VectorStore

Description

Initializes a new Vector Store Client instance. This function acts as the factory for connecting to various vector store providers (such as InMemory, Milvus, Pinecone, Chroma, and Qdrant). It handles connection configuration, authentication, and default collection settings.

Returns

A VectorStoreClient object used to perform data and collection operations.

Category

Vector store functions

Syntax

VectorStore (struct configuration)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
configuration	Struct	Yes	A structure containing connection and collection configuration details.

Configuration struct details

The configuration structure supports the following keys. Applicability may vary by provider (Milvus, Pinecone, Qdrant, Chroma) .

Key	Type	Default	Description	Supported Providers
provider	String	-	The vector store provider. Values: `milvus`, `pinecone`, `qdrant`, `chroma` (or `inmemory` for testing).	All
url	String	-	The endpoint address for the vector store API.	Milvus, Qdrant, Chroma
apiKey	String	-	The authentication	Milvus, Pinecone,

			key for the service.	Qdrant, Chroma
databaseName	String	-	The specific database instance name.	Milvus, Pinecone
collectionName	String	-	The name of the collection to connect to or create.	All
dimension	Numeric	-	The dimensionality of the vectors (e.g., 128, 1536).	Milvus, Pinecone
metricType	String	`COSINE`	The distance metric used for similarity. Values: `COSINE`, `EUCLIDEAN`, `DOTPRODUCT`.	All
indexType	String	`IVF_FLAT`	The indexing algorithm (e.g., `IVF`, `HNSW`, `IVF_FLAT`).	Milvus, Pinecone, Qdrant
embeddingModel	Struct	-	The alias of a configured embedding model. If set, raw vectors	All

			are not required in `add` or `search` operations.	
topK	Numeric	4	Default number of results to return in a search.	All
minScore	Numeric	0	Default minimum similarity score threshold.	All
callTimeout	Numeric	-	Max time (ms) to wait for server response.	All
connectTimeout	Numeric	-	Max time (ms) to wait for connection establishment.	All
maxRetries	Numeric	3	Maximum retry attempts for failed operations.	All
retryOnRateLimit	Boolean		Determines whether to retry when encountering rate limit	Milvus

			errors (HTTP 429)	
initialBackof	Numeric		The first delay interval between retry attempts.	All
maxBackOff	Numeric		Maximum delay interval between retries (caps exponential growth).	Except Chroma
backOffMultiplier	Numeric		Factor by which the delay interval increases after each retry	Except Chroma
retryTimeout	Numeric		The maximum time allowed for retry operations to complete.	Only Milvus
callTimeout	Numeric		Maximum time to wait for responses from the server.	All

connectionTimeout	Numeric		Maximum time to wait for initial connection establishment.	All
keepAlive	Boolean		Whether to send keep-alive pings even when no RPC calls are active.	Except Chroma
keepAliveTime	Numeric		Interval between keep-alive pings to maintain connection.	Except Chroma
keepAliveTimeout	Numeric		Time to wait for keep-alive ping response before considering connection dead.	Except Chroma
idleTimeout	Numeric		Time after which idle connections are closed.	All

maxConnections	Numeric		The maximum number of connections allowed in the connection pool.	Only Chroma
socketTimeout	Numeric		The maximum time to wait for data transfer between the client and server after a connection is established.	Only Chroma
connectionRequestTimeout	Numeric		The maximum time to wait for a connection from the pool when all connections are in use.	Only Chroma

Usage

Use this function to establish the connection before performing any CRUD or search operations. If an embeddingModel is provided in the configuration, the client will automatically handle text-to-vector conversion.

Example

```
<cfscript>
    // Shared configuration for all VectorStore examples
```

```
    vectorStoreConfig = {
        provider       : "milvus",
        url            : "http://localhost:19530",
        apiKey         : "YOUR_API_KEY",
        collectionName : "knowledge_base",
        dimension      : 1536,
        metricType     : "COSINE",
        indexType      : "IVF_FLAT",
        "embeddingModel": {
             "provider": "ollama",
             "modelName": "all-minilm:latest",
             "baseUrl": "http://localhost:11434",
              "maxRetries": 3
        }

        topK           : 4,
        minScore       : 0
    };

    try {
        vectorStoreClient = VectorStore(vectorStoreConfig);
        writeOutput("VectorStore client initialized successfully.<br>");
    } catch (any e) {
        writeOutput("Error initializing VectorStore client: " & e.message &
"<br>");
    }
</cfscript>
```

add

Description

Adds a single item (document) to the vector store. If an embedding model is configured in the client, the vector parameter is optional, and the text will be automatically embedded.

