317 SODA Types

There are several SODA types: SODA_DOCUMENT_T, SODA_COLLECTION_T, SODA_OPERATION_T, and SODA_CURSOR_T. SODA_DOCUMENT_T and SODA_COLLECTION_T represent two primary abstractions provided by SODA: document and collections. SODA_OPERATION_T is used for specifying condition of operations on the collection. SODA_CURSOR_T is a cursor over results of read operations on the collection.

This chapter contains the following topics:

SODA Types Overview

There are several SODA types: SODA_DOCUMENT_T, SODA_COLLECTION_T, SODA_OPERATION_T, and SODA_CURSOR_T. SODA_DOCUMENT_T and SODA_COLLECTION_T represent two primary abstractions provided by SODA: document and collections. SODA_OPERATION_T is used for specifying condition of operations on the collection. SODA_CURSOR_T is a cursor over results of read operations on the collection.

See Also:

- *Oracle Database SODA for PL/SQL Developer's Guide*
- DBMS_SODA

SODA Types Security Model

The SODA Types are available to users with the SODA_APP role.

All SODA types are SYS types. PUBLIC is granted EXECUTE privilege on the SODA types described in this chapter:

- TYPE SODA_Collection_T
- TYPE SODA_Document_T
- TYPE SODA_Operation_T
- TYPE SODA_Cursor_T

Summary of SODA Types

This chapter lists the SODA types and describes them.

Table 317-1 SODA Types

Type	Description
SODA_Collection_T Type	This SODA type represents a SODA collection. This type is not persistable.
SODA_Document_T Type	This SODA type represents a document with content, usually in JSON format. This type is not persistable.
SODA_OPERATION_T Type	This SODA type performs read/write operations, such as document finds with filtering and pagination, removes, and replaces on a SODA collection. This type is not persistable.
SODA_CURSOR_T Type	This SODA type represents the result set of documents. This type is not persistable.

SODA_Collection_T Type

This SODA type represents a SODA collection. A reference of SODA collection can only be obtained by either calling DBMS_SODA.CREATE_COLLECTION() or DBMS_SODA.OPEN_COLLECTION().

Table 317-2 SODA_Collection_T Type Subprograms

Subprogram	Description
CREATE_INDEX Function	Creates an index using an index specification expressed in JSON. Three types of specifications are supported. Each specifying a different type of index: for B-tree, JSON search with Data Guide, and Spatial.
CREATE_VIEW_FROM_DG Function	Creates a view with relational columns, using scalar JSON fields as specified in the data guide.
DROP_INDEX Function	Drops the named index.
FIND Function	Returns the SODA_OPERATION_T object. This is the only way to get the reference of SODA_Operation_T as there is no constructor.
FIND_ONE Function	Fetches the document matching the key.
GET_DATA_GUIDE Function	Returns the JSON data guide as a CLOB.
GET_INDEX Function	This function returns the specification for the supplied index created on the collection.
GET_METADATA Function	Returns the metadata of the collection in JSON format.
GET_NAME Function	Returns the name of the collection.
INSERT_ONE Function	Inserts a document into the collection.

Table 317-2 (Cont.) SODA_Collection_T Type Subprograms

Subprogram	Description
INSERT_ONE_AND_GET Function	Inserts a document into the collection and returns a result document with all components except for content.
LIST_INDEXES Function	This function returns the specifications for all the indexes created on the collection.
REMOVE_ONE Function	Removes the document matching the key.
REPLACE_ONE Function	Replaces the content and (optionally) the media type of the document matching the key.
REPLACE_ONE_AND_GET Function	Replaces the content and (optionally) the media type of the document matching the key and returns a result document with all components (except content).
SAVE Function	Saves a document into the collection.
SAVE_AND_GET Function	Saves a document into the collection.
TRUNCATE Function	Deletes all documents in the collection.

CREATE_INDEX Function

This function creates an index using an index specification expressed in JSON. Three types of specifications are supported. Each specifying a different type of index: for B-tree, JSON search with Data Guide, and Spatial.

Syntax

```
CREATE_INDEX (
     specification VARCHAR2)
  RETURN NUMBER;
```

Parameters

Table 317-3 CREATE_INDEX Function Parameters

Parameter	Description
`specification`	The index specification.

