELDS_SOURCEPAGE] + ": " +
doc[KB_FIELDS_CONTENT].replace("\n", "").replace("\r", "") for doc in r]
content = "\n".join(results)

prompt = prompt_prefix.format(sources=content) + prompt_history + user_input
+ turn_suffix
```

```

completion = openai.Completion.create(
    engine=AZURE_OPENAI_CHATGPT_DEPLOYMENT,
    prompt=prompt,
    temperature=0.7,
    max_tokens=1024,
    stop=[ "<|im_end|> ", "<|im_start|>" ])

prompt_history += user_input + turn_suffix + completion.choices[0].text +
"\n<|im_end|>" + turn_prefix
history.append("user: " + user_input)
history.append("assistant: " + completion.choices[0].text)

print("\n-----\n".join(history))
print("\n-----\nPrompt:\n" + prompt)

```

## ファースト ステップ<sup>1</sup>

- Azure AI Studio を使用して検索インデックスを作成します。
- Azure OpenAI Studio と "データ持ち込み"を使用して、プレイグラウンドで既存の検索インデックスに対するプロンプトを試します。この手順は、使用するモデルを決定するのに役立ち、RAG シナリオで既存のインデックスがどの程度適切に動作するかを示します。
- Azure AI 検索チームによって構築された "データとのチャット" ソリューション アクセラレータ<sup>2</sup>は、独自のカスタム RAG ソリューションを作成するのに役立ちます。
- エンタープライズ チャット アプリ テンプレート<sup>3</sup>では、Contoso と Northwind の架空の医療保険ドキュメントを使用して、Azure リソース、コード、サンプルのグラウンディング データをデプロイします。このエンドツー エンド ソリューションを使用すると、運用チャット アプリをわずか 15 分ほどで利用できます。これらのテンプレートのコードは、いくつかのプレゼンテーションで取り上げられる azure-search-openai-demo です。次のリンクでは言語固有のバージョンが提供されます。
  - .NET<sup>4</sup>
  - Python<sup>5</sup>
  - JavaScript<sup>6</sup>
  - Java<sup>7</sup>
- インデックス作成の概念と戦略を確認して、データを取り込む方法と更新方法を決定します。ベクトル検索、キーワード検索、ハイブリッド検索のどれを使用するかを決定します。検索する必要があるコンテンツの種類と実行するクエリの種類によって、インデックスの設計が決まります。

- クエリの作成について確認して、検索要求の構文と要件の詳細について確認します。

### ① 注意

一部の Azure AI Search 機能は人による操作を目的としており、RAG パターンでは役に立ちません。具体的には、オートコンプリートと候補をスキップできます。ファセットや orderby などの他の機能は役立つ可能性がありますが、RAG シナリオでは一般的ではありません。

## 関連項目

- 取得拡張生成: インテリジェント自然言語処理モデルの作成の合理化 ↗
- Azure Machine Learning プロンプト フローを使用した取得拡張生成
- Azure Cognitive Search と LangChain: 拡張ベクトル検索機能のシームレスな統合 ↗

# Azure AI Search でクエリを実行する

[アーティクル] • 2023/11/15

Azure AI Search では、フリーフォーム テキスト検索から高度に指定されたクエリ パターンやベクトル検索まで、さまざまなシナリオのクエリ コンストラクトがサポートされます。すべてのクエリは、検索可能なコンテンツを格納する検索インデックスに対して実行されます。

## クエリの種類

クエリ	検索可能なコンテンツ	説明
フルテキスト検索	トーカン化された用語の逆インデックス。	フルテキストクエリでは、任意の数の検索ドキュメント内で高速スキャンできるように構造化された転置インデックスを反復処理します。この場合、すべてのフィールドで一致が見つかる可能性があります。テキストは分析されてフルテキスト検索用にトーカン化されます。
ベクトル検索	生成された埋め込みのベクトルインデックス。	ベクトル クエリでは、検索インデックスのベクトル フィールドが反復処理されます。
ハイブリッド検索	1つの検索インデックスで上記のすべて。	1つのクエリ要求でテキスト検索とベクトル検索を組み合わせます。テキスト検索は、"検索可能" および "フィルター処理可能" なフィールドのプレーンテキストコンテンツに対して機能します。ベクトル検索は、ベクトル フィールド内のコンテンツに対して機能します。
その他	プレーンテキストと英数字のコンテンツ。	ソース ドキュメントから逐語的に抽出された未加工のコンテンツは、地理空間検索、あいまい検索、フィールド検索などのフィルターおよびパターン マッチング クエリをサポートします。

この記事では最後のカテゴリ、つまり、フィルターやその他の特殊なクエリ フォームに使用される、元のソースからそのまま抽出された、プレーン テキストと英数字のコンテンツに対して機能するクエリに焦点を当てます。

## オートコンプリートとクエリ候補

[オートコンプリート](#)や[結果候補](#)は、`search` の代替手段です。これは、部分文字列の入力 (各文字の後) に基づいて一連のクエリ要求を入力につれて検索する方式で行いま

す。 このチュートリアルで説明しているように、`autocomplete` と `suggestions` パラメーターは一緒に使用することも、個別に使用することができますが、`search` と一緒に使用することはできません。 完成した用語とクエリ候補はどちらもインデックスの内容から派生されます。 このエンジンは、インデックスに存在しない文字列や候補を返しません。 詳細については、「[オートコンプリート \(REST API\)](#)」と「[検索候補 \(REST API\)](#)」をご覧ください。

## フィルター検索

フィルターは、Azure AI Search を元に構築したアプリで広く使用されています。 アプリケーションページでは、フィルターは多くの場合、ユーザー向けのフィルター処理のためにリンクナビゲーション構造のファセットとして視覚化されます。 フィルターは、インデックス付きコンテンツのスライスを公開するために、内部的にも使用されます。 たとえば、製品カテゴリに対してフィルターを使用して検索ページを初期化したり、インデックスに英語とフランス語の両方のフィールドが含まれている場合は、言語を初期化したりすることができます。

次の表に示すように、特殊なクエリ フォームを呼び出すフィルターが必要な場合もあります。 指定されていない検索 (`search=*`) を含むフィルターを使用することも、用語、語句、演算子、およびパターンを含むクエリ文字列を含むフィルターを使用することもできます。

ユーザ	説明
—	—
範囲フィルター	Azure AI Search では、範囲クエリは filter パラメーターを使用して作成されます。 詳細と例については、「 <a href="#">範囲フィルターの例</a> 」をご覧ください。
—	—
ファセットナビゲーション	ファセットナビゲーションツリーでは、ユーザーがファセットを選択できます。 フィルターでサポートされている場合、クリックするたびに検索結果が絞り込まれます。 ファセットによって指定された条件に一致しなくなったドキュメントを除外するフィルターによって、各ファセットがサポートされます。

### ① 注意

フィルター式で使用されるテキストは、クエリ処理中には分析されません。 テキスト入力は、照合に成功するか失敗するかのどちらかになる逐語的な文字パターンで、大文字と小文字が区別されます。 フィルター式は OData 構文を使用して構築され、インデックス内のすべての "フィルター可能な" フィールドの `filter` パ

ラメーターで渡されます。 詳細については、「[Azure AI Search のフィルター](#)」をご覧ください。

## 地理空間検索

地理空間検索では、"近くを検索" またはマップベースの検索エクスペリエンスのために、場所の緯度と経度の座標との一致を検索します。 Azure AI Search では、次の手順に従って地理空間検索を実装できます。

- [Edm.GeographyPoint](#)、[Collection\(Edm.GeographyPoint\)](#)、[Edm.GeographyPolygon](#) のいずれかの種類のフィルター可能フィールドを定義します。
- 受信ドキュメントに適切な座標が含まれていることを確認します。
- インデックス作成が完了したら、フィルターと[地理空間関数](#)を使用するクエリを作成します。

地理空間検索では、距離にキロメートルを使用します。 座標は `(longitude, latitude)` の形式で指定されます。

地理空間検索のフィルターの例を次に示します。 このフィルターでは、地理的なポイント (この例ではワシントン D.C.) から半径 300 キロ以内に座標を持つ他の `Location` フィールドを検索インデックスから見つけます。 結果にアドレス情報を返し、場所による自己ナビゲーションのためにオプションの `facets` 句を含めます。

### HTTP

```
POST https://{{searchServiceName}}.search.windows.net/indexes/hotels-vector-quickstart/docs/search?api-version=2023-07-01-Public
{
    "count": true,
    "search": "*",
    "filter": "geo.distance(Location, geography'POINT(-77.03241 38.90166)') le 300",
    "facets": [ "Address/StateProvince" ],
    "select": "HotelId, HotelName, Address/StreetAddress, Address/City, Address/StateProvince",
    "top": 7
}
```

詳細および例については、[地理空間検索の例](#)に関するページをご覧ください。

## ドキュメントの検索

前述のクエリ フォームとは対照的に、このフォームでは、対応するインデックス検索またはスキヤンを行わずに、[IDによる検索ドキュメント](#)を 1 つ取得します。1 つのドキュメントのみが要求され、返されます。ユーザーが検索結果で項目を選択する場合、ドキュメントの取得と詳細ページでのフィールドの設定が典型的な応答になり、ドキュメントの検索はこれをサポートする操作になります。

## 高度な検索: あいまい、ワイルドカード、近接、正規表現

高度なクエリ フォームは、特定のクエリ動作をトリガーする完全な Lucene パーサーと演算子に依存します。

クエリの種類	使用法	例/詳細情報
フィールド検索	<code>search</code> パラメーター、 <code>queryType=full</code>	1 つのフィールドを対象とする複合クエリ式を作成します。 <a href="#">フィールド検索の例</a>
あいまい検索	<code>search</code> パラメーター、 <code>queryType=full</code>	構造やスペリングが似ている語句を照合します。 <a href="#">あいまい検索の例</a>
近接検索	<code>search</code> パラメーター、 <code>queryType=full</code>	ドキュメント内で近くにある語句を検索します。 <a href="#">近接検索の例</a>
用語ブースト	<code>search</code> パラメーター、 <code>queryType=full</code>	ブーストされた語を含むドキュメントの順位を、含まないドキュメントよりも引き上げます。 <a href="#">用語ブーストの例</a>
正規表現検索	<code>search</code> パラメーター、 <code>queryType=full</code>	正規表現の内容に基づいて照合します。 <a href="#">正規表現の例</a>
ワイルドカードまたはプレフィックス検索	<code>*</code> ~ または <code>?</code> を使用した <code>search</code> パラメーター、 <code>queryType=full</code>	プレフィックスとチルダ (~) または 1 つの文字 (?) に基づいて照合します。 <a href="#">ワイルドカード検索の例</a>

## 次のステップ°

クエリの実装について詳しく見るには、構文ごとに例を確認します。フルテキスト検索を初めて使用する場合は、クエリ エンジンの機能をよく理解しておくことをお勧めします。

- 簡易クエリの例
- 高度なクエリを作成するための Lucene 構文のクエリの例
- Azure AI Search でのフルテキスト検索のしくみgit

# Azure AI Search でのセマンティック ランク付け

[アーティクル] • 2024/02/08

Azure AI Search のセマンティック ランク付けでは、検索結果を再ランク付けするために、言語理解を利用して検索結果の関連性をある程度高めます。この記事では、概要について説明します。最後のセクションでは、販売状況と価格について説明します。

セマンティック ランカーはプレミアム機能であり、使用量に基づいて課金されます。最初にこの記事を読んで基礎知識を得ることをお勧めしますが、それはかまわないのですぐに使いたいという方は、これらの手順を実行してください。

- ✓ リージョン別の提供状況を確認する
- ✓ Azure portal にサインインして、検索サービスが Basic 以上であることを確認する
- ✓ セマンティック ランク付けを有効にして価格プランを選択する
- ✓ 検索インデックスでセマンティック構成を設定する
- ✓ セマンティック キャプションとハイライトを返すようにクエリを設定する
- ✓ 必要に応じて、セマンティック回答を返す

## ① 注意

セマンティック ランク付けでは、生成 AI やベクトルは使用されません。ベクトルのサポートと類似性検索をお探しの場合、「[Azure AI Search でのベクトル検索](#)」で詳細をご確認いただけます。

## セマンティック ランク付けとは

セマンティック ランカーは、テキストベースのクエリに対する、BM25 でランク付けされたあるいは RRF でランク付けされた最初の検索結果の品質を高めるクエリ関連機能のコレクションです。これを検索サービスで有効にすると、セマンティック ランク付けによってクエリの実行パイプラインに 2 つの機能が追加されます。

- 1 つ目として、BM25 または RRF を使用してスコア付けされた、最初の結果セットに対する二次ランク付けが追加されます。この二次ランク付けでは多言語の、Microsoft Bing から変化したディープ ラーニング モデルが使用されて、セマンティック的に最も関連性の高い結果が奨励されます。

- 2つ目として、キャプションと回答が抽出されて応答で返されます。これを検索ページにレンダリングして、ユーザーの検索エクスペリエンスを向上させることができます。

セマンティック リランカーの機能を次に示します。

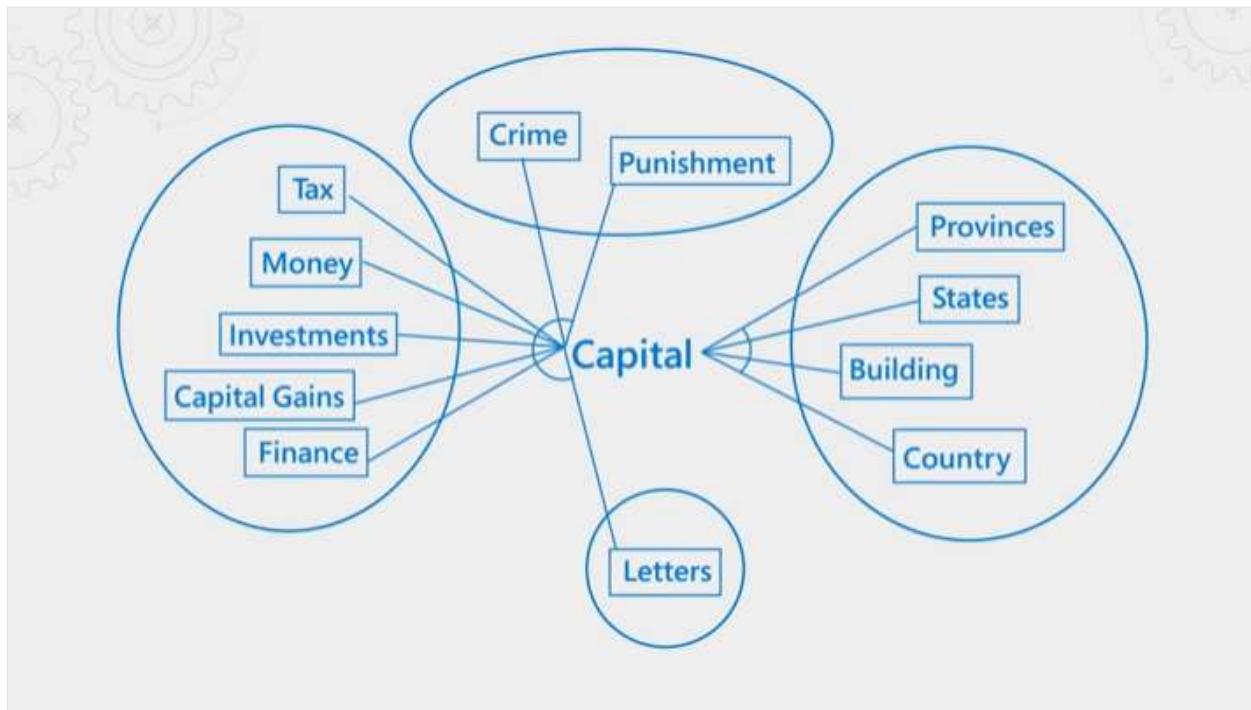
[+] [テーブルを展開する](#)

特徴量	説明
セマンティックランク付け	クエリのコンテキストまたはセマンティックの意味を利用して、事前にランク付けされた結果に対して新しい関連スコアを計算します。
セマンティックキーワードハイライト	コンテンツを最もよく要約している逐語的な文やフレーズをドキュメントから抽出し、スキャンを簡単にするために重要な部分を強調表示します。結果を要約するキーワードハイライトは、個々のコンテンツ フィールドが検索結果ページに対して高密度である場合に便利です。強調表示されたテキストにより、最も関連性の高い用語とフレーズが目立つため、ユーザーはその一致が関連していると見なされた理由を迅速に判断できます。
セマンティック回答	セマンティック クエリから返される省略可能な追加のサブ構造体。これにより、質問のようなクエリに直接回答することができます。ドキュメントには、回答の特性を持つテキストが含まれている必要があります。

## セマンティック ランカーのしくみ

セマンティック ランク付けは、クエリと結果を Microsoft がホストする言語理解モデルにフィードし、より適切な一致をスキャンします。

コンセプトを次の図で説明します。 "capital" という用語を考えてみましょう。これは、コンテキストが財務、法律、地理、文法なのかによって意味が異なります。言語理解では、セマンティックのランカーによってコンテキストが検出され、クエリの意図に沿う結果が昇格されます。



セマンティックの順位付けは、リソースと時間の両方を消費します。 クエリ操作の予想待機時間内に処理を完了するために、セマンティック ランク付けへの入力が統合され、削減されるため、再ランク付けの手順をできるだけ早く完了できます。

セマンティック ランク付けには、サマライゼーションとスコアリングという 2 つの手順があります。 出力が再スコアリングされた結果、キャプション、および回答で構成されます。

## 入力が収集されて要約されるしくみ

セマンティック ランク付けでは、クエリのサブシステムからサマライゼーション モデルとランク付けモデルに検索結果が入力として渡されます。 ランク付けモデルには入力サイズに制約があり、集中的に処理されるため、検索結果は効率的に処理できるようなサイズで構造化(要約)されている必要があります。

1. セマンティック ランク付けは、テキスト クエリの [BM25 でランク付けされた検索結果](#)から、またはハイブリッド クエリの [RRF でランク付けされた結果](#)から開始されます。 テキスト フィールドのみが再ランク付け実行で使用され、結果の数が 50 個を超える場合でも、セマンティック ランク付けが行われるのは上位 50 個の結果のみです。 通常、セマンティック ランク付けで使用されるフィールドは、情報を探する説明的なものです。