Returns

A string containing the ID of the added item.

Category

Vector store functions

Syntax

vectorStoreClient.add(item)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
item	Struct	Yes	The data structure representing the item to store.

Item struct details

Key	Type	Required	Description
id	String	No	Unique identifier. If omitted, one may be auto-generated by the provider or CF.
text	String	Yes	The raw text content of the document.
vector	Array	No	Array of floats representing the embedding. ***Required only if no `embeddingModel` is configured on the client.***

metadata	Struct	No	Key-value pairs for filtering (e.g., category, date).

Example

```
<cfscript>
    config = {
        provider: "milvus",
        url: "https://192.168.1.100:19530",
        apiKey: "YOUR_API_KEY",
        collectionName: "knowledge_base",
        dimension: 1536,
        metricType: "COSINE",
        embeddingModel: "text-embedding-3-small"
    };

    try {
        vsClient = VectorStore(config);
        writeOutput("Client initialized successfully.");
    } catch (any e) {
        writeOutput("Error: " & e.message);
    }

    newItem = {
        id: createUUID(),
        text: "ColdFusion is a powerful rapid application development platform",
        metadata: {
            category: "technology",
            author: "Adobe"
        }
    };

    docId = vsClient.add(newItem);
</cfscript>
```

addAll

Description

Adds multiple items to the vector store in a batch. This function is optimized to handle bulk insertions effectively, utilizing batching mechanisms to manage memory and network limits (default batch size is typically 1000).

Returns

An array of strings containing the IDs of the added items.

Category

Vector store functions

Syntax

vectorStoreClient.addAll(items)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
items	Array	Yes	An array of item structures (see **add** for struct definition).

Example

```
<cfscript>
    config = {
        provider: "milvus",
        url: "https://192.168.1.100:19530",
        apiKey: "YOUR_API_KEY",
        collectionName: "knowledge_base",
        dimension: 1536,
        metricType: "COSINE",
        embeddingModel: "text-embedding-3-small"
    };
```

```
    try {
        vsClient = VectorStore(config);
        writeOutput("Client initialized successfully.");
    } catch (any e) {
        writeOutput("Error: " & e.message);
    }

    docs = [
        { text: "Vector DBs are fast", metadata: { type: "db" } },
        { text: "AI requires context", metadata: { type: "ai" } }
    ];

    ids = vsClient.addAll(docs);
</cfscript>
```

search

Description

Performs a similarity search (for example, Nearest Neighbor search) on the vector store. You can search using raw text (if an embedding model is configured) or a vector array. Results can be refined using metadata filters and score thresholds.

Returns

An array of structs, where each struct contains the item's id, text, vector, metadata, and a similarity score.

Category

Vector store functions

Syntax

vectorStoreClient.search(query)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description

query	Struct	Yes	The search query configuration.

Query struct details

Key	Type	Description
text	String	The text to search for. (Required if `vector` is not provided).
vector	Array	The query embedding vector. (Required if `embeddingModel` is not configured).
topK	Numeric	The number of approximate nearest neighbors to return (Overrides client default).
minScore	Numeric	The minimum similarity score (0.0 to 1.0) required for a result to be included.
filter	Struct	A metadata filter expression to narrow the search scope.

Filter operators

The filter struct supports the following operators:

- **Comparison**: equals, notEquals, isGreaterThan, isGreaterThanOrEqual, isLessThan, isLessThanOrEqual
- **Set**: isIn (array), notIn (array)
- **Logical**: and, or, not

Example

```
<cfscript>
    config = {
        provider: "milvus",
        url: "https://192.168.1.100:19530",
        apiKey: "YOUR_API_KEY",
        collectionName: "knowledge_base",
```

```
        dimension: 1536,
        metricType: "COSINE",
        embeddingModel: "text-embedding-3-small"
    };

    try {
        vsClient = VectorStore(config);
        writeOutput("Client initialized successfully.");
    } catch (any e) {
        writeOutput("Error: " & e.message);
    }

    searchQuery = {
        text: "How do I implement RAG?",
        topK: 5,
        minScore: 0.75,
        filter: {
            category: "documentation",
            year: { isGreaterThanOrEqual: 2025 }
        }
    };

    results = vsClient.search(searchQuery);
</cfscript>
```

delete

Description

Deletes a single item from the vector store identified by its ID.

Returns

Void

Category

Vector store functions

Syntax

vectorStoreClient.delete(id)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
id	String	Yes	The unique identifier of the item to delete.