Example 317-1 Return Values

The function returns:

- `1`—if the index was successfully created
- `0`—if the index was not created

Exceptions

`Error`—If an error occurs creating the index.

See Also:

For more information about SODA Index specifications, see:

- Overview of SODA Indexing
- SODA Index Specifications (Reference)

CREATE_VIEW_FROM_DG Function

This function creates a view with relational columns, using scalar JSON fields as specified in the data guide. A data guide enabled JSON search index is not required for this function; the data guide is passed to the function. An error is thrown if the data guide passed to the function is invalid.

This procedure is available only for Autonomous Database starting 19c release.

Syntax

```
CREATE_VIEW_FROM_DG (
   data_guide             IN   CLOB,
   view_name              IN   VARCHAR2,
   materialize            IN   BOOLEAN DEFAULT FALSE,
   mv_refresh_mode        IN   NUMBER DEFAULT 1,
   path                   IN   VARCHAR2 DEFAULT '$',
   resolve_name_conflicts IN   BOOLEAN   DEFAULT FALSE,
   col_name_prefix        IN   VARCHAR2 DEFAULT NULL,
   mixed_case_columns     IN   BOOLEAN DEFAULT FALSE)
  RETURN NUMBER;
```

Parameters

Table 317-4   CREATE_VIEW_FROM_DG Function Parameters

Parameter	Description
data_guide	Data guide of the collection.
view_name	Name of the view to be created.
materialize	A boolean value to indicate if the view should be materialized or not. The default value is FALSE.
mv_refresh_mode	The materialized view refresh mode. Possible values are: • DBMS_SODA.MV_REFRESH_ON_STATEMENT (default) • DBMS_SODA.MV_REFRESH_ON_COMMIT • DBMS_SODA.MV_REFRESH_ON_DEMAND

Table 317-4 (Cont.) CREATE_VIEW_FROM_DG Function Parameters

Parameter	Description
`path`	The path of the JSON field to be expanded. It uses JSON path expression syntax. For example: • `$` will create a view starting from the JSON document root • `$.purchaseOrder` will create a view starting from purchaseOrder. It expands the children or descendants under `purchaseOrder`, and create view columns for every scalar value. The default value is `$`.
`resolve_name_conflicts`	By default, if there are conflicts among `o:preferred_column_name`, an error is raised. If you set this parameter to `TRUE`, the procedure automatically resolves the view column name conflicts by appending a sequence number. The default value is `FALSE`.
`col_name_prefix`	By default, the view column name is the same as the JSON field name. This parameter allows you to provide a prefix to prepend to the view column names. The default value is `NULL`.
`mixed_case_columns`	By default, the view column names are case sensitive. This parameter allows you to change the behavior to case insensitive. The default value is `FALSE`.

Return Values

The function returns:

- `1`—if the procedure is successfully completed
- `0`—if the procedure could not be successfully completed

Exceptions

`Error`—If an error occurs if the function was unable to create a view.

See Also:

For more info on the JSON data guide, see JSON Data Guide

DROP_INDEX Function

This function drops the named index.

Syntax

```
DROP_INDEX (
     index_Name IN VARCHAR2,
     force       IN BOOLEAN DEFAULT FALSE)
  RETURN NUMBER;
```

Example 317-2 Parameters

Table 317-5 DROP_INDEX Function Parameters

Parameter	Description
`index_Name`	The name of the index.
`force`	The force parameter can be `TRUE` or `FALSE`. Should only be set to `TRUE` for dropping a `JSON` search index or spatial index (not B-tree index). For more information, see DROP INDEX

Return Values

The function returns:

- `1`—If the index was successfully dropped
- `0`—If the index was not dropped. For example, if there was no existing index with the specified name.

Exceptions

`Error`—if an error occurs while dropping the index.

FIND Function

This function returns the operation type for the collection. The operation type allows building and executing various read/write operations. This is the only way to get the reference of `SODA_Operation_T` as there is no constructor.

Syntax

```
FIND ()
 RETURN SODA_Operation_T;
```

Return Values

This function returns `SODA_OPERATION_T` object.

Exceptions

This function does not throw any exception.

FIND_ONE Function

This function fetches the document matching the given key.