2. 検索結果の中の各ドキュメントで、サマライゼーション モデルが受け入れるのは 2,000 トークンまでで、この場合、1 トークンはおよそ 10 文字です。 入力が[セマンティック構成](#)の一覧にある "title"、"keyword"、および "content" フィールドからアセンブルされます。

3. 長さが非常に長い文字列は、全体の長さが概要作成手順の入力要件を満たすようにトリミングされます。優先順位の高い順序でセマンティック構成にフィールドを追加することが重要なのは、このトリミングの実行があるためです。テキストを多用するフィールドを持つ非常に大きいドキュメントがある場合は、最大値制限を超えたテキストは無視されます。

#### [+] テーブルを展開する

セマンティック フィールド	トークンの限度
「タイトル」	128 トークン
"keywords"	128 トークン
"content"	残りのトークン

4. サマライゼーション出力は各ドキュメントの要約文字列であり、各フィールドの中の最も関連した情報で構成されます。要約文字列がスコアリングのためランカーへ送られて、キャプションと回答を求めて機械読み取り理解モデルへ送られます。

セマンティックランカーへ渡されるそれぞれの生成後の要約文字列の最大長さは、256 トークンです。

## セマンティックランカーの出力

各要約文字列から、機械読み取り理解モデルは最も代表的な一節を見つけ出します。

出力は次のようにになります:

- ドキュメントのセマンティックキャプション。各キャプションは、プレーンテキストバージョンと強調表示バージョンで使用できます。また、多くの場合、ドキュメントあたり 200 語未満です。
- `answers` パラメーターを指定した場合、クエリが質問として提示された場合、その質問に対して適していそうな回答を提供する長い文字列で文節が見つかる場合を想定し、オプションのセマンティック回答が返されます。

キャプションと回答は、常にインデックスからの逐語的なテキストです。このワークフローには、新しいコンテンツを作成または構成する生成 AI モデルはありません。

## 要約がスコアリングされるしくみ

スコアリングは、キャプションと、256 のトークン長を埋める要約文字列の中の他のあらゆるコンテンツに対して行われます。

1. キャプションは、指定されたクエリに対して相対的な概念とセマンティックの関連性に対して評価されます。
2. @search.rerankerScore が各ドキュメントに、指定のクエリのドキュメントのセマンティック関連性に基づいて割り当てられます。スコアの範囲は 4 から 0 (高から低) です。スコアが高いほど関連性が高いことを示します。
3. 一致は、スコア順に一覧表示され、クエリ応答ペイロードに含められます。ペイロードには、回答、プレーンテキスト、強調表示されたキャプション、select 句で取得または指定されたフィールドが含まれます。

#### ① 注意

2023 年 7 月 14 日より、@search.rerankerScore の分布が変更されます。スコアの結果を、テスト以外で判断することはできません。この応答プロパティにハードしきい値の依存関係がある場合、テストを返して、しきい値に適切な新しい値を確認してください。

## セマンティック機能と制限

セマンティック ランカーは新しいテクノロジなので、実行できることとできないことについての期待値を設定することが重要です。"できること" は、次のようなことです。

- セマンティック的に元のクエリの意図に近い一致を昇格させます。
- キャプションおよび回答として使用できる文字列を見つけ出します。キャプションと回答は応答で返され、検索結果ページに表示できる文字列が検出されます。

セマンティック ランク付けで "できない" ことは、コーパス全体に対してクエリを再実行して、セマンティックな関連がある結果を検出することです。セマンティック ランク付けでは、既定のランク付けアルゴリズムによってスコアリングされた上位 50 個の結果で構成される既存の結果セットが再ランク付けされます。さらに、セマンティック ランク付けで新しい情報や文字列を作成することはできません。キャプションと回答は、コンテンツから逐語的に抽出されるので、結果に回答のようなテキストが含まれていない場合、その言語モデルではキャプションや回答は生成されません。

セマンティック ランク付けはすべてのシナリオで有益なわけではありませんが、特定のコンテンツではその機能から多くのメリットが得られます。セマンティック ランク

付けの言語モデルは、情報が豊富で、散文として構造化された検索可能なコンテンツに最適です。ナレッジベース、オンラインドキュメント、説明的なコンテンツを含むドキュメントでは、セマンティックランク付け機能から最も多くのメリットが得られます。

テクノロジは Bing と Microsoft Research が基になっており、Azure AI Search インフラストラクチャにアドオン機能として統合されます。研究と AI 投資でバックアップされるセマンティックランク付けに関する詳細については、「[Bing の AI が Azure AI Search にパワーを与えるしくみ \(Microsoft Research ブログ\)](#)」を参照してください。

次の動画では、機能の概要について説明しています。

[https://www.youtube-nocookie.com/embed/yOf0WfVd\\_V0](https://www.youtube-nocookie.com/embed/yOf0WfVd_V0)

## 可用性と料金

セマンティックランカーは、[利用可能なリージョン](#)において、Basic レベル以上の検索サービスで使用できます。

セマンティックランカーを有効にする場合は、機能に対応する価格プランを選択します。

- クエリボリュームが低い場合(月間 1000 件未満)、セマンティックランク付けは無料です。
- クエリのボリュームが大きい場合、標準価格プランを選択します。

[Azure AI Search の価格ページ](#)では、さまざまな通貨とサイクル間隔での課金レートが表示されます。

セマンティックランク付けの料金は、`queryType=semantic` を含む、検索文字列が空でないクエリ要求(たとえば `search=pet friendly hotels in New York`)を使用した場合に発生します。検索文字が空の場合(`search=*`)、`queryType` が `semantic` に設定されても課金されません。

## 関連項目

- [セマンティックランク付けを有効にする](#)
- [セマンティックランク付けを構成する](#)
- [ブログ: ハイブリッド検索とランク付け機能でベクトル検索の性能を上回る](#)

# キーワード検索での関連性 (BM25 スコアリング)

[アーティクル] • 2024/05/08

この記事では、[フルテキスト検索](#)で検索スコアを計算するために使用される BM25 関連性スコアリングのアルゴリズムについて説明します。BM25 関連性は、フルテキスト検索に限定されます。フィルター クエリ、オートコンプリート、提案されたクエリ、ワイルドカード検索、あいまい検索の各クエリは、関連性についてスコアリングもランク付けもされません。

## フルテキスト検索で使用されるスコアリング アルゴリズム

Azure AI Search には、フルテキスト検索用の次のスコアリング アルゴリズムが用意されています：

[] テーブルを展開する

アルゴリズム	使用法	Range
BM25Similarity	2020 年 7 月以降に作成されたすべての検索サービスでアルゴリズムを修正しました。このアルゴリズムは構成可能ですが、古いアルゴリズム (クラシック) に切り替えることはできません。	無制限。
ClassicSimilarity	以前の検索サービスに存在します。 <a href="#">BM25 をオプトイン</a> し、インデックスごとにアルゴリズムを選択できます。	0 < 1.00

BM25 とクラシックはいずれも TF-IDF タイプの取得関数です。この関数では、単語の出現頻度 (TF) と逆文書頻度 (IDF) が変数として使用され、ドキュメントとクエリの組みごとに関連スコアが計算されます。ドキュメントとクエリの組みはその後、ランク付けの結果に使用されます。概念的にはクラシックと似ていますが、BM25 は確率論的情報取得に根ざしており、ユーザーの調査で測定した場合のように、より直感的な一致が生成されます。

BM25 には高度なカスタマイズ オプションがあります。たとえば、ユーザーは、一致した単語の出現頻度で関連性スコアが変動するしくみを決定できます。詳細については、[スコアリング アルゴリズムの構成](#)に関するページを参照してください。

### ① 注意

2020年7月より前に作成された検索サービスを使用している場合、スコアリングアルゴリズムは当時の既定のアルゴリズム (`ClassicSimilarity`) である可能性が高く、その場合は、インデックスごとにアップグレードを行うことができます。 詳細については、「[以前のサービスで BM25 スコアリングを有効にする](#)」を参照してください。

次のビデオ セグメントでは、Azure AI Search で使用される一般提供の優先度付けアルゴリズムの説明に早送りしています。 詳しい背景情報については、ビデオ全編をご覧ください。

[https://www.youtube-nocookie.com/embed/Y\\_X6USgvB1g?version=3&start=322&end=643](https://www.youtube-nocookie.com/embed/Y_X6USgvB1g?version=3&start=322&end=643) ↗

## BM25 ランク付けのしくみ

関連性のスコアリングとは、検索スコア (`@search.score`) を計算することです。これは、現在のクエリのコンテキストにおける項目の関連性のインジケーターとして機能します。範囲は無制限です。ただし、スコアが高いほど、項目の関連性が高くなります。

検索スコアは、文字列入力とクエリ自体の統計プロパティに基づいて計算されます。 Azure AI Search では、検索語句に一致するドキュメント (`searchMode` に応じて一部または全部) が検索され、検索語句の多くのインスタンスを含むドキュメントが優先されます。データインデックス全体での語句の出現頻度は低いがドキュメント内ではよく使用されている場合、検索スコアはより高くなります。関連性を計算するこのアプローチの基礎となる手法は、*TF-IDF* (単語の出現頻度 - 逆文書頻度) と呼ばれています。

検索スコアは、結果セット全体で繰り返すことができます。同じ検索スコアを持つ項目が複数ヒットした場合、同じスコアを持つ項目の順序付けは定義されていないので安定しません。クエリを再度実行すると、特に、複数のレプリカで無料のサービスまたは課金対象サービスを使用している場合は、項目の位置が変わることがあります。同ースコアの項目が 2 つ存在する場合、どちらが最初に表示されるかは特定できません。

繰り返しスコアの間の関係を解除するには、`$orderby` 句を追加することで、まずスコアで並べ替えを行い、次に別の並べ替え可能なフィールド (`$orderby=search.score() desc, Rating desc` など) で並べ替えを行うことができます。 詳細については、[\\$orderby](#) に関するページを参照してください。

スコア付けには、インデックスで `searchable` としてマークされているフィールド、またはクエリで `searchFields` としてマークされているフィールドが使用されます。

`retrievable` としてマークされたフィールド、またはクエリの `select` で指定されたフィールドのみが、検索スコアと共に検索結果に返されます。

### ① 注意

`@search.score = 1` は、スコア付けまたは順位付けが行われていない結果セットを示します。スコアは、すべての結果にわたって均一です。スコア付けされていない結果が発生するのは、クエリ フォームがファジー検索、ワイルドカード、または正規表現のクエリである場合、または空の検索である場合 (`search=*` がフィルターとペアで指定され、一致を返す主な手段がフィルターである場合があります) です。

## テキスト結果のスコア

結果がランク付けされるたびに、`@search.score` プロパティには結果の順序付けに使用される値が含まれます。

次の表に、各一致、アルゴリズム、および範囲で返されるスコアリング プロパティを示します。

[+] テーブルを展開する

検索メソッド	パラメーター	スコアリング アルゴリズム	Range
フルテキスト検索	<code>@search.score</code>	BM25 アルゴリズム。インデックスで指定されたパラメーターを使用します。	無制限。

## スコアバリエーション

検索スコアは、一般的な意味での関連性を示すものであり、同じ結果セット内の他のドキュメントと比べた場合の一一致の強さが反映されています。ただし、スコアは、あるクエリと次のものとの間で必ずしも一貫しているとは限らないため、クエリを操作していると、検索ドキュメントの順序付け方法における小さな不一致に気付くことがあります。これが発生する理由については、次のいくつかの説明があります。

[+] テーブルを展開する

原因	説明
同一のスコア	複数のドキュメントのスコアが同じである場合、それらはいずれも最初に表示される可能性があります。
データの不安定性	ドキュメントを追加、変更、または削除すると、インデックスコンテンツは変動します。インデックスの更新が徐々に処理されるに従って用語頻度は変化し、一致するドキュメントの検索スコアに影響を与えます。
複数のレプリカ	複数のレプリカを使用するサービスの場合、クエリは、各レプリカに対して並列に発行されます。検索スコアを計算するために使用されるインデックス統計はレプリカごとに計算され、クエリ応答の中で結果がマージされ、順序付けられます。レプリカのほとんどは互いのミラーですが、状態の小さな違いのために統計は異なる場合があります。たとえば、あるレプリカで、他のレプリカからマージされた、その統計に寄与しているドキュメントが削除されることがあります。通常、レプリカごとの統計の違いは、小さなインデックスの方がより顕著です。この条件について詳しくは、以下のセクションで説明します。

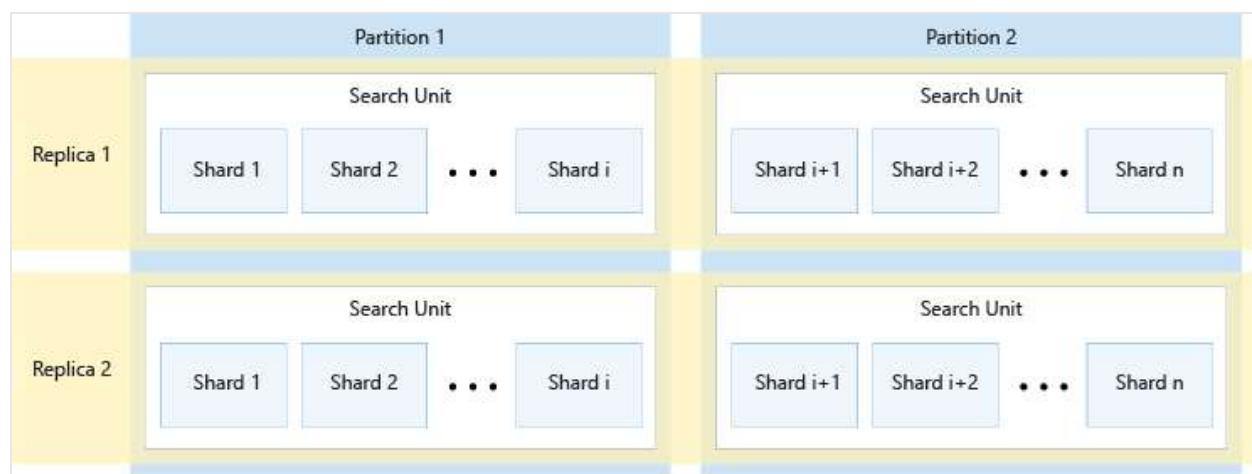
## クエリ結果に対するシャーディング効果

"シャード"とは、インデックスのチャンクです。 Azure AI Search では、インデックスをさらに "シャード" に分割し、(シャードを新しい検索単位に移動することによって)パーティションを追加するプロセスを高速化しています。検索サービスでは、シャード管理は実装の詳細であり、構成できませんが、インデックスがシャード化されていると分かっていれば、順位付けやオートコンプリートの動作で不定期に発生する異常を容易に把握できます。

- 異常のランク付け: 検索スコアは最初にシャード レベルで計算され、続いて単位の結果セットに集計されます。シャード コンテンツの特性に応じて、あるシャードからの一致が別のシャードの一致よりも高い順位になる場合があります。検索結果の順位付けが直観に反しているように感じられる場合は、シャーディングの影響が原因である可能性が最も高いです (特にインデックスが小さい場合)。[インデックス全体でグローバルにスコアを計算する](#)ように選択すれば、これらの順位付けの異常を回避できますが、その場合、パフォーマンスが低下します。

- オートコンプリートの異常: オートコンプリート クエリでは、部分的に入力された語句の最初のいくつかの文字で照合が行われますが、スペルの少しの間違いを許容するあいまいパラメーターが使用されます。オートコンプリートの場合、あいまい一致は現在のシャード内の用語に限定されます。たとえば、シャードに "Microsoft" が含まれており、"micro" という部分的な語句が入力された場合、検索エンジンはそのシャード内の "Microsoft" と一致しますが、インデックスの残りの部分を保持した他のシャードでは一致しません。

次の図は、レプリカ、パーティション、シャード、および検索単位間の関係を示しています。ここでは、2つのレプリカと2つのパーティションを持つサービスで、どのように1つのインデックスが4つの検索単位にわたっているかを例で示しています。4つの検索単位それぞれには、インデックスのシャードの半分だけが格納されます。左側の列の検索単位は、シャードの前半を格納して最初のパーティションを構成し、右側の列の検索単位は、シャードの後半を格納して2番目のパーティションを構成しています。レプリカが2つあるため、各インデックスシャードのコピーは2つあります。上部の行の検索単位は、1つのコピーを格納して最初のレプリカを構成し、下部の行の検索単位は、別のコピーを格納して2番目のレプリカを構成しています。



上の図は1つの例にすぎません。パーティションとレプリカはさまざまに組み合わせることができ、最大で合計36の検索単位が可能です。

### ① 注意

レプリカとパーティションの数は、均等に12分割されます(具体的には、1、2、3、4、6、12)。Azure AI Searchでは各インデックスを12のシャードに事前に分割して、それぞれがすべてのパーティションに均等に分散されるようにします。たとえば、サービスに3つのパーティションがあり、インデックスを作成する場合、各パーティションにはインデックスの4つのシャードを含めます。Azure AI Searchでインデックスがどのようにシャードされるかは実装の詳細であり、今後のリリースで変更される場合があります。今日12個であっても、今後も必ず12個になるとは限りません。

# スコアリング統計とステイッキーセッション

スケーラビリティのために、Azure AI Search はシャーディングプロセスを通じて各インデックスを水平方向に配布します。つまり、[インデックスの部分は物理的に個別です。](#)

既定では、ドキュメントのスコアは、"シャード内" のデータの統計プロパティに基づいて計算されます。このアプローチは、一般に、データの大規模なコーパスでは問題にならず、すべてのシャードの情報に基づいてスコアを計算する必要がある場合よりもパフォーマンスが向上します。ただし、このパフォーマンスの最適化を使用すると、2つの非常に類似したドキュメント（またはまったく同一のドキュメント）は、それぞれが異なるシャードになる場合、関連性スコアが異なる可能性があります。

すべてのシャードの統計プロパティに基づいてスコアを計算する場合、これを行うには、`scoringStatistics=global` をクエリ パラメーターとして追加します（またはクエリ要求の本文 パラメーターとして `"scoringStatistics": "global"` を追加します）。

HTTP

```
POST https://[service name].search.windows.net/indexes/hotels/docs/search?  