Example

```
<cfscript>
    config = {
        provider: "milvus",
        url: "https://192.168.1.100:19530",
        apiKey: "YOUR_API_KEY",
        collectionName: "knowledge_base",
        dimension: 1536,
        metricType: "COSINE",
        embeddingModel: "text-embedding-3-small"
    };

    try {
        vsClient = VectorStore(config);
        writeOutput("Client initialized successfully.");
    } catch (any e) {
        writeOutput("Error: " & e.message);
    }

    newItem = {
        id: createUUID(),
        text: "ColdFusion is a powerful rapid application development platform",
        metadata: {
            category: "technology",
            author: "Adobe"
        }
    };

    docs = [
        { id: "DOC-123",text: "Vector DBs are fast", metadata: { type: "db" } },
```

```
        { id: "DOC-456", text: "AI requires context", metadata: { type: "ai" } }
    ];

    ids = vsClient.addAll(docs);

    vsClient.delete("DOC-123")

</cfscript>
```

deleteAll

Description

Deletes items from the vector store. It can be used in three ways: deleting specific IDs, deleting items matching a metadata filter, or deleting all items in the collection.

Returns

Void

Category

Vector store functions

Syntax

// 1. Delete by IDs

vectorStoreClient.deleteAll(ids)

// 2. Delete by Filter

vectorStoreClient.deleteAll(filter)

// 3. Delete All

vectorStoreClient.deleteAll()

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description

ids	Array	No	A list of specific IDs to delete.
filter	Struct	No	A metadata filter struct (see `search` for operators). Deletes all items matching the criteria.
(None)	-	No	If no argument is provided, all items in the store are deleted.

Example

```
<cfscript>
    config = {
        provider: "milvus",
        url: "https://192.168.1.100:19530",
        apiKey: "YOUR_API_KEY",
        collectionName: "knowledge_base",
        dimension: 1536,
        metricType: "COSINE",
        embeddingModel: "text-embedding-3-small"
    };

    try {
        vsClient = VectorStore(config);
        writeOutput("Client initialized successfully.");
    } catch (any e) {
        writeOutput("Error: " & e.message);
    }

    newItem = {
        id: createUUID(),
        text: "ColdFusion is a powerful rapid application development platform",
        metadata: {
            category: "technology",
            author: "Adobe"
```

```
        }
    };

    docs = [
        { id: "DOC-123",text: "Vector DBs are fast", metadata: { type: "db" },
status: "active" },
        { id: "DOC-456", text: "AI requires context", metadata: { type:
"ai" },status: "active" },
        { id: "DOC-789", text: "ColdFusion makes development easy", metadata:
{ type: "dev" }, status: "inactive" },
        { id: "DOC-007", text: "ColdFusion doesn't use COM and DCOM", metadata:
{ type: "dev" }, status: "inactive" }
    ];

    vsClient.addAll(docs);

    // Delete by filter
    vsClient.deleteAll({
        status: "deprecated"
    });
</cfscript>
```

listCollections

Description

Retrieves a list of all collection names available in the connected vector store.

Returns

An array of strings (collection names).

Category

Vector store functions

Syntax

vectorStoreClient.listCollections()

History

New in ColdFusion 2025.1

Parameters

None

Example

```
collections = vectorStoreClient.listCollections();
// Returns: ["colors", "languages"]
```

deleteCollection

Description

Permanently deletes a specific collection and all vector data contained within it.

Returns

Void

Category

Vector store functions

Syntax

deleteCollection(collectionName)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
collectionName	String	Yes	The name of the collection to delete.

Example

vectorStoreClient.deleteCollection("colors");

getEmbeddingModel

Description

Initializes a new Embedding Model client instance. This function acts as the factory for connecting to various embedding providers (such as OpenAI, Azure OpenAI, local models, etc.). It encapsulates model configuration (model name, dimension, provider) and connection settings.

Returns

An EmbeddingModel object used to generate embeddings for text input.

Category

Embedding model functions

Syntax

getEmbeddingModel(struct configuration)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
configuration	Struct	Yes	Structure containing provider and model configuration.

Configuration struct details

The configuration structure supports the following common keys. Exact support varies by provider (OpenAI, AzureOpenAI, local, etc.).

Key	Type	Default	Description
provider	String	-	Embedding provider. Examples: `"openai"`, `"azureOpenAI"`, `"local"`.
apiKey	String	-	API key or token for the provider (if required).

baseUrl	String	-	Optional base endpoint URL (for self-hosted or Azure endpoints).
modelName	String	-	Embedding model identifier. Example: `"text-embedding-3-small"`.
dimension	Numeric	-	Expected embedding dimension (for validation).
timeout	Numeric	10000	Request timeout in milliseconds.
maxRetries	Numeric	2	Maximum retry attempts for transient failures.
logRequests	Boolean	true	Whether to log outbound requests (for debugging).
logResponses	Boolean	true	Whether to log responses (for debugging).
metadata	Struct	{}	Optional key-value metadata to tag embedding calls.