Syntax

```
FIND_ONE (
     key          IN VARCHAR2)
 RETURN SODA_Document_T;
```

Parameters

Table 317-6   FIND_ONE Function Parameters

Parameter	Description
`key`	The key of the document to be fetched.

Return Values

This function returns the document that matches the key. Returns `NULL` if no match is found.

Exceptions

`Error`—If an error occurs while finding the document.

GET_DATA_GUIDE Function

This function fetches the JSON data guide as a CLOB. The JSON data guide is essentially inferred schema for the JSON documents in the collection. In order to be able to return the JSON data guide, a collection must have a JSON Search Index defined on it, with the data guide enabled.

Syntax

```
GET_DATA_GUIDE ()
 RETURN CLOB;
```

Return Values

The function returns the JSON data guide as a CLOB.

See Also:

For more info on the JSON data guide, see JSON Data Guide

Exceptions

`Error`—If an error occurs while fetching the data guide.

GET_INDEX Function

This function returns the specification for the supplied index created on the collection.

Syntax

```
GET_INDEX (
     index_name           IN VARCHAR2,
     schema_name          IN VARCHAR2 DEFAULT NULL)
  RETURN VARCHAR2;
```

Parameters

Table 317-7 GET_INDEX Function Parameters

Parameter	Description
`index_name`	The name of the index to be described.
`schema_name`	Name of the schema containing the index. This parameter is optional. If this parameter is not set, the method will look for the specified index in the schema from which this method is called.

Return Values

The function returns the index specification in `JSON` format.

Exceptions

`Error`—If an error occurs while returning the index specification.

GET_METADATA Function

This function returns the metadata of the collection in `JSON` format.

Syntax

```
GET_METADATA ()
  RETURN VARCHAR2;
```

Return Values

This function returns the metadata of the collection in `JSON` format.

GET_NAME Function

This function returns the name of the collection.

Syntax

```
GET_NAME ()
 RETURN NVARCHAR2;
```

Return Values

This function returns the name of the collection.

INSERT_ONE Function

This function inserts a document into the collection.

Syntax

```
INSERT_ONE (
     document       IN SODA_Document_T)
 RETURN NUMBER;
```

Parameters

Table 317-8 INSERT_ONE Function Parameters

Parameter	Description
`document`	The input document.

Return Values

The function returns a number– `1` if the doc was inserted successfully, `0` otherwise.

Exceptions

`Error`—If an error occurs while inserting the document into the collection.

INSERT_ONE_AND_GET Function

This function inserts a document into the collection.

Syntax

```
INSERT_ONE_AND_GET (
     document      IN SODA_Document_T,
     hint          IN VARCHAR2 DEFAULT NULL)
 RETURN SODA_Document_T;
```

Parameters

Table 317-9 INSERT_ONE_AND_GET Function Parameters

Parameter	Description
`document`	The input document.
`hint`	A hint string in Oracle SQL format, without the enclosing `/*+` and `*/`. This parameter is optional.

Return Values

The function returns the result document containing all document components supported by the given collection, with the exception of content.

Exceptions

`Error`—If an error occurs while inserting the document into the collection.

LIST_INDEXES Function

This function returns the specifications for all the indexes created on the collection.

Syntax

```
LIST_INDEXES ( )
  RETURN SODA_Index_List_T;
```

Return Values

The function returns index specifications in `JSON` format as an instance of `SODA_Index_List_T`.

Exceptions

`Error`—If an error occurs while returning the index specification.

REMOVE_ONE Function

This function removes the document matching the given key.

Syntax

```
REMOVE_ONE (
     key        IN VARCHAR2)
 RETURN NUMBER;
```

Parameters

Table 317-10 REMOVE_ONE Function Parameters

Parameter	Description
`key`	The key of the document.

Return Values

This function returns the following values:

- `1`–If the document was successfully removed.
- `0`–If the document with the specified key was not found.

Exceptions

`Error`—If an error occurs while deleting the document from the collection.

REPLACE_ONE Function

This function updates the existing document with a new content and media type using the key. Any components set in `document` with the exception of content and media type are not used during the replace. They are ignored.