api-version=2020-06-30  
{  
    "search": "<query string>",  
    "scoringStatistics": "global"  
}
```

`scoringStatistics` を使用すると、同じレプリカのすべてのシャードで同じ結果が得られるようになります。ただし、レプリカはインデックスの最新の変更で常に更新されるため、それぞれ若干異なる場合があります。一部のシナリオでは、ユーザーが "クエリ セッション" 中により一貫した結果を得られるようにすることが必要な場合があります。このようなシナリオでは、クエリの一部として `sessionId` を指定できます。

`sessionId` は、一意のユーザー セッションを参照するために作成する一意の文字列です。

HTTP

```
POST https://[service name].search.windows.net/indexes/hotels/docs/search?  
api-version=2020-06-30  
{  
    "search": "<query string>",  
    "sessionId": "<string>"  
}
```

同じ `sessionId` が使用されていれば、同じレプリカをターゲットにするためにベストエフォートの試行が行われるので、ユーザーに表示される結果の一貫性が向上します。

### ① 注意

同じ `sessionId` 値を繰り返し再利用すると、レプリカ間での要求の負荷分散が妨げられ、検索サービスのパフォーマンスに悪影響を与える可能性があります。`sessionId` として使用される値は、'\_' 文字で始めることはできません。

## 関連性のチューニング

Azure AI Search では、BM25 アルゴリズム パラメーターを構成し、次のメカニズムを使用して検索の関連性を調整し、検索スコアを向上させることができます：

[+] [テーブルを展開する](#)

アプローチ	実装	説明
スコアリング アルゴリズムの構成	Search index	
スコアリング プロファイル	Search index	コンテンツ特性に基づいて一致の検索スコアを向上させる基準が指定されます。たとえば、収益の見込みに基づいて一致をブーストしたり、より新しい項目のレベルを上げたり、場合によっては在庫期間が長すぎる項目をブーストしたりできます。スコアリング プロファイルは、インデックス定義の一部であり、重み付けされたフィールド、関数、およびパラメーターで構成されます。スコアリング プロファイルの変更によって既存のインデックスを更新できます。インデックスの再構築は必要ありません。
セマンティック ランク付け	クエリ 要求	検索結果に機械読解を適用し、意的に関連性の高い結果を上位に昇格させます。
featuresMode パラメーター	クエリ 要求	このパラメーターは、多くの場合、スコアのアンパックに使用されますが、 <a href="#">カスタム スコアリング ソリューション</a> を提供するコードで使用できます。

## featuresMode パラメーター (プレビュー)

ドキュメントの検索の要求には、フィールド レベルでの関連性に関するさらに詳細な情報を提供できる新しい `featuresMode` パラメーターがあります。`@searchScore` はドキュメント全体に対して計算されますが (このクエリのコンテキストにおけるこのドキ

メントの関連度)、`featuresMode` を使用すると、`@search.features` 構造体で表現された、個々のフィールドに関する情報を取得できます。この構造体には、クエリで使用されるすべてのフィールド (クエリ内の `searchFields` を介した特定のフィールド、またはインデックス内で検索可能として属性が付けられているすべてのフィールド) が含まれます。フィールドごとに、次の値が取得されます。

- フィールド内で見つかった一意のトークン数
- 類似性スコア。つまり、クエリ用語に対するフィールド内容の類似度のメジャーアルゴリズムによるスコア
- 用語の頻度。つまり、フィールド内でクエリ用語が見つかった回数

"Description" および "title" フィールドを対象とするクエリの場合、`@search.features` を含む応答は次のようにになります。

```
JSON

"value": [
  {
    "@search.score": 5.1958685,
    "@search.features": {
      "description": {
        "uniqueTokenMatches": 1.0,
        "similarityScore": 0.29541412,
        "termFrequency": 2
      },
      "title": {
        "uniqueTokenMatches": 3.0,
        "similarityScore": 1.75451557,
        "termFrequency": 6
      }
    }
  ]
]
```

カスタムのスコアリングソリューション<sup>↗</sup>でこれらのデータ ポイントを使用したり、この情報を使用して検索の関連性の問題をデバッグしたりできます。

## フルテキスト クエリ応答のランク付けされた結果の数

既定では、改ページを使用していない場合、検索エンジンはフルテキスト検索では上位 50 位のランクの一一致を返します。`top` パラメーターを使用して、返される項目数を減らしたり増やしたりできます (1 回の応答で 1000 個まで)。フルテキスト検索には、最大 1,000 件の一一致という制限が適用されます ([API 応答の制限](#) を参照)。1,000 件の一一致が見つかると、検索エンジンはそれ以上の検索を行いません。

返す結果をさらに増やす、または減らすには、ページング パラメーター `top`、`skip`、`next` を使用します。ページングは、各論理ページ上の結果の数を決定し、完全なペイロードを導く方法です。 詳細については、「[検索結果の操作方法](#)」を参照してください。

## 関連項目

- [スコアリング プロファイル](#)
- [REST API リファレンス](#)
- [ドキュメント API の検索](#)
- [Azure AI Search .NET SDK](#)

# ベクトル検索での関連性

[アーティクル] • 2024/04/18

ベクトル クエリの実行時、検索エンジンは類似のベクトルを検索して、検索結果に返される最適な候補を見つけます。ベクトルコンテンツのインデックス付け方法に応じて、関連する一致の検索は網羅的であるか、ニアネイバーに制限されて処理が高速化されます。候補が見つかると、類似性メトリックを使用して、一致の強度に基づいて各結果のスコアが付けられます。

この記事では、関連性のある一致を見つけるために使用されるアルゴリズムと、スコアリングに使用される類似性メトリックについて説明します。また、検索結果が期待に沿っていない場合に関連性を向上させるヒントも提供します。

## ベクトル検索で使用されるアルゴリズム

ベクトル検索アルゴリズムには、完全な  $k$  ニアレスト ネイバー (KNN) や Hierarchical Navigable Small World (HNSW) があります。

- 完全な KNN は、ベクトル空間全体をスキャンするブルート フォース検索を実行します。
- HNSW は、[近似ニアレスト ネイバー \(ANN\)](#) 検索を実行します。

検索とスコアリングに使用されるのは、インデックスに `searchable` としてマークされているフィールド、またはクエリの `searchFields` としてマークされたベクトル フィールドだけです。

## 完全な KNN を使用する場合

完全な KNN は、データ ポイントのすべてのペア間の距離を計算し、クエリ ポイントの正確な  $k$  ニアレスト ネイバーを見つけます。これは、高い再現率が最も重要であり、ユーザーにクエリ待ち時間のトレードオフを受け入れる意思があるシナリオを対象としています。計算負荷が高いため、小規模から中規模のデータセット、または精度要件がクエリ パフォーマンスの考慮事項を上回る場合は、完全な KNN を使用します。

2 番目の用途は、近似ニアレスト ネイバー アルゴリズムの呼び戻しを評価するデータセットの構築です。完全な KNN を使用して、ニアレスト ネイバーのグラウンド トゥ ルースのセットを構築できます。

完全な KNN サポートは、[2023-11-01 REST API](#) および [2023-10-01-Preview REST API](#) と、そのいずれかの REST API バージョンを対象とする Azure SDK クライアント ライブ

ラリで利用できます。

## HNSW を使用する場合

インデックス作成中に、HNSW は、データ ポイントを階層グラフ構造に編成して、より高速な検索のために追加のデータ構造を作成します。 HNSW には、検索アプリケーションのスループット、待機時間、および再現目標を達成するために調整できるいくつかの構成パラメーターがあります。 たとえば、クエリ時に、ベクトルフィールドに HNSW のインデックスが付いている場合でも、包括的な検索のオプションを指定できます。

クエリの実行中、HNSW はグラフ内を移動して高速な近隣クエリを有効にします。 このアプローチにより、検索の正確さと計算効率のバランスが取れます。 HNSW は、大規模なデータ セットを検索するときの効率が高いため、ほとんどのシナリオに推奨されます。

## ニアレストネイバー検索のしくみ

ベクトル クエリは、同じ埋め込みモデルから生成されたベクトルで構成される埋め込み空間に対して実行されます。 一般に、クエリ要求内の入力値は、ベクトルインデックスに埋め込みを生成したのと同じ機械学習モデルにフィードされます。 出力は、同じ埋め込み空間内のベクトルです。 同様のベクトルが近接してクラスター化されるため、一致を見つけることは、クエリベクトルに最も近いベクトルを見つけ、関連するドキュメントを検索結果として返すのと同じです。

たとえば、クエリ要求がホテルに関する場合、モデルは、ホテルに関するドキュメントを表すベクトルのクラスター内のどこかに存在するベクトルにクエリをマップします。 類似度メトリックに基づいて、クエリに最も似ているベクトルを特定すると、最も関連性の高いドキュメントが決まります。

完全な KNN に対してベクトルフィールドのインデックスが作成されると、クエリは "all neighbors" に対して実行されます。 HNSW 用にインデックスが作成されたフィールドの場合、検索エンジンは HNSW グラフを使用して、ベクトルインデックス内のノードのサブセットを検索します。

## HNSW グラフの作成

インデックス作成中、検索サービスは HNSW グラフを構築します。 HNSW グラフに新しいベクトルのインデックスを作成する目的は、効率的なニアレストネイバー検索できるような方法でグラフ構造に追加することです。 次の手順は、このプロセスをまとめたものです。

1. 初期化: 空の HNSW グラフ、または新しいインデックスでない場合は既存の HNSW グラフから開始します。
2. エントリ ポイント: これは階層グラフの最上位レベルであり、インデックス作成の開始点として機能します。
3. グラフへの追加: 階層レベルが異なると、グラフの細分性が異なり、レベルが高いほどグローバルになり、レベルが低いほど細かく表示されます。グラフ内の各ノードは、ベクトル ポイントを表します。
  - 各ノードは、近隣にある最大  $m$  個のネイバーに接続されます。これが  $m$  パラメーターです。
  - 候補接続と見なされるデータ ポイントの数は、`efConstruction` パラメーターによって管理されます。この動的リストは、アルゴリズムが考慮する既存のグラフ内の最も近いポイントのセットを形成します。`efConstruction` 値が大きいほど、多くのノード数が考慮され、多くの場合、各ベクトルのオーナル近傍が高密度になります。
  - これらの接続では、構成された類似性 `metric` を使用して距離を決定します。一部の接続は、さまざまな階層レベルで接続する "長距離" 接続であり、検索効率を向上させるショートカットをグラフに作成します。
4. グラフの枝刈りと最適化: これは、すべてのベクトルのインデックス作成後に発生する可能性があり、HNSW グラフのナビゲート性と効率が向上します。

## クエリ時の HNSW グラフ内の移動

ベクトル クエリは、一致をスキャンするために階層グラフ構造内を移動します。プロセスの手順の概要を次に示します:

1. 初期化: アルゴリズムは、階層グラフの最上位レベルで検索を開始します。このエントリ ポイントには、検索の開始点として機能するベクトルのセットが含まれています。
2. トラバーサル: 次に、グラフをレベルごとに走査し、最上位レベルから下位レベルに移動し、コサイン類似性など、構成された距離メトリックに基づいてクエリ ベクトルに近い候補ノードを選択します。
3. 枝刈り: 効率を向上させるために、アルゴリズムは、ニアレーストネイバーを含む可能性が高いノードのみを考慮して、検索領域を刈り込みます。これは、潜在的な候補の優先順位キューを維持し、検索が進むにつれてそれを更新することによつ

て達成されます。このキューの長さは、パラメーター `efSearch` によって構成されます。

4. 絞り込み: アルゴリズムがより低く、より細かいレベルに移行すると、HNSW は クエリの近くでより多くのネイバーを考慮し、ベクトルの候補セットを絞り込み、精度を向上させます。
5. 完了: 検索は、ニアレストネイバーの必要な数が特定された場合、または他の停止条件が満たされたときに完了します。このニアレストネイバーのこの必要な数は、クエリ時間パラメーター `k` によって制御されます。

## 類似性の測定に使用される類似性メトリック

アルゴリズムは、類似性を評価する候補ベクトルを検索します。このタスクを実行するために、類似性メトリック計算では、候補ベクトルとクエリベクトルを比較し、類似性を測定します。アルゴリズムは、検索された最も類似しているベクトルの順序付けされたセットを追跡し続け、アルゴリズムが完了したときにランク付けされた結果セットを形成します。

[+] テーブルを展開する

メトリック	説明
<code>cosine</code>	このメトリックは、2つのベクトル間の角度を測定し、ベクトルの長さの違いによる影響を受けません。数学的には、2つのベクトル間の角度を計算します。コサインは <a href="#">Azure OpenAI 埋め込みモデル</a> で使用される類似性メトリックであるため、Azure OpenAI を使用している場合は、ベクトル構成で <code>cosine</code> を指定します。
<code>dotProduct</code>	このメトリックは、2つのベクトルの各ペアの長さと、それらの間の角度の両方を測定します。数学的には、ベクトルの大きさとそれらの間の角度の積を計算します。正規化されたベクトルの場合、これは <code>cosine</code> 類似性と同じですが、パフォーマンスは若干高くなります。
<code>euclidean</code>	(別名 <code>l2 norm</code> ) このメトリックは、2つのベクトル間のベクトル長の差を測定します。数学的には、2つのベクトルの差の L2 ノルムである 2つのベクトル間のユークリッド距離を計算します。

## ベクトル検索結果のスコア

スコアが計算されて、それぞれの一致に割り当てられ、最も高い一致が `k` の結果として返されます。 `@search.score` プロパティにスコアが含まれています。次の表は、スコアが該当する範囲を示しています。

検索メソッド	パラメーター	スコアリング メトリック	Range
ベクトル検索	@search.score	コサイン	0.333 - 1.00

`cosine` メトリックについては、計算された `@search.score` がクエリ ベクトルとドキュメント ベクトルの間のコサイン値ではないことに注意することが重要です。代わりに、Azure AI Search では、スコア関数が単調に減少するような変換が適用されます。つまり、スコア値は、類似性が悪化すると常に値が減少します。この変換により、検索スコアをランク付け目的で使用できます。

類似性スコアにはいくつかの微妙な違いがあります:

- コサイン類似性は、2つのベクトル間の角度のコサインとして定義されます。
- コサイン距離は `1 - cosine_similarity` として定義されます。

単調に減少する関数が作成されるように `@search.score` は `1 / (1 + cosine_distance)` として定義されます。

合成値の代わりにコサイン値が必要な開発者は、数式を使用して、検索スコアをコサイン距離に戻すことができます:

C#

```
double ScoreToSimilarity(double score)
{
    double cosineDistance = (1 - score) / score;
    return -cosineDistance + 1;
}
```

元のコサイン値を持たせておくと、低品質の結果をトリミングするためのしきい値を設定するカスタム ソリューションにおいて有用です。

## 関連性のチューニングに関するヒント

関連性のある結果が得られない場合は、[クエリの構成](#)の変更を試してください。ベクトル クエリには、スコアリング プロファイルやフィールド、用語ブーストなどの特定のチューニング機能はありません。

- [チャネル サイズと重複](#)について実験します。チャネル サイズを大きくし、チャネル間のコンテキストまたは継続性を維持するのに十分な重複を確保します。

- HNSW の場合は、近接グラフの内部構成を変更するさまざまなレベルの `efConstruction` を試します。既定値は 400 です。範囲は 100 ~ 1,000 です。
- チャット モデルを使用している場合は、`k` の結果を増やして、より多くの検索結果をチャット モデルにフィードします。
- セマンティック ランク付けを使用して[ハイブリッド クエリ](#)を試します。ベンチマーク テストでは、この組み合わせが一貫して最も関連性の高い結果をもたらしました。

## 次のステップ<sup>°</sup>

- [クイックスタートを試す](#)
- [埋め込みの詳細](#)
- [データ チャンクの詳細](#)

# Reciprocal Rank Fusion (RRF) を使用したハイブリッド検索での関連性スコアリング

[アーティクル] • 2023/10/26

## ① 重要

ハイブリッド検索では、現在パブリック プレビュー段階にある**ベクター機能を補足の利用規約** のもとに使用します。

Reciprocal Rank Fusion (RRF) は、以前にランク付けされた複数の結果の検索スコアを評価して、統合された結果セットを生成するアルゴリズムです。 Azure Cognitive Search では、並列で実行される 2 つ以上のクエリがある場合は、常に RRF が使用されます。各クエリがランク付けされた結果セットを生成し、RRF はランク付けをマージして、クエリ応答で返される 1 つの結果セットに均質化するために使用されます。 RRF が必要なシナリオの例としては、[ハイブリッド検索](#)、同時に実行される複数のベクター クエリなどがあります。

RRF は、"逆順位" の概念に基づいています。これは、検索結果のリスト内の最初の関連ドキュメントのランクの逆数です。この手法の目標は、元のランキング内の項目の位置を考慮して、複数のリストで上位にランク付けされた項目により高い重要性を与えることです。これにより、最終的なランク付けの全体的な品質と信頼性が向上し、複数の順序付けされた検索結果を融合するタスクに役立ちます。

## RRF ランク付けのしくみ

RRF は、複数の方法から検索結果を取得し、結果の各ドキュメントに逆順位スコアを割り当て、スコアを組み合わせて新しいランク付けを作成することで機能します。この概念では、複数の検索方法で上位の位置に表示されるドキュメントはより関連性が高い可能性があるため、結合された結果の上位にランク付けされます。

RRF プロセスの簡単な説明を次に示します。

- 並列で実行される複数のクエリからランク付けされた検索結果を取得します。
- ランク付けされた各リストの結果に対して逆順位スコアを割り当てます。 RRF は、各結果セットの一致ごとに新しい `@search.score` を生成します。 検索結果のドキュメントごとに、エンジンはリスト内の位置に基づいて逆順位スコアを割り

当てます。スコアは  $1/(rank + k)$  として計算されます。ここで、`rank` はリスト内のドキュメントの位置であり、`k` は定数で、60 などの小さな値に設定されている場合に最適に実行されることが実験で確認されています。この `k` 値は RRF アルゴリズムの定数であり、最も近い近傍の数を制御する `k` とは完全に別の定数であることに注意してください。

3. スコアを結合します。各ドキュメントについて、エンジンは各検索システムから取得した逆順位スコアを合計し、各ドキュメントの結合スコアを生成します。
4. エンジンは、結合されたスコアに基づいてドキュメントをランク付けし、それらを並べ替えます。結果のリストは融合されたランキングです。

スコアリングには、インデックスで `searchable` としてマークされているフィールド、またはクエリで `searchFields` としてマークされているフィールドのみが使用されます。`retrievable` としてマークされたフィールド、またはクエリの `select` で指定されたフィールドのみが、検索スコアと共に検索結果に返されます。

## 並列クエリの実行

RRF は、複数のクエリが実行されるたびに使用されます。次の例は、並列クエリ実行が行われるクエリパターンを示しています。

- フルテキスト クエリと 1 つのベクター クエリ (単純なハイブリッド シナリオ) は、2 つのクエリ実行に相当します。
- フルテキスト クエリと、2 つのベクター フィールドを対象とする 1 つのベクター クエリは、3 つのクエリ実行に相当します。
- フルテキスト クエリと、5 つのベクター フィールドを対象とする 2 つのベクター クエリは、11 個のクエリ実行に相当します

## ハイブリッド検索結果のスコア

結果がランク付けされるたびに、`@search.score` プロパティには結果の順序付けに使用される値が含まれます。スコアは、方法ごとに異なるランク付けアルゴリズムによって生成されます。各アルゴリズムには、独自の範囲と大きさがあります。

次の表に、各関連性ランク付けアルゴリズムの各一致で返されるスコアリング プロパティ、アルゴリズム、スコアの範囲を示します。

検索メソッド	パラメーター	スコアリングアルゴリズム	Range
フルテキスト検索 (full-text search)	<code>@search.score</code>	BM25 アルゴリズム	上限なし。
ベクトル検索	<code>@search.score</code>	HNSW アルゴリズム。HNSW 構成で指定された類似性メトリックを使用します。	0.333 - 1.00 (Cosine)、Euclidean と DotProduct の場合は 0 から 1。