Usage

Use GetEmbeddingModel once during initialization. The returned embeddingModel instance exposes:

- embed(text) – generate an embedding for a single text string.
- embedAll(textArray) – generate embeddings for an array of text strings.

The model configuration (provider, model name, dimension) is fixed per instance.

Example

```
<cfscript>
    // Create an Embedding Model client for OpenAI
    embedConfig = {
```

```
        provider  : "openai",
        apiKey    : "YOUR_OPENAI_API_KEY",
        baseUrl   : "https://api.openai.com/v1",      // optional for OpenAI-
style endpoints
        modelName : "text-embedding-3-small",
        dimension : 1536
    };

    try {
        embeddingModel = GetEmbeddingModel(embedConfig);
        writeOutput("Embedding model initialized successfully.<br>");
    } catch (any e) {
        writeOutput("Error initializing embedding model: " & e.message & "<br>");
    }
</cfscript>
```

EmbeddingModel.embed

Description

Generates an embedding vector for a single text input using the configured embedding model. This is typically used for:

- indexing documents into a VectorStore, or
- creating a query vector for similarity search.

Returns

A struct containing:

- embeddings – the embedding vector (array of numbers)
- tokenUsage – provider-specific token usage metrics (if available)
- finishReason – reason the call completed (if provided by the provider)
- metadata – additional metadata returned by the provider

Category

Embedding model functions

Syntax

embeddingModel.embed(string text)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
text	String	Yes	The input text to embed.

Usage

Use embed() for one-off embedding generation, such as:

- embedding a single user query before calling vectorStoreClient.search({vector: ...}), or
- embedding a newly created document you're about to upsert into a VectorStore.

For batch ingestion of many texts, prefer embedAll() to reduce overhead.

Example

```
<cfscript>
    // Assume embeddingModel was created via GetEmbeddingModel(embedConfig)

    inputText = "ColdFusion integrates AI models and vector databases.";

    result = embeddingModel.embed(inputText);

    // result is a struct with:
    // - result.embeddings   : array of numeric values (the vector)
    // - result.tokenUsage   : provider-specific token stats (if supported)
    // - result.finishReason : string (if provided)
    // - result.metadata     : struct with extra information (if any)

    writeOutput("<h4>Single embedding result</h4>");
    writeDump(result);

    // Use the embedding vector directly with a VectorStore search, e.g.:
    //
    // queryVector = result.embeddings;
    // searchResult = vectorStoreClient.search({ vector: queryVector, topK: 5 });
</cfscript>
```

embeddingModel.embedAll

Description

Generates embeddings for multiple text inputs in a single call. This method is optimized for throughput and is the recommended way to embed:

- document batches during ingestion, or
- large sets of knowledge-base entries.

Returns

A struct containing:

- embeddings – an array of vectors (one per input string, in the same order).
- tokenUsage – aggregated token usage metrics (if available).
- finishReason – reason the call completed (if provided by the provider).
- metadata – provider-specific metadata.

Category

Embedding model functions

Syntax

embeddingModel.embedAll(array text)

History

New in ColdFusion 2025.1

Parameters

Parameter	Type	Required	Description
text	Array	Yes	Array of text strings to embed (batch input).

Usage

Use embedAll() whenever you have more than one string to embed. It is more efficient and easier to coordinate with batch insert/upsert operations into a VectorStore.

Example

```
<cfscript>
    // Assume embeddingModel was created via GetEmbeddingModel(embedConfig)
```

```
    // Batch of texts to embed
    texts = [
        "How to reset your account password in ColdFusion.",
        "Using vector databases for semantic search.",
        "Retrieval-Augmented Generation (RAG) with ColdFusion."
    ];

    batchResult = embeddingModel.embedAll(texts);

    // batchResult.embeddings is an array of vectors with the same length as
`texts`.
    writeOutput("<h4>Batch embedding result</h4>");
    writeDump(batchResult);

    // Example: prepare documents for a VectorStore addAll()
    docs = [];
    for (i = 1; i <= arrayLen(texts); i++) {
        doc = {
            id       : "DOC-" & numberFormat(i, "000"),
            text     : texts[i],
            vector   : batchResult.embeddings[i],
            metadata : {
                source   : "kb",
                language : "en",
                index    : i
            }
        };
        arrayAppend(docs, doc);
    }

    // Now docs can be passed to vectorStoreClient.addAll(docs);
</cfscript>
```