Syntax

```
REPLACE_ONE (
     key             IN VARCHAR2,
     document        IN SODA_Document_T)
 RETURN NUMBER;
```

Parameters

Table 317-11 REPLACE_ONE Parameters

Parameter	Description
`key`	The key of the document.
`document`	The document with the new content and media type to replace the old one.

Return Values

This function returns a number—`1` if the document was replaced, `0` otherwise.

Exceptions

`Error`—If an error occurs while replacing the document in the collection.

REPLACE_ONE_AND_GET Function

This function updates the existing document with a new content and media type using the key. Any components set in `document` with the exception of content and media type are not used during the replace. They are ignored.

Syntax

```
REPLACE_ONE_AND_GET (
     key             IN VARCHAR2,
     document        IN SODA_Document_T)
 RETURN SODA_Document_T;
```

Parameters

Table 317-12 REPLACE_ONE_AND_GET Function Parameters

Parameter	Description
`key`	The key of the document.
`document`	The document with the new content and media type to replace the old one.

Return Values

The function returns the result document containing all document components supported by the given collection, with the exception of content. Last-modified and version components, if supported by the given collection, will be updated with new values. If no document in the collection had the supplied key, `NULL` is returned instead of the result document.

Exceptions

`Error`—If an error occurs while replacing the document in the collection.

SAVE Function

This function saves a document into the collection. This function is equivalent to the `INSERT_ONE(document)` function except that if client-assigned keys are used, and the document with the specified key already exists in the collection, it will be replaced with the input document. The key is automatically created, unless this collection is configured with client-assigned keys and the key is provided in the input document.

Syntax

```
SAVE (
     document IN  SODA_Document_T)
  RETURN NUMBER;
```

Parameters

Table 317-13 SAVE Parameters

Parameter	Description
`document`	The input document. This cannot be null.

Return Values

The function returns a number- `1` if the function successfully completed, `0` otherwise.

Exceptions

`Error`—If an error occurs while saving the document.

SAVE_AND_GET Function

This function saves a document into the collection. This method is equivalent to `INSERT_ONE_AND_GET(document)` except that if client-assigned keys are used, and the document with the specified key already exists in the collection, it will be replaced with the input document. The key will be automatically created, unless this collection is configured with client-assigned keys and the key is provided in the input document.

Syntax

```
SAVE_AND_GET (
     document      IN SODA_Document_T,
     hint          IN VARCHAR2 DEFAULT NULL)
  RETURN SODA_Document_T;
```

Parameters

Table 317-14 SAVE_AND_GET Function Parameters

Parameter	Description
`document`	The input document. This cannot be null.
`hint`	A hint string in Oracle SQL format, without the enclosing `/*+` and `*/`. This parameter is optional.

Return Values

The function returns the result document containing all document components supported by the given collection, with the exception of content.

Exceptions

`Error`—If an error occurs while saving or getting the document.

TRUNCATE Function

This function deletes all documents in the collection.

Syntax

```
TRUNCATE ( )
  RETURN Number;
```

Return Values

The function returns:

- `1`—if the function is successfully completed
- `0`—if the function could not be successfully completed

Exceptions

`Error`—if an error occurs while deleting the documents in the collection.

SODA_Document_T Type

This `SODA` type represents a document with content, that is usually in `JSON` format.

This type is not persistable `pl/sql` type. However, `SODA` is a system that basically provides persistence — it has read and write operations. So you do not persist `SODA_DOCUMENT_T` directly, but you pass it to a write operation (like `insert` or `replace`), which is defined on `SODA_COLLECTION_T`, in order to write the document content and other components to the database.

A document has the following components:

- key
- content
- created-on timestamp
- last-modified timestamp
- version
- media type

Table 317-15 SODA_Document_T Type Subprograms

Subprogram	Description
GET_BLOB Function	Fetches the BLOB content of a `BLOB`-based document.