ハイブリッド検索	<code>@search.score</code>	RRF アルゴリズム	上限は結合されるクエリの数によって制限され、各クエリは RRF スコアに対して最大約 1 を寄与します。たとえば、3 つのクエリをマージすると、2 つの検索結果のみがマージされる場合よりも RRF スコアが高くなります。
セマンティックランク付け	<code>@search.rerankerScore</code>	セマンティックランク付け	1.00 - 4.00

セマンティックランク付けは RRF には関係していません。そのスコア (`@search.rerankerScore`) は、常にクエリ応答で個別に報告されます。セマンティックランク付けでは、フルテキスト検索結果とハイブリッド検索結果を再ランク付けでき、それらの検索結果には意的豊富なコンテンツを持つフィールドが含まれていると仮定しています。

## ハイブリッド クエリ応答のランク付けされた結果の数

既定では、改ページを使用していない場合、検索エンジンは、フルテキスト検索では上位 50 位のランクの一致、ベクトル検索では最も類似した `k` 個の一致を返します。ハイブリッド クエリでは、`top` によって応答の結果の数が決まります。既定に基づいて、統合結果セットの上位 50 にランク付けされた一致が返されます。

多くの場合、検索エンジンは `top` や `k` よりも多くの結果を見つけます。より多くの結果を返すには、ページング パラメーター `top`、`skip`、`next` を使用します。ページングは、各論理ページ上の結果の数を決定し、完全なペイロードを導く方法です。

フルテキスト検索には、最大 1,000 件の一致という制限が適用されます（「[API 応答の制限](#)」を参照）。1,000 件の一致が見つかると、検索エンジンはそれ以上の検索を行いません。

詳細については、「[検索結果の操作方法](#)」を参照してください。

## 関連項目

- [ハイブリッド検索の詳細を確認する](#)
- [ベクトル検索の詳細を確認する](#)

# Azure AI Search のセキュリティの概要

[アーティクル] • 2024/04/11

この記事では、データと操作を保護する Azure AI Search のセキュリティ機能について説明します。

## データ フロー (ネットワーク トラフィック パターン)

Azure AI Search サービスは Azure でホストされ、通常はパブリック ネットワーク接続を使用してクライアント アプリケーションからアクセスされます。このようなパターンとなることが多いですが、他のトラフィック パターンにも注意する必要があります。開発環境と運用環境をセキュリティ保護するには、すべてのエントリ ポイントと送信トラフィックについて理解しておく必要があります。

Azure AI Search には、3 つの基本的なネットワーク トラフィック パターンがあります。

- クライアントによって行われる、検索サービスへのインバウンド要求 (主要なパターン)
- 検索サービスによって発行される、Azure やそれ以外の場所の他のサービスへのアウトバウンド要求
- セキュリティ保護された Microsoft バックボーン ネットワークを介して行われる内部サービス間の要求

## 受信トラフィック

検索サービス エンドポイントを対象とする受信要求には、次が含まれます。

- 検索サービスでオブジェクトを作成、読み取り、更新、または削除する
- 検索ドキュメントのインデックスを読み込む
- インデックスのクエリ
- インデクサーまたはスキルセットの実行をトリガーする

[REST API](#) のページで、検索サービスによって処理される受信要求の全範囲が説明されています。

少なくとも、すべての受信要求は、次のいずれかのオプションを使用して認証される必要があります。

- キーベースの認証 (デフォルト)。受信要求が、有効な API キーを提供します。

- ロールベースのアクセス制御。認可は、検索サービスの Microsoft Entra ID とロールの割り当て経由で行われます。

さらに、[ネットワークセキュリティ機能](#)を追加して、エンドポイントへのアクセスをさらに制限できます。IP ファイアウォールでの受信の規則、またはパブリックインターネットから検索サービスを完全に遮断するプライベートエンドポイントのいずれかを作成することができます。

## 内部トラフィック

内部要求は、Microsoft によってセキュリティで保護され、管理されます。これらの接続を構成または制御することはできません。ネットワークアクセスをロックダウンしている場合、顧客が内部トラフィックを構成できないため、顧客からのアクションは必要ありません。

内部トラフィックは次で構成されます。

- タスク (Microsoft Entra ID 経由の認証と認可、Azure Monitor に送信されたリソースログ、Azure Private Link を使用した[プライベートエンドポイント接続](#)など) のサービス間呼び出し。
- [組み込みスキル](#)のための Azure AI サービス API への要求。
- [セマンティックランク付け](#)をサポートする機械学習モデルに対して行われた要求。

## 送信トラフィック

送信要求は、ユーザーがセキュリティで保護および管理できます。送信要求は、検索サービスから他のアプリケーションに向けて発生します。通常、これらの要求は、クエリ時にテキストベースのインデックス作成、スキルベースの AI エンリッチメント、ベクター化を行うために、インデクサーによって行われます。送信要求には、読み取りと書き込みの両方の操作が含まれます。

次の一覧は、セキュリティで保護された接続を構成できる送信要求の完全な列挙です。検索サービスは、検索自体のため、およびインデクサーまたはカスタム スキルのために要求を行います。

[+] [テーブルを展開する](#)

操作	シナリオ
インデクサー	外部データ ソースに接続してデータを取得します。 詳細については、「 <a href="#">Azure ネットワークセキュリティで保護されたコンテンツへのインデクサー アクセス</a> 」を参照してください。

操作	シナリオ
インデクサー	Azure Storage に接続して、 <a href="#">ナレッジストア</a> 、 <a href="#">キャッシュされたエンリッチメント</a> 、 <a href="#">デバッグ セッション</a> を持続させます。
カスタム スキル	サービス外でホストされている外部コードを実行している Azure Functions、Azure Web アプリ、またはその他のアプリに接続します。スキルセットの実行中に、外部処理に対する要求が送信されます。
インデクサーと垂直統合	Azure OpenAI とデプロイされた埋め込みモデルに接続するか、カスタム スキルを経由して、指定する埋め込みモデルに接続します。検索サービスは、インデックス作成中にベクター化のために埋め込みモデルにテキストを送信します。
ベクター化	クエリ時に Azure OpenAI またはその他の埋め込みモデルに接続して、ベクター検索化のために <a href="#">ユーザー テキスト文字列をベクターに変換</a> します。
検索サービス	機密データの暗号化および復号化に使用される <a href="#">カスタマー マネージド暗号化キー</a> を取得するために、Azure Key Vault に接続します。

送信接続は、キーまたはデータベース ログインを含むリソースのフルアクセス接続文字列を使用して確立することも、Microsoft Entra ID とロールベースのアクセスを使用している場合は[マネージド ID](#)を使用して確立することもできます。

ファイアウォールの内側にある Azure リソースにアクセスするには、[他の Azure リソースに検索サービス要求を許可する受信規則を作成](#)します。

Azure Private Link によって保護された Azure リソースにアクセスするには、インデクサーが接続を確立するために使用的する[共有プライベート リンクを作成](#)します。

## 同じリージョンの検索サービスとストレージ サービスの例外

Azure Storage と Azure AI Search が同じリージョンにある場合、ネットワーク トラフィックはプライベート IP アドレス経由でルーティングされ、Microsoft バックボーン ネットワークで発生します。プライベート IP アドレスが使用されるため、ネットワークセキュリティ用に IP ファイアウォールまたはプライベート エンドポイントを構成することはできません。

次のいずれかの方法を使用して、同じリージョンの接続を構成します。

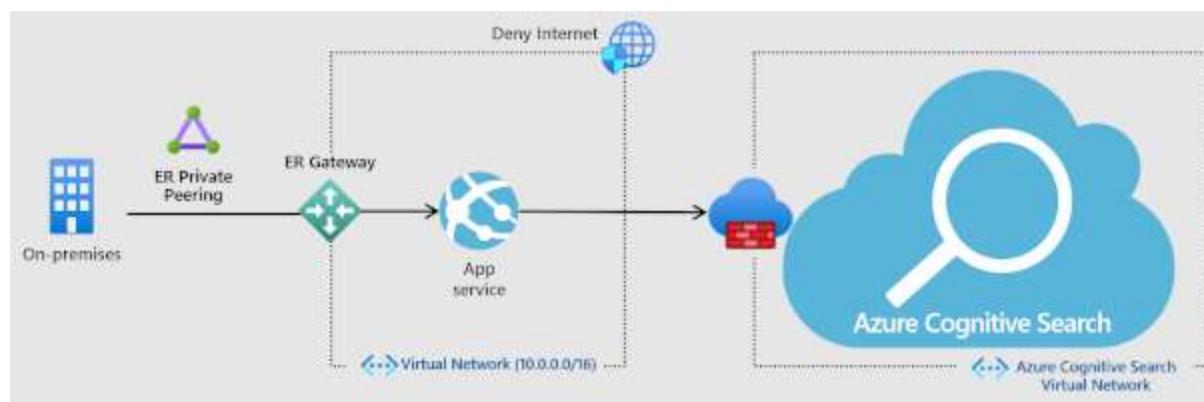
- [信頼されたサービスの例外](#)
- [リソース インスタンス ルール](#)

## ネットワークのセキュリティ

ネットワークセキュリティは、ネットワークトラフィックに制御を適用することにより、未承認のアクセスや攻撃からリソースを保護します。 Azure AI Search は、未承認のアクセスに対する防御の前線になり得るネットワーク機能をサポートしています。

## IP ファイアウォール経由の受信接続

検索サービスは、パブリック IP アドレスを使用して、アクセスを許可するパブリックエンドポイントによりプロビジョニングされます。パブリックエンドポイントを経由するトラフィックを制限するには、特定の IP アドレスまたは IP アドレスの特定の範囲から要求を許可する受信ファイアウォール規則を作成します。すべてのクライアント接続は、許可された IP アドレスを使用して行う必要があります。それ以外の場合、接続は拒否されます。



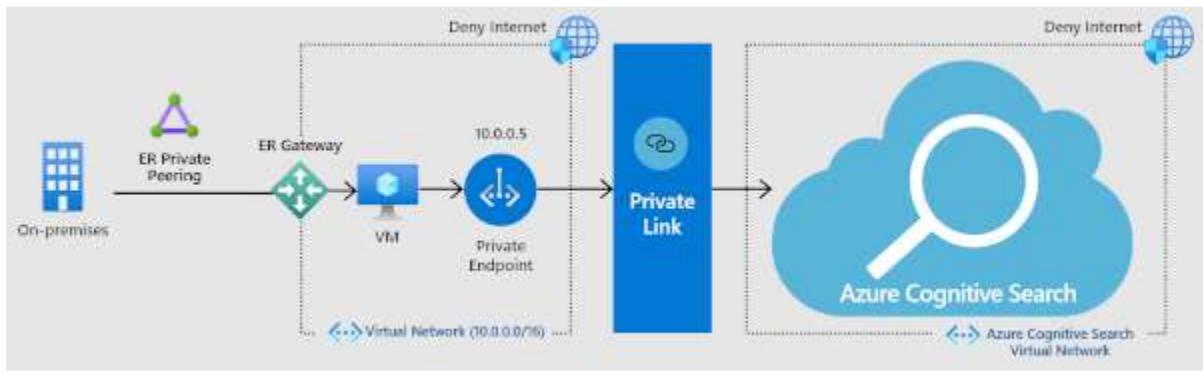
ファイアウォールアクセスを構成するには、ポータルを使用します。

または、管理 REST API を使用します。API バージョン 2020-03-13 以降では、[IpRule](#) パラメーターを指定することで、検索サービスへのアクセスを付与する IP アドレスを個別に、あるいは範囲で特定することで、サービスへのアクセスを制限できます。

## プライベートエンドポイントへの受信接続 (ネットワーク分離、インターネットトラフィックなし)

より強力なセキュリティには、Azure AI Search の[プライベートエンドポイント](#)を確立して、[仮想ネットワーク](#)上のクライアントが [Private Link](#) を介して、検索インデックス内のデータに安全にアクセスできるようにします。

プライベートエンドポイントでは、検索サービスに接続するために仮想ネットワークのアドレス空間の IP アドレスが使用されます。クライアントと検索サービス間のネットワークトラフィックは、仮想ネットワークおよび Microsoft バックボーン ネットワーク上のプライベートリンクを経由することで、パブリックインターネットでの露出を排除します。仮想ネットワークを使用すると、オンプレミス ネットワークやインターネットで、リソース間の安全な通信が可能になります。



このソリューションは最も安全ですが、追加のサービスを使用すると、さらなるコストがかかります。そのため、使用の前に利点の詳細を明確に理解しておく必要があります。コストの詳細については、[価格ページ](#)を参照してください。これらのコンポーネントを連携させる方法の詳細については、[こちらのビデオ](#)をご覧ください。プライベート エンドポイント オプションの説明は、ビデオの 5:48 から始まります。エンドポイントを設定する方法については、[Azure AI Search でのプライベート エンドポイントの作成](#)に関するページを参照してください。

## 認証

検索サービス宛ての要求が承認された後も、要求が許可されているかどうかを判断する認証と認可を受ける必要があります。Azure AI Search は、2 つの方法をサポートします。

- Microsoft Entra 認証では、認証された ID として（要求ではなく）呼び出し元が確立されます。Azure ロールの割り当てが認可を決定します。
- キーベースの認証は、API キーにより（呼び出し元のアプリやユーザーではなく）要求に対して行われます。このキーは、要求が信頼できるソースからの要求であることを証明する、ランダムに生成された数字と文字で構成される文字列です。キーは要求ごとに必要です。有効なキーの送信は、要求が信頼されたエンティティのものであることの証明と見なされます。

両方の認証方法を使用することも、検索サービスで使用可能にしない[方法を無効にする](#)こともできます。

## 承認

Azure AI Search には、サービス管理とコンテンツ管理のためのさまざまな認可モデルが用意されています。

## Azure サービス管理

リソース管理は、Microsoft Entra テナント内の[ロールベースのアクセス制御](#)によって認可されます。

Azure AI Search では、Resource Manager を使用して、サービスの作成または削除、API キーの管理、サービスのスケーリング、セキュリティの構成が行われます。そのため、[ポータル](#)、[PowerShell](#)、[Management REST API](#) のどれを使用しているかにかかわらず、Azure で割り当てられているロールによって、これらのタスクを実行できるユーザーが決定されます。

[3 つの基本ロール](#) (所有者、共同作成者、閲覧者) が検索サービスの管理に適用されます。ロールの割り当ては、サポートされている任意の方法 (ポータル、PowerShell など) を使用して行うことができ、サービス全体に適用されます。

### ① 注意

Azure 全体のメカニズムを使用して、サブスクリプションまたはリソースをロックし、管理者権限を持つユーザーが検索サービスを誤って、または許可なく削除しないようにすることができます。 詳細については、[リソースのロックによる予期せぬ削除の防止](#)に関するページを参照してください。

## コンテンツへのアクセスを承認する

コンテンツ管理とは、検索サービスで作成およびホストされるオブジェクトを指します。

- ロールベースの認可の場合、[Azure のロールの割り当て](#)を使用して、操作に対する読み書きアクセスを確立します。
- キーベースの認可の場合、[API キー](#)と修飾されたエンドポイントによってアクセスが決定されます。エンドポイントはサービス自体、インデックスコレクション、特定のインデックス、ドキュメントコレクション、特定のドキュメントなどである場合があります。連結されている場合、エンドポイント、操作 (作成または更新要求など)、キーの種類 (管理者またはクエリ) によってコンテンツへのアクセスと操作が認可されます。

## インデックスへのアクセスの制限

Azure ロールを使用している場合は、プログラムによって実行される限り、[個々のインデックスに対するアクセス許可を設定](#)できます。

キーを使用すると、サービスに対する[管理者キー](#)を持っている人は誰でも、そのサービスのインデックスの読み取り、変更、削除を行えます。インデックスが誤って削除さ

れたり、悪意によって削除されたりすることを防止するうえで、コード資産の社内ソース管理は、望ましくないインデックスの削除または変更を元に戻すための解決策になります。 Azure AI Search は可用性を確保するためにクラスター内のフェールオーバーを備えていますが、インデックスの作成または読み込みに使用される専用コードを格納したり実行したりしません。

インデックス レベルでセキュリティ境界を必要とするマルチテナント ソリューションの場合、通常、アプリケーション コードの中間層でインデックス分離を処理します。マルチテナントのユース ケースの詳細については、「[マルチテナント SaaS アプリケーションと Azure AI Search の設計パターン](#)」を参照してください。

## ドキュメントへのアクセスの制限

"行レベル セキュリティ" とも呼ばれるドキュメント レベルのユーザー アクセス許可是、Azure AI Search でネイティブにはサポートされていません。 Azure Cosmos DB など、行レベルセキュリティを提供する外部システムからデータをインポートする場合、Azure AI Search によってインデックス付けされているため、そのようなアクセス許可はデータと共に転送されません。

検索結果のコンテンツに対するアクセス許可が必要な場合、ユーザー ID に基づいてドキュメントを含めるか、除外するフィルターを適用する手法があります。この回避策では、グループまたはユーザー ID を表す文字列フィールドをデータ ソースに追加します。このフィールドは、インデックスでフィルター可能にできます。次の表では、承認されていないコンテンツの検索結果をトリミングする 2 つのアプローチについて説明しています。

[+] [テーブルを展開する](#)

アプローチ	説明
<a href="#">ID フィルターに基づいたセキュリティによるトリミング</a>	ユーザー ID アクセス制御を実装する基本的なワークフローについて記載しています。また、インデックスへのセキュリティ ID の追加について取り上げているほか、そのフィールドに対してフィルター処理を行い、禁止されているコンテンツの結果をトリミングする方法について説明しています。
<a href="#">Microsoft Entra ID に基づくセキュリティトリミング</a>	この記事は前の記事を拡張したものであり、Azure クラウド プラットフォームの <a href="#">無料サービス</a> の 1 つである Microsoft Entra ID から ID を取得する手順について説明しています。

## データの保存場所

お客様は、検索サービスを設定するときに、顧客データがどこで格納および処理されるかを決定する場所またはリージョンを選びます。構成した機能が別の Azure リソースに依存し、そのリソースが別のリージョンにプロビジョニングされている場合を除き、Azure AI Search は、指定したリージョンの外部に顧客データを格納しません。

現在、検索サービスが書き込む唯一の外部リソースは Azure Storage です。ストレージアカウントは、お客様が指定したストレージアカウントであり、任意のリージョンに存在する可能性があります。[エンリッチメント キャッシュ](#)、[デバッグ セッション](#)、[ナレッジストア](#)のいずれかの機能を使用する場合、検索サービスは Azure Storage に書き込みます。

## データ所在地のコミットメントに対する例外

オブジェクト名は、お客様が選んだリージョンまたは場所以外の場所に格納され、処理されます。お客様は、名前のフィールドに機密データを配置することや、これらのフィールドに機密データが格納されるように設計したアプリケーションを作成することはできません。このデータは、Microsoft がサービスのサポートを提供するために使うテレメトリログに表示されます。オブジェクト名には、インデックス、インデクサー、データソース、スキルセット、リソース、コンテナー、キー コンテナーストアの名前が含まれます。

テレメトリログは 1 年半保持されます。その間、Microsoft は次の条件下でオブジェクト名にアクセスして参照する場合があります。

- 問題の診断、機能の改善、バグの修正を行います。このシナリオでは、データ アクセスは内部のみであり、サードパーティがアクセスすることはありません。
- サポート中、問題への迅速な解決策を提供し、必要に応じて製品チームを昇格させるために、この情報が使用される場合があります

## データ保護

ストレージ層には、インデックスやシノニム マップ、およびインデクサー、データソース、スキルセットの定義など、ディスクに保存されるすべてのサービス マネージドコンテンツに対するデータ暗号化が組み込まれています。サービス マネージド暗号化は、長期データストレージと一時データストレージの両方に適用されます。

必要に応じて、インデックス付きコンテンツの補足暗号化用にカスタマー マネージドキー (CMK) を追加し、保存データの二重暗号化を行うことができます。2020 年 8 月 1 日以降に作成されたサービスでは、CMK 暗号化は一時ディスクの短期データにも拡張されています。

## 転送中のデータ

Azure AI Search では、暗号化は接続時および転送時に開始されます。パブリックインターネット上の検索サービスでは、Azure AI Search によって HTTPS ポート 443 がリッスンされます。すべてのクライアントとサービスの間の接続では、TLS 1.2 暗号化が使用されます。より前のバージョン (1.0 または 1.1) はサポートされていません。

## 保存データ

検索サービスによって内部で処理されるデータについて、次の表で [データ暗号化モデル](#) を説明しています。ナレッジストア、インクリメンタルエンリッチメント、インデクサー ベースのインデックス作成などの一部の機能は、他の Azure サービスのデータ構造から読み書きされます。Azure Storage に依存するサービスでは、そのテクノロジの暗号化機能を使用できます。

[+] [テーブルを展開する](#)

モデル	キー	必要条件	制限	適用対象
サーバー側暗号化	Microsoft のマネージドキー	なし (組み込み)	なし。2018 年 1 月 24 日以降に作成されたコンテンツについては、すべてのリージョンのすべての階層で使用できます。	データディスクおよび一時ディスク上のコンテンツ (インデックスとシノニム マップ) と定義 (インデクサー、データソース、スキルセット)
サーバー側暗号化	カスタマー マネージド キー	Azure Key Vault	2020 年 8 月 1 日以降に作成されたコンテンツについては、特定のリージョンの請求対象階層で使用できます。	データディスク上のコンテンツ (インデックスとシノニム マップ)
サーバー側の完全暗号化	カスタマー マネージド キー	Azure Key Vault	2021 年 5 月 13 日以降の検索サービスについては、すべてのリージョンの請求対象階層で使用できます。	データディスクおよび一時ディスク上のコンテンツ (インデックスとシノニム マップ)

## サービス マネージド キー

サービス マネージド暗号化とは、256 ビット AES 暗号化<sup>1</sup>を使用する Microsoft 内部操作です。(2018 年 1 月より前に作成された) 完全に暗号化されていないインデックスに対する増分更新を含む、すべてのインデックス作成で自動的に行われます。

サービス マネージド暗号化は、長期および短期ストレージ上のすべてのコンテンツに適用されます。

## カスタマー マネージド キー (CMK)

カスタマー マネージド キーには、Azure Key Vault という別の請求対象のサービスが必要です。このリージョンは別であってもかまいませんが、Azure AI Search と同じサブスクリプションのものである必要があります。