Table 317-15 (Cont.) SODA_Document_T Type Subprograms

Subprogram	Description
GET_CLOB Function	Fetches the CLOB content of a CLOB-based document.
GET_CREATED_ON Function	Fetches the created-on timestamp in VARCHAR2.
GET_DATA_TYPE Function	Fetches the SQL datatype of the document content with which it was created.
GET_JSON Function	Fetches the JSON content of a JSON-based document.
GET_KEY Function	Fetches the document key in VARCHAR2.
GET_LAST_MODIFIED Function	Fetches the last modified timestamp in VARCHAR2.
GET_MEDIA_TYPE Function	Fetches the media type of the document content in VARCHAR2.
GET_VARCHAR2 Function	Fetches the VARCHAR2 content of a VARCHAR2-based document.
GET_VERSION Function	Fetches the version of the document in VARCHAR2.
SODA_Document_T Function	There are three different SODA_DOCUMENT_T constructor functions. Each constructor function instantiates a document object using key, content, and media type.

GET_BLOB Function

This functions fetches the BLOB content of the document. It assumes that the document was constructed with BLOB content, or was returned from a collection with BLOB content. Otherwise, an error is returned.

Syntax

```
GET_BLOB ()
 RETURN BLOB;
```

Return Values

This function returns the BLOB content of a document.

Exceptions

SODA Error: If the document was initially not created with BLOB content.

GET_CLOB Function

The function fetches CLOB content of the document. It assumes that the document was constructed with CLOB content, or was returned from a collection with CLOB content. Otherwise, an error is returned.

Syntax

```
GET_CLOB ()
 RETURN CLOB;
```

Return Values

This function returns the `CLOB` content of a document.

Exceptions

`SODA Error:` If the document was initially not created with `CLOB` content.

GET_CREATED_ON Function

This function fetches the created-on timestamp. The timestamp string is in `ISO-8601` format, in particular this form: `YYYY-MM-DDThh:mm:ss.ssssssZ` format. As indicated by the `Z` at the end, timestamps are returned in UTC (`Z` indicates zero UTC offset).

Syntax

```
GET_CREATED_ON ()
 RETURN VARCHAR2;
```

Return Values

This function returns the created-on timestamp.

GET_DATA_TYPE Function

This function fetches the SQL datatype of the document content with which it was created.

Syntax

```
GET_DATA_TYPE ()
 RETURN PLS_INTEGER;
```

Return Values

Table 317-16 GET_DATA_TYPE Return Values

Constant	Value	Description
`DOC_VARCHAR2 CONSTANT PLS_INTEGER`	`1`	`VARCHAR2`
`DOC_BLOB CONSTANT PLS_INTEGER`	`2`	`BLOB`
`DOC_CLOB CONSTANT PLS_INTEGER`	`3`	`CLOB`
`DOC_JSON CONSTANT PLS_INTEGER`	`4`	`JSON`

GET_JSON Function

This functions fetches the `JSON` content of the document. It assumes that the document was constructed with `JSON` type content, or was returned from a collection with `JSON` type content. Otherwise, an error is returned.

Syntax

```
GET_JSON ()
 RETURN JSON;
```

Return Values

This function returns the `JSON` content of a document.

Exceptions

`SODA Error`: If the document was initially not created with `JSON` content.

GET_KEY Function

This function fetches the document key.

Syntax

```
GET_KEY ()
 RETURN VARCHAR2;
```

Return Values

This function returns the document key.

GET_LAST_MODIFIED Function

This function fetches the last modified timestamp. The timestamp string is in `ISO-8601` format, in particular this form: `YYYY-MM-DDThh:mm:ss.ssssssZ` format. As indicated by the `Z` at the end, timestamps are returned in UTC (`Z` indicates zero UTC offset).

Syntax

```
GET_LAST_MODIFIED ()
 RETURN VARCHAR2;
```

Return Values

This function returns the last modified timestamp.

GET_MEDIA_TYPE Function

This function fetches the media type of the document content.

Syntax

```
GET_MEDIA_TYPE ()
 RETURN VARCHAR2;
```

Return Values

This function returns the media type of the document content. `application/JSON` is the media type for `JSON` documents (default).

GET_VARCHAR2 Function

This function fetches the VARCHAR2 content of the document. It assumes that the document was constructed with VARCHAR2 content, or was returned from a collection with VARCHAR2 content. Otherwise, an error is returned.

Syntax

```
GET_VARCHAR2 ()
 RETURN VARCHAR2;
```

Return Values

This function returns the VARCHAR2 content of a document.

Exceptions

SODA Error: If the document was initially not created with VARCHAR2 content.

GET_VERSION Function

This function fetches the version of the document.

Syntax

```
GET_VERSION ()
 RETURN VARCHAR2;
```

Return Values

This function returns the version of the document.

SODA_Document_T Function

This function instantiates a document object using key, content, and media type. There are three different SODA_DOCUMENT_T constructor functions. The second parameter (<v|b|c>_Content) is different in each constructor. It is VARCHAR2 in the first variant, BLOB in the second, and CLOB in the third.

Key and media type are optional parameters (will be defaulted to NULL). All three parameters can be set to NULL. If media_Type is set to NULL, it will be defaulted to application/json.

Use key and j_Content with the constructor to instantiate a document object using key and content. Media type parameter is not present in this constructor as the data is of JSON type. Therefore, media type is understood to be application/json.

Syntax

```
SODA_DOCUMENT_T (
     key            IN VARCHAR2 DEFAULT NULL,
     v_Content      IN VARCHAR2,
     media_Type     IN VARCHAR2 DEFAULT NULL)
 RETURN SODA_Document_T;

SODA_DOCUMENT_T (
     key            IN VARCHAR2 DEFAULT NULL,
     b_Content      IN BLOB,
     media_Type     IN VARCHAR2 DEFAULT NULL)
 RETURN SODA_Document_T;

SODA_DOCUMENT_T (
     key           IN VARCHAR2 DEFAULT NULL,
     c_Content     IN CLOB,
     media_Type    IN VARCHAR2 DEFAULT NULL)
 RETURN SODA_Document_T;

SODA_DOCUMENT_T (
     key           IN VARCHAR2 DEFAULT NULL,
     j_Content     IN JSON)
 RETURN SODA_Document_T;
```

Parameters

Table 317-17 SODA_Document_T Parameters

Parameter	Description
`key`	The key of the document.
`v_Content`	The content of the document in `VARCHAR2` format.
`b_Content`	The content of the document in `BLOB` format.
`c_Content`	The content of the document in `CLOB` format.
`j_Content`	The content of the document as `JSON` type instance.
`media_Type`	The media type of the document. The media type could be `application/json` for `JSON` documents.

Note:

`v_Content`, `b_Content`, and `c_Content` are not all parameters of a single `SODA_DOCUMENT_T` constructor function. Each one corresponds to a particular variant of the constructor function as shown in the `Syntax` section.

Return Values

This function returns a document of type `SODA_Document_T`.

SODA_OPERATION_T Type

This `SODA` type is used to perform read/write operations, such as document finds with filtering and pagination, removes, and replaces on a `SODA` collection.

Table 317-18 SODA_OPERATION_T Type Subprograms

Subprogram	Description
ACQUIRE_LOCK Function	This function ensures that the document(s) affected by a `read` operation are locked for update (which is equivalent to SQL '`select for update`').

Table 317-18 (Cont.) SODA_OPERATION_T Type Subprograms

Subprogram	Description
AS_OF_SCN Function	This function sets the SCN value for the operation.
AS_OF_TIMESTAMP Function	This function sets the the timestamp value for the operation.
COUNT Function	Returns a count of the number of documents in the collection that match the criteria. If skip(...) or limit(...) were chained together with this count(), an exception is raised.
FILTER Function	Sets the filter (also known as QBE or query-by-example) criteria on the operation. Returns the same `SODA_OPERATION_T` object so that further criteria can be chained together if required.
GET_CURSOR Function	Returns a `SODA_CURSOR_T` object that can be used to iterate over the documents that match the criteria.
GET_DATA_GUIDE Function	This function gets the data guide.
GET_ONE Function	Returns a single `SODA_DOCUMENT_T` object that matches the criteria. Note that, if multiple documents match the criteria, only the first document is returned.
HINT Function	This function sets the hint attribute of the operation.
KEY Function	Specifies that the document with the specified key should be returned. This causes any previous calls made to this function and `KEYS`(...), when they appear in the same chain, to be ignored. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.
KEYS Function	Specifies that documents that match the keys supplied to this function should be returned. This causes any previous calls made to this function and `KEY`(...), when they appear in the same chain, to be ignored. Returns the same `SODA_OPERATION_T` object, so that further operation criteria can be chained together, if needed.
LIMIT Function	Sets a limit on the specified number of documents the operation should return. This setting is only usable for read operations such as `GET_CURSOR`. For write operations, any value set using this method is ignored. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.
REMOVE Function	Removes all of the documents in the collection that match the criteria. Returns the number of documents that was removed.