CMK のサポートは、2 つのフェーズでロールアウトされました。最初のフェーズで検索サービスを作成した場合、CMK 暗号化は長期ストレージと特定のリージョンに制限されていました。2021 年 5 月以降の 2 番目のフェーズで作成されたサービスでは、任意のリージョンで CMK 暗号化を使用できます。2 番目のウェーブのロールアウトの一環として、コンテンツは長期ストレージと短期ストレージの両方で CMK 暗号化されます。CMK のサポートの詳細については、「[完全二重暗号化](#)」を参照してください。

CMK での暗号化を有効にすると、インデックスのサイズが増加し、クエリのパフォーマンスが低下します。これまでの観測に基づくと、実際のパフォーマンスはインデックスの定義やクエリの種類によって異なりますが、クエリ時間が 30 から 60 パーセント増加することが予想されます。パフォーマンスへの悪影響があるため、この機能を本当に必要とするインデックスでのみ有効にすることをお勧めします。 詳細については、[Azure AI Search でのカスタマー マネージド暗号化キーの構成に関するページ](#)を参照してください。

## セキュリティと管理

### API キーを管理する

API キーベースの認証に依存するということは、Azure のセキュリティのベストプラクティスに従って、定期的に管理者キーを再生成するための計画を立てる必要があることを意味します。Search サービスごとに最大 2 個の管理キーがあります。API キーのセキュリティと管理の詳細については、[API キーの作成と管理](#)に関する記事を参照してください。

### アクティビティとリソースのログ

Azure AI Search では、ユーザー ID はログに記録されないため、特定のユーザーに関する情報のログを参照することはできません。ただし、このサービスでは、ログの作成、読み取り、更新、削除の各操作がログに記録されるため、これらのログを他のログと関連付けて、特定のアクションの機関を理解できる場合があります。

Azure でアラートとログ記録インフラストラクチャを使用すると、クエリ ボリュームの急増や、予想されるワークロードから逸脱したその他のアクションを検出できます。ログの設定の詳細については、[ログ データの収集と分析](#)および[クエリ要求の監視](#)に関する記事を参照してください。

## 認定資格とコンプライアンス

Azure AI Search は通常の監査に参加し、パブリック クラウドと Azure Government の両方について、グローバル、リージョン、および業界固有のさまざまな標準に対して認定を受けています。完全な一覧については、公式の監査レポート ページから [Microsoft Azure Compliance Offerings ホワイトペーパー](#) をダウンロードしてください。

コンプライアンスのために、[Microsoft クラウド セキュリティ ベンチマーク](#)の安全性の高いベストプラクティスを、[Azure Policy](#) を使用して実装できます。Microsoft クラウド セキュリティ ベンチマークは、サービスやデータに対する脅威を軽減するために実行する必要のある主要なアクションにマップされる、セキュリティ コントロールに体系化された、セキュリティに関する推奨事項を集めたものです。現在は、[ネットワークセキュリティ](#)、ログ記録および監視、[データ保護](#)などを含む 12 のセキュリティ コントロールがあります。

Azure Policy は、Microsoft クラウド セキュリティ ベンチマークの標準を含む複数の標準に対するコンプライアンスの管理に役立つ、Azure に組み込まれた機能です。広く知られたベンチマークについては、コンプライアンス非対応の場合に使用できる、基準と実施可能な対応の両方の組み込みの定義が、Azure Policy によって提供されています。

Azure AI Search には、現在 1 つの定義が組み込まれています。これはリソース ログ用です。リソース ログが欠落している検索サービスを識別するポリシーを割り当てて、有効にできます。 詳細については、「[Azure AI Search 用の Azure Policy 規制コンプライアンス コントロール](#)」を参照してください。

## このビデオを観る

セキュリティ アーキテクチャと各機能カテゴリーの概要については、こちらのビデオをご覧ください。

<https://learn.microsoft.com/Shows/AI-Show/Azure-Cognitive-Search-Whats-new-in-security/player>

## 関連項目

- Azure セキュリティの基礎
- Azure Security ↗
- Microsoft Defender for Cloud

# Azure ネットワーク セキュリティで保護されたコンテンツへのインデクサー アクセス

[アーティクル] • 2024/05/06

概念に関するこの記事では、Azure リソースが Azure 仮想ネットワークにデプロイされている場合に、検索インデクサーがネットワーク セキュリティによって保護されているコンテンツにアクセスする方法について説明します。送信トラフィック パターンとインデクサー実行環境について説明します。また、Azure AI Search でサポートされるネットワーク保護と、セキュリティ戦略に影響を与える可能性のある要因についても説明します。最後に、Azure Storage はデータ アクセスと永続ストレージの両方に使用されるため、この記事では、[検索とストレージの接続](#)に固有のネットワークの考慮事項についても説明します。

代わりに詳細な手順をお探しですか? [インデクサー アクセスを許可するようにファイアウォール規則を構成する方法](#)または[プライベート エンドポイントを介して送信接続を行う方法](#)に関する記事を参照してください。

## インデクサーによってアクセスされるリソース

Azure AI 検索インデクサーは、次の 3 つの状況において、さまざまな Azure リソースへの送信呼び出しを行うことができます。

- インデックス作成中に外部のデータ ソースに接続する場合
- カスタム スキルを含むスキルセットを介して外部のカプセル化されたコードに接続する場合
- スキルセットの実行中に Azure Storage に接続してエンリッチメントをキャッシュしたり、デバッグ セッションの状態を保存したり、ナレッジストアに書き込んだりする場合

通常の実行でインデクサーがアクセスする可能性がある Azure リソースの種類を次の表に一覧表示します。

〔〕 テーブルを展開する

リソース	インデクサー実行内の目的
Azure Storage (BLOB、ADLS Gen 2、ファイル、テーブル)	データ ソース

リソース	インデクサー実行内の目的
Azure Storage (BLOB、テーブル)	スキルセット (エンリッチメントのキャッシュ、セッションのデバッグ、ナレッジストアのプロジェクト)
Azure Cosmos DB (さまざまな API)	データ ソース
Azure SQL データベース	データ ソース
Azure 仮想マシン上の SQL Server	データ ソース
SQL Managed Instance	データ ソース
Azure Functions	スキルセットに接続され、カスタム Web API スキルのホスティングに使用される

### ① 注意

インデクサーは、組み込みのスキルのために Azure AI サービスにも接続します。ただし、その接続は内部ネットワーク経由で行われ、制御下のネットワークプロビジョニングの対象なりません。

インデクサーは、次の方法を使用してリソースに接続します。

- パブリック エンドポイントと資格情報
- Azure Private Link を使用したプライベート エンドポイント
- 信頼されたサービスとしての接続
- IP アドレス指定を通して接続する

Azure リソースが仮想ネットワーク上にある場合は、プライベート エンドポイントまたは IP アドレス指定を使用して、データへのインデクサー接続を許可する必要があります。

## サポートされているネットワーク保護

Azure リソースは、Azure によって提供される任意の数のネットワーク分離メカニズムを使用して保護できます。リソースとリージョンに応じて Azure AI Search インデクサーは、次の表に示す制限付きで、IP ファイアウォールとプライベート エンドポイント経由で送信接続を行うことができます。

 [テーブルを展開する](#)

リソース	IP 制限	プライベート エンドポイント
Azure Storage のテキストベースのインデックス作成 (BLOB、ADLS Gen 2、ファイル、テーブル)	ストレージ アカウントと検索サービスが異なるリージョンにある場合にのみサポートされます。	サポートされています
Azure Storage の AI エンリッチメント (キャッシュ、デバッグ セッション、ナレッジ ストア)	ストレージ アカウントと検索サービスが異なるリージョンにある場合にのみサポートされます。	サポートされています
NoSQL 用 Azure Cosmos DB	サポートされています	サポートされています
Azure Cosmos DB for MongoDB	サポートされています	サポートされていない
Azure Cosmos DB for Apache Gremlin	サポートされています	サポートされていない
Azure SQL データベース	サポートされています	サポートされています
Azure 仮想マシン上の SQL Server	サポートされています	該当なし
SQL Managed Instance	サポートされています	該当なし
Azure Functions	サポートされています	Azure Functions の特定の層に対してのみサポートされます

## インデクサー実行環境

Azure AI Search には、ジョブの特性に基づいて処理を最適化する "インデクサー実行環境" の概念があります。2つの環境があります。IP ファイアウォールを使用して Azure リソースへのアクセスを制御している場合は、実行環境について理解しておくと、両方の環境を含む IP 範囲を設定するのに役立ちます。

指定されたインデクサーの実行に対し、Azure AI Search で、そのインデクサーを実行するための最適な環境が決定されます。インデクサーは、割り当てられているタスクの数と種類に応じて、2つの環境のどちらかで実行されます。

[+] テーブルを開く

実行環境	説明
プライベート	検索サービスの内部。プライベート環境で実行されているインデクサーは、コンピューティング リソースを、同じ検索サービス上の他のインデックス作成およびクエリのワー

実行 説明
環境
<p>一ト クロードと共有します。通常、この環境では、テキストベースのインデックス作成(スキルセットを使わない)を実行するインデクサーのみが実行されます。インデクサーとデータの間にプライベート接続を設定する場合は、これが使用できる唯一の実行環境です。</p> <p>マルチテナント追加料金なしで Microsoft によって管理およびセキュリティ保護されます。これは、ご自分の管理下にあるどのネットワーク プロビジョニングの対象にもなりません。この環境は、大量のコンピューティング処理を要する処理の負荷を軽減して、サービス固有のリソースをルーチン処理に残しておくために使います。リソースを大量に消費するインデクサー ジョブの例には、スキルセットのアタッチ、大規模なドキュメントの処理、大量のドキュメントの処理などがあります。</p>

## インデクサー実行の IP 範囲の設定

このセクションでは、どちらの実行環境からの要求でも許可する IP ファイアウォール構成について説明します。

Azure リソースがファイアウォールの内側に存在する場合は、インデクサー要求を発信できるすべての IP に対してインデクサー接続を許可する受信規則を設定します。これには、検索サービスで使用される IP アドレスと、マルチテナント環境で使用される IP アドレスが含まれます。

- 検索サービス(およびプライベート環境)の IP アドレスを取得するには、`nslookup`(または`ping`)で検索サービスの完全修飾ドメイン名(FQDN)を使用します。パブリック クラウドの検索サービスの FQDN は、`<service-name>.search.windows.net`です。
- インデクサーが実行される可能性のあるマルチテナント環境の IP アドレスを取得するには、`AzureCognitiveSearch` サービス タグを使用します。

[Azure サービス タグ](#)には、リージョンごとのマルチテナント環境の公開された IP アドレス範囲が含まれています。これらの IP は、[Discovery API](#) または[ダウンロード可能な JSON ファイル](#)を使用して調べることができます。IP 範囲はリージョン別に割り当てられるため、開始する前に検索サービスのリージョンを確認してください。

## Azure SQL の IP 規則の設定

マルチテナント環境の IP 規則を設定する場合、特定の SQL データ ソースで IP アドレスの指定に対する単純なアプローチがサポートされます。規則内のすべての IP アドレスを列挙する代わりに、`AzureCognitiveSearch` サービス タグを指定する[ネットワークセキュリティ グループ規則](#)を作成できます。

データ ソースが次のいずれかである場合は、サービス タグを指定できます。

- Azure 仮想マシン上の SQL Server
- SQL マネージド インスタンス

マルチテナント環境の IP 規則にサービス タグを指定した場合でも、`nslookup` から取得した、プライベート実行環境 (検索サービスそのものを意味する) の明示的な受信規則が必要であることに注意してください。

## 接続方法の選択

検索サービスは、仮想マシン上でネイティブに実行されている特定の仮想ネットワークにプロビジョニングすることはできません。一部の Azure リソースでは [仮想ネットワークサービス エンドポイント](#) が提供されますが、この機能は Azure AI 検索では提供されません。次のいずれかの方法の実装を計画してください。

[+] テーブルを展開する

アプローチ	詳細
Azure リソースへの受信接続をセキュリティで保護する	インデクサーによるデータの要求を許可する受信ファイアウォール規則を Azure リソースに構成します。ファイアウォール構成には、マルチテナント実行のサービス タグと検索サービスの IP アドレスを含める必要があります。 <a href="#">インデクサーへのアクセスを許可するファイアウォール規則の構成</a> に関する記事を参照してください。
Azure AI 検索と Azure リソースの間のプライベート接続	リソースへの接続のために検索サービスによって排他的に使用される共有プライベートリンクを構成します。接続は内部ネットワークを経由し、パブリックインターネットをバイパスします。リソースが完全にロックダウンされている場合 (保護された仮想ネットワークで実行されている場合、またはパブリック接続で使用できない場合)、プライベート エンドポイントが唯一の選択肢となります。 <a href="#">プライベート エンドポイントを経由した送信接続の作成</a> に関するページを参照してください。

プライベート エンドポイント経由の接続は、検索サービスのプライベート実行環境から開始する必要があります。

IP ファイアウォールの構成は無料です。 Azure Private Link に基づくプライベート エンドポイントは、課金に影響します。 詳細については、「[Azure Private Link の価格](#)」をご覧ください。

ネットワークセキュリティを構成したら、続いてロールの割り当てによって、どのユーザーとグループにデータと操作に対する読み取りと書き込みのアクセス権があるかを指定します。

# プライベート エンドポイントの使用に関する考慮事項

このセクションでは、プライベート接続オプションに絞って説明します。

- 共有プライベート リンクには、課金対象の検索サービスが必要です。その最小レベルは、テキストベースのインデックス作成向けの Basic、またはスキルベースのインデックス作成向けの Standard 2 (S2) のいずれかです。 詳細については、[プライベート エンドポイントの数に対するレベルの制限](#)に関する記事を参照してください。
- 共有プライベート リンクが作成されると、それは検索サービスで常にその特定の Azure リソースへのすべてのインデクサー接続に使用されます。 プライベート接続はロックされ、内部的に強制されます。 パブリック接続のためにプライベート接続をバイパスすることはできません。
- 課金対象の Azure Private Link リソースが必要です。
- サブスクリプション所有者がプライベート エンドポイント接続を承認する必要があります。
- インデクサーのマルチテナント実行環境をオフにする必要があります。

これを行うには、インデクサーの `executionEnvironment` を `"Private"` に設定します。 この手順により、すべてのインデクサー実行が、検索サービス内でプロビジョニングされたプライベート環境に限定されます。 この設定のスコープは、検索サービスではなくインデクサーです。 すべてのインデクサーをプライベート エンドポイント経由で接続する場合は、それぞれに次の構成が必要です。

JSON

```
{  
    "name" : "myindexer",  
    ... other indexer properties  
    "parameters" : {  
        ... other parameters  
        "configuration" : {  
            ... other configuration properties  
            "executionEnvironment": "Private"  
        }  
    }  
}
```

リソースに対して承認されたプライベート エンドポイントができると、`private` に設定されているインデクサーは、Azure リソース用に作成および承認されたプライベート リンクを介してアクセスを取得しようとします。

Azure AI 検索で、プライベート エンドポイントの呼び出し元に適切なロールの割り当てがなされていることが検証されます。たとえば、読み取り専用のアクセス許可があるストレージ アカウントへのプライベート エンドポイント接続を要求した場合、この呼び出しは拒否されます。

プライベート エンドポイントが承認されていない場合、またはインデクサーがプライベート エンドポイント接続を使用していない場合は、インデクサーの実行履歴に `TransientFailure` エラー メッセージが表示されます。

## トークン認証を使用してネットワーク セキュリティを補完する

ファイアウォールとネットワーク セキュリティは、データと操作への未承認のアクセスを防ぐための最初の手順です。次の手順となるのが承認です。

ロールベースのアクセスをお勧めします。この場合、Microsoft Entra ID のユーザーとグループは、サービスへの読み取りと書き込みのアクセス権を決定するロールに割り当てられます。組み込みロールの説明とカスタム ロールを作成する手順については、[ロールベースのアクセス制御を使用した Azure AI 検索への接続](#)に関するページを参照してください。

キーベースの認証が必要ない場合は、API キーを無効にし、ロールの割り当てのみを使用することをお勧めします。

## ネットワークで保護されたストレージ アカウントへのアクセス

検索サービスでは、インデックスとシノニム リストを格納します。ストレージを必要とするその他の機能の場合、Azure AI Search は Azure Storage に依存します。エンリッチメント キャッシュ、デバッグ セッション、ナレッジストアは、このカテゴリに分類されます。各サービスの場所と、ストレージ用のネットワーク保護によって、データアクセス戦略が決まります。

## 同じリージョンのサービス

Azure Storage では、ファイアウォール経由でアクセスするには、要求が別のリージョンから送信されている必要があります。Azure Storage と Azure AI Search が同じリージョンにある場合は、検索サービスのシステム ID の下にあるデータにアクセスして、ストレージ アカウントの IP 制限を回避できます。

システム ID を使用してデータ アクセスをサポートするには、次の 2 つのオプションがあります。

- Azure Storage で[信頼済みサービス](#)として実行し、[信頼済みサービスの例外](#)を使用するように検索を構成します。
- Azure リソースからの受信要求を許可する[リソース インスタンス ルール](#)を Azure Storage で構成します。

上記のオプションは認証用の Microsoft Entra ID によって異なります。つまり、Microsoft Entra ログインで接続する必要があります。現在、ファイアウォール経由の同じリージョン接続では、[Azure AI Search システム割り当てマネージド ID](#) のみがサポートされています。

## 異なるリージョンのサービス

検索とストレージが異なるリージョンにある場合は、前述のオプションを使用するか、お使いのサービスからの要求を許可する IP ルールを設定できます。ワークフローによっては、次のセクションで説明するように、複数の実行環境のルールを設定する必要がある場合があります。

## 次のステップ

Azure 仮想ネットワークにデプロイされたソリューションのインデクサー データアクセス オプションについて理解したら、次の手順として次のハウツー記事のいずれかを確認します。

- [プライベート エンドポイントへのインデクサー接続を確立する方法](#)
- [IP ファイアウォールを介してインデクサー接続を確立する方法](#)

# Azure AI Search の Azure Policy 規制コンプライアンスコントロール

[アーティクル] • 2024/02/29

Azure Policy を使用して Microsoft クラウド セキュリティ ベンチマークのレコメンデーションを適用している場合は、準拠していないサービスを識別して修正するためにポリシーを作成できることは、既にご存じかもしれません。このようなポリシーは、カスタムの場合もあれば、よく知られたベスト プラクティスのためのコンプライアンス条件と適切なソリューションを提供する組み込み定義に基づく場合もあります。

Azure AI Search の場合、現在、以下に示す 1 つの組み込み定義があり、ポリシー割り当てで使用できます。組み込みは、ログ記録と監視のためのものです。 [作成するポリシー](#) でこの組み込み定義を使用すると、システムによってリソース ログのない検索サービスがスキャンされ、それに応じて有効にされます。

Azure Policy の規制コンプライアンスにより、さまざまなコンプライアンス基準に関連するコンプライアンス ドメインおよびセキュリティ コントロールに対して、"組み込み" と呼ばれる、Microsoft によって作成および管理されるイニシアチブ定義が提供されます。このページでは、Azure AI Search のコンプライアンス ドメインおよびセキュリティ コントロールの一覧を示します。セキュリティ コントロールの組み込みを個別に割り当てることで、Azure リソースを特定の基準に準拠させることができます。

各組み込みポリシー定義のタイトルは、Azure portal のポリシー定義にリンクしています。 [ポリシーのバージョン] 列のリンクを使用すると、[Azure Policy GitHub リポジトリ](#) のソースを表示できます。

## ① 重要

各コントロールは、1 つ以上の [Azure Policy](#) 定義に関連付けられています。これらのポリシーは、コントロールの [コンプライアンスの評価](#) に役立つ場合があります。ただし、多くの場合、コントロールと 1 つ以上のポリシーとの間には、一对一、または完全な一致はありません。そのため、Azure Policy での準拠は、ポリシー自体のみを指しています。これによって、コントロールのすべての要件に完全に準拠していることが保証されるわけではありません。また、コンプライアンス標準には、現時点でどの Azure Policy 定義でも対応されていないコントロールが含まれています。したがって、Azure Policy でのコンプライアンスは、全体のコンプライアンス状態の部分的ビューでしかありません。これらのコンプライアンス標準に対するコントロールと Azure Policy 規制コンプライアンス定義の間の関連付けは、時間の経過と共に変わることがあります。

# CIS Microsoft Azure Foundations Benchmark

## 1.3.0

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - CIS Microsoft Azure Foundations Benchmark 1.3.0](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[CIS Microsoft Azure Foundations Benchmark](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
5 ログと監視	5.3	診断ログがそれをサポートするすべてのサービスで有効になっていることを確認する	Search サービスのリソースログを有効にする必要がある	<a href="#">5.0.0</a>

# CIS Microsoft Azure Foundations Benchmark

## 1.4.0

すべての Azure サービスで使用可能な Azure Policy の組み込みがこのコンプライアンス標準にどのように対応しているのかを確認するには、[CIS v1.4.0 に関する Azure Policy の規制コンプライアンスの詳細](#)に関する記事を参照してください。このコンプライアンス標準の詳細については、[CIS Microsoft Azure Foundations Benchmark](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
5 ログと監視	5.3	診断ログがサポートされているすべてのサービスに対して有効になっていることを確認する。	Search サービスのリソースログを有効にする必要がある	<a href="#">5.0.0</a>

# CIS Microsoft Azure Foundations Benchmark

## 2.0.0

すべての Azure サービスで使用可能な Azure Policy の組み込みがこのコンプライアンス標準にどのように対応しているのかを確認するには、[CIS v2.0.0 に関する Azure Policy の規制コンプライアンスの詳細](#)に関する記事を参照してください。このコンプライアンス標準の詳細については、[CIS Microsoft Azure Foundations Benchmark](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
5	5.4	Azure Monitor リソース ログが、それをサポートするすべてのサービスで有効になっていることを確認します	Search サービスのリソース ログを有効にする必要がある	5.0.0

## CMMC レベル 3

すべての Azure サービスで使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - CMMC レベル 3](#)に関する記事をご覧ください。このコンプライアンス標準の詳細については、[サイバーセキュリティ成熟度モデル認定 \(CMMC\)](#) に関するドキュメントをご覧ください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	AC.1.001	情報システムへのアクセスを、許可されているユーザー、許可されているユーザーの代わりに動作するプロセス、およびデバイス (他の情報システムを含む) に制限する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
アクセス制御	AC.1.002	情報システムへのアクセスを、許可されているユーザーが実行を許可されているトランザクションおよび機能の種類に制限する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
アクセス制御	AC.2.016	承認された認可に従って CUI のフローを制御する。	Azure AI Services リソースでネットワー	3.1.0

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
ク アクセスを制限する必要がある				
構成管理	CM.3.068	不要なプログラム、関数、ポート、プロトコル、およびサービスの使用を制限、無効化、または禁止する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
システムと通信の保護	SC.1.175	組織システムの外部境界と主要な内部境界で、通信 (つまり、組織システムによって送受信される情報) を監視、制御、および保護する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
システムと通信の保護	SC.3.183	ネットワーク通信トラフィックを既定で拒否し、ネットワーク通信トラフィックを例外的に許可する (つまり、すべて拒否し、例外的に許可する)。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0

## FedRAMP High

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - FedRAMP High](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[FedRAMP High](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	AC-2	アカウント管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0
アクセス制御	AC-2 (1)	システム アカウント管理の自動化	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0
アクセス制御	AC-2 (7)	ロールベースのスキーム	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	AC-3	アクセスの適用	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure AI Services リソースでネットワーク アクセスを制限する必要がある	3.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスではパブリックネットワーク アクセスを無効にする必要がある	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17	リモート アクセス	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	AC-17	リモート アクセス	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17 (1)	自動監視/制御	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	AC-17 (1)	自動監視/制御	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
監査とアカウントアビリティ	AU-6 (4)	一元的なレビューと分析	Search サービスのリソース ログを有効にする必要があります	5.0.0 ↗
監査とアカウントアビリティ	AU-6 (5)	統合またはスキヤンと監視機能	Search サービスのリソース ログを有効にする必要があります	5.0.0 ↗
監査とアカウントアビリティ	AU-12	監査の生成	Search サービスのリソース ログを有効にする必要があります	5.0.0 ↗
監査とアカウントアビリティ	AU-12 (1)	システム全体または時間相関の監査証跡	Search サービスのリソース ログを有効にする必要があります	5.0.0 ↗
識別と認証	IA-2	識別と認証 (組織のユーザ)	Azure AI Services リソースのキー アクセスが無効になっている必要があります	1.1.0 ↗

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
	一)		す (□一カル認証を無効にする) ↗	
識別と認証	IA-4	識別子の管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一カル認証を無効にする) ↗	1.1.0 ↗
システムと通信の保護	SC-7	境界保護	Azure AI Services リソースでネットワーク アクセスを制限する必要がある ↗	3.1.0 ↗
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスではパブリックネットワーク アクセスを無効にする必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure AI Services リソースでネットワーク アクセスを制限する必要がある ↗	3.1.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスではパブリックネットワーク アクセスを無効にする必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗

## FedRAMP Moderate

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - FedRAMP Moderate](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[FedRAMP Moderate](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	AC-2	アカウント管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一から認証を無効にする) ↗	1.1.0 ↗
アクセス制御	AC-2 (1)	システムアカウント管理の自動化	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一から認証を無効にする) ↗	1.1.0 ↗
アクセス制御	AC-2 (7)	ロールベースのスキーム	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一から認証を無効にする) ↗	1.1.0 ↗
アクセス制御	AC-3	アクセスの適用	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一から認証を無効にする) ↗	1.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure AI Services リソースでネットワークアクセスを制限する必要がある ↗	3.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスではパブリックネットワークアクセスを無効にする必要がある ↗	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗
アクセス制御	AC-17	リモートアクセス	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
アクセス制御	AC-17	リモートアクセス	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗
アクセス制御	AC-17 (1)	自動監視/制御	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
アクセス制御	AC-17 (1)	自動監視/制御	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗
監査とアカウンタビリティ	AU-12	監査の生成	Search サービスのリソースログを有効にする必要があります ↗	5.0.0 ↗

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
識別と認証	IA-2	識別と認証 (組織のユーザー)	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一カル認証を無効にする) ↗	1.1.0 ↗
識別と認証	IA-4	識別子の管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (□一カル認証を無効にする) ↗	1.1.0 ↗
システムと通信の保護	SC-7	境界保護	Azure AI Services リソースでネットワーク アクセスを制限する必要がある ↗	3.1.0 ↗
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスではパブリックネットワーク アクセスを無効にする必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure AI Services リソースでネットワーク アクセスを制限する必要がある ↗	3.1.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスではパブリックネットワーク アクセスを無効にする必要がある ↗	1.0.0 ↗
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある ↗	1.0.0 ↗

## HIPAA HITRUST 9.2

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - HIPAA HITRUST 9.2](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[HIPAA HITRUST 9.2](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
12 監査ログと監視	1208.09aa3System.1-09.aa	1208.09aa3System.1-09.aa 09.10 監視	Search サービスのリソース ログを有効にする必要がある	5.0.0 ↗

## Microsoft クラウド セキュリティ ベンチマーク

Microsoft クラウド セキュリティ ベンチマークでは、Azure 上のクラウド ソリューションをセキュリティで保護する方法に関する推奨事項が提供されます。このサービスを完全に Microsoft クラウド セキュリティ ベンチマークにマップする方法については、「[Azure Security Benchmark mapping files ↗](#)」(Azure セキュリティ ベンチマークのマッピング ファイル) を参照してください。

すべての Azure サービスに対して使用可能な Azure Policy 組み込みを、このコンプライアンス基準に対応させる方法については、[Azure Policy の規制コンプライアンス - Microsoft クラウド セキュリティ ベンチマーク](#)に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
ネットワークのセキュリティ	NS-2	ネットワーク制御を使用してクラウド サービスをセキュリティで保護します	Azure AI Services リソースでネットワーク アクセスを制限する必要がある	3.1.0 ↗
ID 管理	IM-1	一元的な ID および認証システムを使用する	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
ログと脅威検出	LT-3	セキュリティ調査のためのログを有効にする	Search サービスのリソース ログを有効にする必要がある	5.0.0 ↗

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - NIST SP 800-171 R2](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[NIST SP 800-171 R2](#) に関するページを参照してください。

[+] テーブルを展開する

[ドメイン]	コントロール	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	3.1.1	承認されているユーザー、承認されているユーザーの代わりに動作するプロセス、およびデバイス (他のシステムを含む) へのシステムアクセスを制限する。	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	3.1.1	承認されているユーザー、承認されているユーザーの代わりに動作するプロセス、およびデバイス (他のシステムを含む) へのシステムアクセスを制限する。	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	3.1.1	承認されているユーザー、承認されているユーザーの代わりに動作するプロセス、およびデバイス (他のシステムを含む) へのシステムアクセスを制限する。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	3.1.12	リモート アクセス セッションの監視および制御を行う。	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	3.1.12	リモート アクセス セッションの監視および制御を行う。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	3.1.13	リモート アクセス セッションの機密性を保護するため暗号化メカニズムを採用する。	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	3.1.13	リモート アクセス セッションの機密性を保護するため暗号化メカニズムを採用する。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗

[ドメイン]	コントロール ロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	3.1.14	管理対象のアクセス制御ポイントを介してリモート アクセスをルーティングする。	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0 ↗
アクセス制御	3.1.14	管理対象のアクセス制御ポイントを介してリモート アクセスをルーティングする。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	3.1.2	システム アクセスを、許可されたユーザーが実行を許可されているトランザクションおよび機能の種類に限定する。	Azure AI Services リソースのキー アクセスが無効になっている必要があります(ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	3.1.3	承認された認可に従って CUI のフローを制御する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0 ↗
アクセス制御	3.1.3	承認された認可に従って CUI のフローを制御する。	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0 ↗
アクセス制御	3.1.3	承認された認可に従って CUI のフローを制御する。	Azure Cognitive Search サービスではパブリック ネットワーク アクセスを無効にする必要がある	1.0.0 ↗
アクセス制御	3.1.3	承認された認可に従って CUI のフローを制御する。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
システムと通信の保護	3.13.1	組織システムの外部境界と主要な内部境界で、通信(つまり、組織システムによって送受信される情報)を監視、制御、および保護する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0 ↗
システムと通信の保護	3.13.1	組織システムの外部境界と主要な内部境界で、通信(つまり、組織システムによって送受信される情報)を監視、制御、および保護する。	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0 ↗
システムと通信	3.13.1	組織システムの外部境界と主要な内部境界で、通信(つまり、組織シ	Azure Cognitive Search サービスではパブリック ネ	1.0.0 ↗

[ドメイン]	コントロール ロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
信の保護		システムによって送受信される情報を監視、制御、および保護する。	ネットワークアクセスを無効にする必要がある	
システムと通信の保護	3.13.1	組織システムの外部境界と主要な内部境界で、通信 (つまり、組織システムによって送受信される情報を監視、制御、および保護する。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0
システムと通信の保護	3.13.2	組織のシステム内で効果的な情報セキュリティを促進するアーキテクチャ設計、ソフトウェア開発手法、システムエンジニアリングの原則を採用する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
システムと通信の保護	3.13.2	組織のシステム内で効果的な情報セキュリティを促進するアーキテクチャ設計、ソフトウェア開発手法、システムエンジニアリングの原則を採用する。	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0
システムと通信の保護	3.13.2	組織のシステム内で効果的な情報セキュリティを促進するアーキテクチャ設計、ソフトウェア開発手法、システムエンジニアリングの原則を採用する。	Azure Cognitive Search サービスではパブリックネットワークアクセスを無効にする必要がある	1.0.0
システムと通信の保護	3.13.2	組織のシステム内で効果的な情報セキュリティを促進するアーキテクチャ設計、ソフトウェア開発手法、システムエンジニアリングの原則を採用する。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0
システムと通信の保護	3.13.5	内部ネットワークから物理的または論理的に分離されている、公的にアクセス可能なシステムコンポーネントのサブネットワークを実装する。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
システムと通信の保護	3.13.5	内部ネットワークから物理的または論理的に分離されている、公的にアクセス可能なシステムコンポーネントのサブネットワークを実装する。	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0
システムと通信	3.13.5	内部ネットワークから物理的または論理的に分離されている、公的に	Azure Cognitive Search サービスではパブリックネ	1.0.0

[ドメイン]	コントロール ロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
信の保護		にアクセス可能なシステム コンポーネントのサブネットワークを実装する。	ネットワーク アクセスを無効にする必要がある	
システムと通信の保護	3.13.5	内部ネットワークから物理的または論理的に分離されている、公的にアクセス可能なシステム コンポーネントのサブネットワークを実装する。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0
システムと通信の保護	3.13.6	ネットワーク通信トラフィックを既定で拒否し、ネットワーク通信トラフィックを例外的に許可する(つまり、すべて拒否し、例外的に許可する)。	Azure AI Services リソースでネットワーク アクセスを制限する必要がある	3.1.0
システムと通信の保護	3.13.6	ネットワーク通信トラフィックを既定で拒否し、ネットワーク通信トラフィックを例外的に許可する(つまり、すべて拒否し、例外的に許可する)。	Azure Cognitive Search サービスではパブリック ネットワーク アクセスを無効にする必要がある	1.0.0
監査とアカウントアビリティ	3.3.1	違法または承認されていないシステム アクティビティの監視、分析、調査、および報告を有効にするために必要な範囲までシステム監査ログとレコードを作成して保持する	Search サービスのリソース ログを有効にする必要がある	5.0.0
監査とアカウントアビリティ	3.3.2	個々のシステム ユーザーのアクションからそのユーザーまで一意に確実にたどれるようにし、彼らが自分のアクションの責任を負えるようにする。	Search サービスのリソース ログを有効にする必要がある	5.0.0
識別と認証	3.5.1	システム ユーザー、ユーザーの代わりに動作するプロセス、およびデバイスを特定する。	Azure AI Services リソース のキー アクセスが無効になっている必要があります(ローカル認証を無効にする)	1.1.0
識別と認証	3.5.2	組織システムへのアクセスを許可するための前提条件として、ユーザー、プロセス、またはデバイスの ID を認証(または検証)する。	Azure AI Services リソース のキー アクセスが無効になっている必要があります(ローカル認証を無効にする)	1.1.0

[ドメイン] [コントロール ID]	コントロールのタイトル	ポリシー (Azure portal)	ポリシー のバージ ョン (GitHub)
識別と認証 3.5.5	定義された期間、識別子の再利用を防止する。	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) ↗	1.1.0 ↗
識別と認証 3.5.6	定義された非アクティブな期間の経過後に識別子を無効にする。	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) ↗	1.1.0 ↗

## NIST SP 800-53 Rev. 4

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - NIST SP 800-53 Rev. 4](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[NIST SP 800-53 Rev. 4](#) に関するページを参照してください。

[+] テーブルを展開する

[ドメイン] [コントロール ID]	コントロール のタイトル	ポリシー (Azure portal)	ポリシーのバージ ョン (GitHub)
アクセス制御 AC-2	アカウント管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) ↗	1.1.0 ↗
アクセス制御 (1)	システム アカウント管理の自動化	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) ↗	1.1.0 ↗
アクセス制御 (7)	ロールベースのスキーム	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) ↗	1.1.0 ↗
アクセス制御 AC-3	アクセスの適用	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) ↗	1.1.0 ↗

[ドメイン]	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	AC-4	情報フローの適用	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスではパブリックネットワークアクセスを無効にする必要がある	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17	リモートアクセス	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17	リモートアクセス	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17 (1)	自動監視/制御	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17 (1)	自動監視/制御	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
監査とアカウンタビリティ	AU-6 (4)	一元的なレビューと分析	Search サービスのリソースログを有効にする必要がある	5.0.0 ↗
監査とアカウンタビリティ	AU-6 (5)	統合またはスキャンと監視機能	Search サービスのリソースログを有効にする必要がある	5.0.0 ↗
監査とアカウンタビリティ	AU-12	監査の生成	Search サービスのリソースログを有効にする必要がある	5.0.0 ↗
監査とアカウンタビリティ	AU-12 (1)	システム全体または時間相関の監査証跡	Search サービスのリソースログを有効にする必要がある	5.0.0 ↗
識別と認証	IA-2	識別と認証(組織のユーザー)	Azure AI Services リソースのキーアクセスが無効になっている必要があります(ローカル認証を無効にする)	1.1.0 ↗

[ドメイン]	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
識別と認証	IA-4	識別子の管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) <a href="#">↗</a>	1.1.0 <a href="#">↗</a>
システムと通信の保護	SC-7	境界保護	Azure AI Services リソースでネットワークアクセスを制限する必要がある <a href="#">↗</a>	3.1.0 <a href="#">↗</a>
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある <a href="#">↗</a>	1.0.0 <a href="#">↗</a>
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスではパブリックネットワークアクセスを無効にする必要がある <a href="#">↗</a>	1.0.0 <a href="#">↗</a>
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある <a href="#">↗</a>	1.0.0 <a href="#">↗</a>
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure AI Services リソースでネットワークアクセスを制限する必要がある <a href="#">↗</a>	3.1.0 <a href="#">↗</a>
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある <a href="#">↗</a>	1.0.0 <a href="#">↗</a>
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスではパブリックネットワークアクセスを無効にする必要がある <a href="#">↗</a>	1.0.0 <a href="#">↗</a>
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある <a href="#">↗</a>	1.0.0 <a href="#">↗</a>

## NIST SP 800-53 Rev. 5

すべての Azure サービスに対して使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy の規制コンプライアンス - NIST SP 800-53 Rev. 5](#) に関するページを参照してください。このコンプライアンス標準の詳細については、[NIST SP 800-53 Rev. 5](#) に関するページを参照してください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
アクセス制御	AC-2	アカウント管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	AC-2 (1)	システム アカウント管理の自動化	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	AC-2 (7)	特権ユーザー アカウント	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	AC-3	アクセスの適用	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure AI Services リソースでネットワーク アクセスを制限する必要がある	3.1.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスではパブリックネットワーク アクセスを無効にする必要がある	1.0.0 ↗
アクセス制御	AC-4	情報フローの適用	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17	リモート アクセス	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	AC-17	リモート アクセス	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
アクセス制御	AC-17 (1)	監視および制御	Azure Cognitive Search サービスでは、プライベートリンクをサポートする SKU を使用する必要がある	1.0.0 ↗
アクセス制御	AC-17 (1)	監視および制御	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0 ↗
監査とアカウ	AU-6	一元的なレビ	Search サービスのリソース ログを有効	5.0.0 ↗

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
セキュリティとコンプライアンス	(4)	ユーザーと分析	にする必要がある	
監査とアカウント管理	AU-6 (5)	監査レコードの統合分析	Search サービスのリソース ログを有効にする必要がある	5.0.0
監査とアカウント管理	AU-12	監査レコードの生成	Search サービスのリソース ログを有効にする必要がある	5.0.0
監査とアカウント管理	AU-12 (1)	システム全体および時間に関する監査証跡	Search サービスのリソース ログを有効にする必要がある	5.0.0
識別と認証	IA-2	識別と認証 (組織のユーザー)	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0
識別と認証	IA-4	識別子の管理	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0
システムと通信の保護	SC-7	境界保護	Azure AI Services リソースでネットワーク アクセスを制限する必要がある	3.1.0
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスでは、プライベート リンクをサポートする SKU を使用する必要がある	1.0.0
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスではパブリック ネットワーク アクセスを無効にする必要がある	1.0.0
システムと通信の保護	SC-7	境界保護	Azure Cognitive Search サービスはプライベート リンクを使用する必要がある	1.0.0
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure AI Services リソースでネットワーク アクセスを制限する必要がある	3.1.0
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスでは、プライベート リンクをサポートする SKU を使用する必要がある	1.0.0
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスではパブリック ネットワーク アクセスを無効にする必要がある	1.0.0
システムと通信の保護	SC-7 (3)	アクセス ポイント	Azure Cognitive Search サービスはプライベート リンクを使用する必要がある	1.0.0

# NL BIO Cloud Theme

すべての Azure サービスで使用可能な Azure Policy の組み込みがこのコンプライアンス標準にどのように対応しているのかを確認するには、「[NL BIO Cloud Theme に関する Azure Policy の規制コンプライアンスの詳細](#)」を参照してください。このコンプライアンス標準の詳細については、「[ベースライン情報セキュリティ政府サイバーセキュリティ - デジタル政府 \(digitaleoverheid.nl\)](#)」を参照してください。

[+] テーブルを展開する

[ドメイン]	コントロール	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
U.07.1 データ分離 - 分離	U.07.1	データの永続的な分離は、マルチテナントアーキテクチャの 1 つです。パッチは制御された方法で実現されます。	Azure AI Services リソースでネットワークアクセスを制限する必要がある	3.1.0
U.07.1 データの分離 - 分離	U.07.1	データの永続的な分離は、マルチテナントアーキテクチャの 1 つです。パッチは制御された方法で実現されます。	Azure Cognitive Search サービスでは、プライベートリンクをサポートするSKUを使用する必要がある	1.0.0
U.07.1 データの分離 - 分離	U.07.1	データの永続的な分離は、マルチテナントアーキテクチャの 1 つです。パッチは制御された方法で実現されます。	Azure Cognitive Search サービスではパブリックネットワークアクセスを無効にする必要がある	1.0.0
U.07.1 データの分離 - 分離	U.07.1	データの永続的な分離は、マルチテナントアーキテクチャの 1 つです。パッチは制御された方法で実現されます。	Azure Cognitive Search サービスはプライベートリンクを使用する必要がある	1.0.0
U.07.3 データの分離 - 管理機能	U.07.3	U.07.3 - CSC データおよび/または暗号化キーを表示あるいは変更する権限は、制御された方法で付与され、使用状況がログに記録されます。	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする)	1.1.0
U.10.2 IT サービスとデータ	U.10.2	CSP の責任の下、アクセス権が管理者に付与されます。	Azure AI Services リソースのキー アクセスが無効になっている必要があります	1.1.0

ドメイン	コントロールのタイトル	ポリシーのバージョン
ID	(Azure portal)	(GitHub)
U.10.3 IT サービスとデータへのアクセス - ユーザー	IT サービスとデータにアクセスできるのは、認証された機器を持つユーザーだけです。	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) <a href="#">↗</a>
U.10.5 IT サービスとデータへのアクセス - 適格性	IT サービスとデータへのアクセスは技術的な手段によって制限され、実装されています。	Azure AI Services リソースのキー アクセスが無効になっている必要があります (ローカル認証を無効にする) <a href="#">↗</a>
U.15.1 ログ記録と監視 - イベントの記録	ポリシー規則の違反は、CSP と CSC によって記録されます。	Search サービスのリソース ログを有効にする必要がある <a href="#">↗</a>

## インド準備銀行の銀行向けの IT フレームワーク v2016

すべての Azure サービスで使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy 規制コンプライアンス - RBI ITF Banks v2016](#) に関する記事を参照してください。このコンプライアンス標準の詳細については、[RBI ITF Banks v2016 \(PDF\) \[↗\]\(#\)](#) を参照してください。

[\[+\] テーブルを展開する](#)

Domain	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
ID			
フィッシング詐欺対策	フィッシング詐欺対策-14.1	Azure AI Services リソースでネットワーク アクセスを制限する必要がある <a href="#">↗</a>	3.1.0 <a href="#">↗</a>

## RMIT マレーシア

すべての Azure サービスで使用可能な Azure Policy 組み込みがこのコンプライアンス標準にどのように対応するのかを確認するには、[Azure Policy 規制コンプライアンス - RMIT マレーシア](#) に関する記事をご覧ください。このコンプライアンス標準の詳細については、[RMIT マレーシア \[↗\]\(#\)](#) に関するドキュメントをご覧ください。

[+] テーブルを展開する

Domain	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
デジタルサービスのセキュリティ	10.66	デジタルサービスのセキュリティ - 10.66	Search サービスの診断設定をイベントハブにデプロイする	2.0.0 ↗
デジタルサービスのセキュリティ	10.66	デジタルサービスのセキュリティ - 10.66	Search サービスの診断設定を Log Analytics ワークスペースにデプロイする	1.0.0 ↗

## SWIFT CSP-CSCF v2021

すべての Azure サービスで使用できる Azure Policy の組み込みが、このコンプライアンス標準にどのように対応するかを確認するには、「[SWIFT CSP-CSCF v2021 についての Azure Policy の規制コンプライアンスの詳細](#)」を参照してください。このコンプライアンス標準の詳細については、「[SWIFT CSP CSCF v2021 ↗](#)」を参照してください。

[+] テーブルを展開する

[ドメイン]	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
システムまたはトランザクション レコードに対する異常なアクティビティの検出	6.4	ログ記録と監視	Search サービスのリソース ログを有効にする必要がある	5.0.0 ↗

## SWIFT CSP-CSCF v2022

すべての Azure サービスで使用できる Azure Policy の組み込みが、このコンプライアンス標準にどのように対応するかを確認するには、「[SWIFT CSP-CSCF v2022 についての Azure Policy の規制コンプライアンスの詳細](#)」を参照してください。このコンプライアンス標準の詳細については、「[SWIFT CSP CSCF v2022 ↗](#)」を参照してください。

[+] テーブルを展開する

ドメイン	コントロール ID	コントロールのタイトル	ポリシー (Azure portal)	ポリシーのバージョン (GitHub)
6.システムまたはトランザクションコードに対する異常なアクティビティの検出	6.4	セキュリティ イベントを記録し、ローカルの SWIFT 環境内の異常なアクションと操作を検出する。	Search サービスのリソース ログを有効にする必要があります。	<a href="#">5.0.0</a>

## 次のステップ<sup>°</sup>

- Azure Policy の規制コンプライアンスの詳細を確認します。
- Azure Policy GitHub リポジトリ[のビルトイン](#)を参照します。

# Azure Cognitive Search の Azure セキュリティ ベースライン

[アーティクル] • 2023/09/22

このセキュリティ ベースラインは、[Microsoft クラウド セキュリティ ベンチマーク バージョン 1.0](#) のガイダンスをAzure Cognitive Searchに適用します。 Microsoft クラウド セキュリティ ベンチマークでは、Azure 上のクラウド ソリューションをセキュリティで保護する方法に関する推奨事項が提供されます。 コンテンツは、Microsoft クラウド セキュリティ ベンチマークと、Azure Cognitive Searchに適用できる関連ガイダンスによって定義されたセキュリティ制御によってグループ化されます。

このセキュリティ ベースラインとその推奨事項は、Microsoft Defender for Cloud を使用して監視できます。 Azure Policy 定義は、[クラウド ポータルの Microsoft Defender] ページの [規制コンプライアンス] セクションに一覧表示されます。

機能に関連するAzure Policy定義がある場合は、Microsoft クラウド セキュリティ ベンチマークの制御と推奨事項への準拠を測定するのに役立つ、このベースラインに一覧表示されます。一部の推奨事項では、特定のセキュリティ シナリオを有効にするために有料Microsoft Defenderプランが必要になる場合があります。

## ① 注意

Azure Cognitive Searchに適用されない機能は除外されています。 microsoft クラウド セキュリティ ベンチマークに完全にマップ Azure Cognitive Search 方法については、[完全な Azure Cognitive Search セキュリティ ベースライン マッピング ファイル](#) を参照してください。

## セキュリティ プロファイル

セキュリティ プロファイルは、Azure Cognitive Search の影響の大きい動作をまとめたものです。これにより、セキュリティに関する考慮事項が高まる可能性があります。

[+] テーブルを展開する

サービス動作属性	値
製品カテゴリ	AI+ML、モバイル、Web
お客様は HOST/OS にアクセスできます	アクセス権なし

サービス動作属性	値
サービスは顧客の仮想ネットワークにデプロイできます	False
顧客のコンテンツを保存する	○

## ネットワークのセキュリティ

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: ネットワーク セキュリティ](#)」を参照してください。

### NS-1: ネットワーク セグメント化の境界を確立する

#### 機能

##### Virtual Network 統合

**説明:** サービスは、顧客のプライベート Virtual Network (VNet) へのデプロイをサポートします。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

##### ネットワーク セキュリティ グループのサポート

**説明:** サービス ネットワーク トラフィックは、サブネット上のネットワーク セキュリティ グループルールの割り当てを尊重します。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## NS-2: ネットワーク制御を使用してクラウド サービスをセキュリティで保護する

### 機能

#### Azure Private Link

**説明:** ネットワーク トラフィックをフィルター処理するためのサービス ネイティブ IP フィルタリング機能 (NSG や Azure Firewall と混同しないように)。 詳細については、[こちらを参照してください。](#)

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	False	Customer

**機能に関する注意事項:** プライベート エンドポイント経由の送信接続については、「[プライベート エンドポイント経由で送信接続を行う](#)」を参照してください。

**構成ガイダンス:** プライベート エンドポイントをデプロイして、リソースのプライベート アクセス ポイントを確立します。 検索サービス向けのパブリック エンドポイント上のすべての接続をブロックします。 仮想ネットワークからのデータの流出をブロックし、仮想ネットワークのセキュリティを強化します。

**リファレンス:** [Azure Cognitive Searchへのセキュリティで保護された接続用のプライベート エンドポイントを作成する](#)

### パブリック ネットワーク アクセスの無効化

**説明:** サービスでは、サービス レベルの IP ACL フィルター規則 (NSG または Azure Firewall ではなく) または [パブリック ネットワーク アクセスの無効化] トグルスイッチを使用して、パブリック ネットワーク アクセスを無効にできます。 詳細については、[こちらを参照してください。](#)

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** Azure Cognitive Searchでは、Azure 仮想ネットワーク セキュリティ グループで見つかる IP 規則と同様に、ファイアウォール経由の受信アクセスの IP 規則がサポートされています。IP 規則を利用してことで、検索サービスのアクセスを、承認された一連のマシンとクラウド サービスに制限できます。これらの承認された一連のマシンやサービスから検索サービスに格納されているデータにアクセスするには、引き続き呼び出し側で有効な認可トークンを提示する必要があります。

**リファレンス:** [Azure Cognitive Search用に IP ファイアウォールを構成する](#)

## ID 管理

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: ID 管理](#)」を参照してください。

## IM-1: 一元的な ID および認証システムを使用する

### 機能

#### データ プレーン アクセスに必要な Azure AD Authentication

**説明:** サービスでは、データ プレーン アクセスに Azure AD 認証を使用できます。 詳細については、[こちらを参照してください](#)。

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	True	Microsoft

**構成ガイダンス:** 既定のデプロイでこれが有効になっているので、追加の構成は必要ありません。

#### データ プレーン アクセスのローカル認証方法

**説明:** ローカルユーザー名やパスワードなど、データ プレーン アクセスでサポートされるローカル認証方法。 詳細については、[こちらを参照してください](#)。

## [+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	False	Customer

**機能に関するメモ:** ローカル認証方法またはアカウントの使用は避けてください。これらは可能な限り無効にする必要があります。代わりに、可能な場合は Azure AD を使用して認証します。

**構成ガイダンス:** Cognitive Search では、主要な認証方法としてキーベースの認証が使用されます。インデックスの作成やクエリを実行する要求など、検索サービス エンド ポイントへの受信要求の場合、一般に使用可能な認証オプションは API キーだけです。

**リファレンス:** [Azure Cognitive Search 認証に API キーを使用する](#)

## IM-3: アプリケーション ID を安全かつ自動的に管理する

### 機能

#### マネージド ID

**説明:** データプレーンアクションでは、マネージド ID を使用した認証がサポートされます。 詳細については、[こちらを参照してください](#)。

## [+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** 可能な場合は、サービスプリンシパルの代わりに Azure マネージド ID を使用します。これにより、Azure Active Directory (Azure AD) 認証をサポートする Azure サービスとリソースに対して認証できます。マネージド ID の資格情報は、プラットフォームによって完全に管理、ローテーション、保護されており、ソースコードまたは構成ファイル内でハードコーディングされた資格情報を使用せずに済みます。

**リファレンス:** [Azure Active Directory を使用して検索アプリへのアクセスを承認する](#)

### サービス プリンシパル

**説明:** データプレーンでは、サービスプリンシパルを使用した認証がサポートされています。 詳細については、[こちらを参照してください](#)。

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** この機能の構成に関する現在の Microsoft ガイダンスはありません。 organizationがこのセキュリティ機能を構成するかどうかを確認して確認してください。

## IM-7: 条件に基づいてリソースへのアクセスを制限する

### 機能

#### データプレーンへの条件付きアクセス

**説明:** データプレーンアクセスは、Azure AD 条件付きアクセス ポリシーを使用して制御できます。 詳細については、[こちらを参照してください](#)。

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** ワークロード内の Azure Active Directory (Azure AD) 条件付きアクセスに適用できる条件と条件を定義します。 特定の場所からのアクセスのブロックや許可、危険なサインイン動作のブロック、特定のアプリケーションに対するorganizationマネージド デバイスの要求など、一般的なユースケースを検討してください。

## IM-8: 資格情報とシークレットの公開を制限する

### 機能

#### Azure Key Vault での、サービス資格情報とシークレットの統合とストレージのサポート

**説明:** データプレーンでは、資格情報とシークレットストアに対する Azure Key Vault のネイティブな使用がサポートされています。 詳細については、[こちらを参照してください。](#)

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## 特権アクセス

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: 特権アクセス](#)」を参照してください。

### PA-1: 高い特権を持つ/管理者ユーザーを分離して制限する

#### 機能

#### ローカル 管理 アカウント

**説明:** サービスには、ローカル管理アカウントの概念があります。 詳細については、[こちらを参照してください。](#)

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

### PA-7: Just Enough Administration (最小限の特権の原則) に従う

#### 機能

## Azure RBAC for Data Plane

**説明:** Azure Role-Based Access Control (Azure RBAC) を使用して、サービスのデータ プレーン アクションへのアクセスを管理できます。 詳細については、[こちらを参照してください。](#)

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** Azure は、プラットフォーム上で実行されているすべてのサービスに対してグローバル ロールベースのアクセス制御 (RBAC) 承認システムを提供します。 Cognitive Search では、次の目的で Azure ロールを使用できます。

- コントロール プレーン操作 (Azure Resource Managerを介したサービス管理タスク)。
- インデックスの作成、読み込み、クエリなどのデータ プレーン操作。

**リファレンス:** [Azure Cognitive Searchで Azure ロールベースのアクセス制御 \(Azure RBAC\) を使用する](#)

## PA-8: クラウド プロバイダー サポートのアクセス プロセスを決定する

### 機能

#### カスタマー ロックボックス

**説明:** カスタマー ロックボックスは、Microsoft サポートへのアクセスに使用できます。 詳細については、[こちらを参照してください。](#)

[+] [テーブルを展開する](#)

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** Microsoft がデータにアクセスする必要があるサポート シナリオでは、カスタマー ロックボックスを使用して確認し、Microsoft の各データ アクセス要求を承認または拒否します。

# データの保護

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: データ保護](#)」を参照してください。

## DP-1: 機密データを検出、分類、ラベル付けする

### 機能

#### 機密データの検出と分類

**説明:** ツール (Azure Purview や Azure Information Protection など) は、サービスでのデータの検出と分類に使用できます。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## DP-2: 機密データをターゲットにした異常と脅威を監視する

### 機能

#### データ漏えい/損失防止

**説明:** サービスでは、機密データの移動 (顧客のコンテンツ内) を監視するための DLP ソリューションがサポートされています。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## DP-3: 転送中の機密データの暗号化

### 機能

#### 転送中データの暗号化

**説明:** サービスでは、データプレーンの転送中のデータ暗号化がサポートされています。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	True	Microsoft

**構成ガイダンス:** 既定のデプロイでこれが有効になっているので、追加の構成は必要ありません。

**リファレンス:** [転送中の暗号化中のデータを Azure Cognitive Search する](#)

## DP-4: 保存データ暗号化を既定で有効にする

### 機能

#### プラットフォーム キーを使用した保存データの暗号化

**説明:** プラットフォーム キーを使用した保存データの暗号化がサポートされています。 保存中の顧客コンテンツは、これらの Microsoft マネージド キーで暗号化されます。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	True	Microsoft

**構成ガイダンス:** 既定のデプロイでこれが有効になっているので、追加の構成は必要ありません。

**リファレンス:** [サービスマネージド キーを使用した既定のデータ暗号化の Azure Cognitive Search](#)

## DP-5: 必要に応じて保存データ暗号化でカスタマー マネージド キー オプションを使用する

### 機能

#### CMK を使用した保存データの暗号化

**説明:** カスタマー マネージド キーを使用した保存データの暗号化は、サービスによって格納される顧客コンテンツでサポートされています。 詳細については、[こちらを参照してください。](#)

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** 規制コンプライアンスに必要な場合は、カスタマー マネージド キーを使用した暗号化が必要なユースケースとサービス スコープを定義します。 それらのサービスでカスタマー マネージド キーを使って、保存データ暗号化を有効にして実装します。

**リファレンス:** [Azure Cognitive Searchでデータ暗号化用にカスタマー マネージド キーを構成する](#)

## DP-6: セキュア キー管理プロセスの使用

### 機能

#### Azure Key Vault でのキー管理

**説明:** このサービスでは、カスタマー キー、シークレット、または証明書に対する Azure Key Vault 統合がサポートされています。 詳細については、[こちらを参照してください。](#)

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** Azure Key Vaultを使用して、キーの生成、配布、ストレージなど、暗号化キーのライフサイクルを作成および制御します。定義されたスケジュールに基づいて、またはキーの廃止や侵害が発生した場合に、Azure Key Vaultとサービスのキーをローテーションして取り消します。ワークロード、サービス、またはアプリケーションレベルでカスタマー マネージド キー (CMK) を使用する必要がある場合は、キー管理のベストプラクティスに従ってください。キー階層を使用して、キー コンテナーにキー暗号化キー (KEK) を使用して別のデータ暗号化キー (DEK) を生成します。キーが Azure Key Vaultに登録され、サービスまたはアプリケーションのキー ID を介して参照されていることを確認します。独自のキー (BYOK) をサービスに持ち込む必要がある場合 (オンプレミスの HSM から Azure Key Vault に HSM で保護されたキーをインポートする場合など)、初期キーの生成とキー転送を実行するための推奨ガイドラインに従ってください。

**リファレンス:** [Azure Cognitive Search でデータ暗号化用にカスタマー マネージド キーを構成する](#)

## DP-7: セキュリティで保護された証明書管理プロセスを使用する

### 機能

#### Azure Key Vault での証明書管理

**説明:** このサービスでは、顧客証明書に対する Azure Key Vault 統合がサポートされます。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

### アセット管理

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: 資産管理](#)」を参照してください。

## AM-2: 承認済みのサービスのみを使用する

### 機能

#### Azure Policy のサポート

**説明:** サービス構成は、Azure Policy経由で監視および適用できます。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** Microsoft Defender for Cloud を使用して、Azure リソースの構成を監査および適用するAzure Policyを構成します。 Azure Monitor を使用し、リソースで構成の逸脱が検出されたときにアラートを作成します。 [deny] と [deploy if not exists] 効果Azure Policy使用して、Azure リソース全体でセキュリティで保護された構成を適用します。

**リファレンス:** [Azure Cognitive Search ポリシー](#)

## ログと脅威検出

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: ログ記録と脅威検出](#)」を参照してください。

## LT-1: 脅威検出機能を有効にする

### 機能

#### サービス/製品のオファリングのための Microsoft Defender

**説明:** サービスには、セキュリティの問題を監視およびアラートするためのオファリング固有のMicrosoft Defender ソリューションがあります。 詳細については、[こちらを参照してください](#)。

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## LT-4: セキュリティ調査のためのログを有効にする

### 特徴

#### Azure リソース ログ

**説明:** サービスは、サービス固有のメトリックとログ記録を強化できるリソース ログを生成します。お客様はこれらのリソース ログを構成し、ストレージ アカウントやログ分析ワークスペースなどの独自のデータ シンクに送信できます。 詳細については、[こちらを参照してください。](#)

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
True	False	Customer

**構成ガイダンス:** サービスのリソース ログを有効にして、Azure Cognitive Search操作ログ、検索メトリックなどを表示します。

**リファレンス:** [リソース ログ](#) [Azure Cognitive Search](#)

## バックアップと回復

詳細については、「[Microsoft クラウド セキュリティ ベンチマーク: バックアップと回復](#)」を参照してください。

## BR-1:定期的な自動バックアップを保証する

### 機能

## Azure Backup

**説明:** サービスは、Azure Backup サービスによってバックアップできます。 詳細については、[こちらを参照してください。](#)

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## サービス ネイティブ バックアップ機能

**説明:** サービスでは、独自のネイティブ バックアップ機能がサポートされます (Azure Backupを使用していない場合)。 詳細については、[こちらを参照してください。](#)

[+] テーブルを展開する

サポートされています	既定で有効	構成の責任
False	適用しない	適用しない

**機能に関するメモ:** Azure Cognitive Searchはプライマリ データストレージ ソリューションではないので、Microsoft はセルフサービスのバックアップと復元のための正式なメカニズムを提供していません。 ただし、独自のコードを使用してインデックスをバックアップおよび復元できます。 「[バックアップと復元の代替手段](#)」を参照してください。

**構成ガイダンス:** この機能は、このサービスをセキュリティで保護するためにサポートされていません。

## 次のステップ

- Microsoft クラウド セキュリティ ベンチマークの概要を参照してください
- Azure セキュリティ ベースラインの詳細について学習する