Table 317-18 (Cont.) SODA_OPERATION_T Type Subprograms

Subprogram	Description
REPLACE_ONE Function	Replaces a single document in the collection with the specified document. Returns a number that indicates if the document was replaced or not. Currently, before calling this function, you must call the function `KEY`(...) to uniquely identify the document being replaced. Any components set in the input document with the exception of content and media type are not used during the replace. They are ignored.
REPLACE_ONE_AND_GET Function	Replaces a single document in the collection with the specified document. Returns a result document if the document was replaced, `NULL` otherwise. Currently, before calling this function, you must call the function `KEY`(...) to uniquely identify the document being replaced. This function is similar to `REPLACE_ONE`. The only difference is that `REPLACE_ONE_AND_GET` also returns the result document with updated components, such as version and last-modified timestamp. The result document does not contain the content component. Any components set in the input document with the exception of content and media type are not used during the replace. They are ignored.
SAMPLE Function	Sets the sampling parameters to be used for the operation.
SKIP Function	Sets the number of documents that match the operation criteria that will be skipped from the operation result. This setting is only usable for read operations such as `GET_CURSOR`. For write operations, any value set using this method is ignored. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.
VERSION Function	Specifies that only documents with the supplied version should be returned. Typically, this is chained together with `KEY`(...) to implement optimistic locking for write operations such as `REMOVE` and `REPLACE`. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.

ACQUIRE_LOCK Function

This function ensures that the the document(s) affected by a read operation are locked for update (which is equivalent to SQL '`select for update`'). An operation involving `LOCK()` would be followed by another operation that updates the document in the collection. For example, via

REPLACE or REMOVE functions. The lock will prevent other transactions from modifying the document in the meantime.

The next commit or rollback, performed after the operation involving this ACQUIRE_LOCK() function, will unlock the document, i.e. commit or rollback the transaction which is holding the lock on this document.

This function should only be used in conjunction with read operations, other than COUNT() and GET_DATA_GUIDE() functions.

Specifying it in conjunction with SKIP and LIMIT functions is also not allowed (an error is thrown).

If specified in conjunction with a write operation, such as, REPLACE or REMOVE, it's simply ignored and has no effect.

Syntax

```
ACQUIRE_LOCK ( )
 RETURN SODA_Operation_T;
```

Return Values

This function returns the same SODA_OPERATION_T object it was invoked on.

AS_OF_SCN Function

This function sets the SCN value for the operation.

Syntax

```
AS_OF_SCN (
     scn         IN NUMBER)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-19 AS_OF_SCN Function Parameters

Parameter	Description
scn	The input value for SCN. This cannot be NULL.

Return Values

This function returns the same SODA_OPERATION_T object it was invoked on.

AS_OF_TIMESTAMP Function

This function sets the the timestamp value for the operation.

Syntax

```
AS_OF_TIMESTAMP (
     timestamp         IN VARCHAR2)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-20 AS_OF_TIMESTAMP Function Parameters

Parameter	Description
`timestamp`	The input timestamp. This cannot be `NULL`.

Return Values

This function returns the same `SODA_OPERATION_T` object it was invoked on.

COUNT Function

This function returns a count of the number of documents in the collection that match the criteria. If skip(...) or limit(...) were chained together with this count(), an exception is raised.

Syntax

```
COUNT ()
 RETURN NUMBER;
```

Return Values

This function returns the number of documents matching the criteria specified in the operation.

Exceptions

`Error`—If an error occurs while finding the count.

FILTER Function

Sets the filter (also known as QBE or query-by-example) criteria on the operation. Returns the same `SODA_OPERATION_T` object so that further criteria can be attached if needed.

Syntax

```
FILTER (
     qbe         IN VARCHAR2)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-21 FILTER Function Parameters

Parameter	Description
`qbe`	The string representing the query by example.

Return Values

This function returns the same SODA_OPERATION_T object it was invoked on.

See Also:

- Overview of SODA Filter Specifications (QBEs)
- SODA Filter Specifications (Reference)

GET_CURSOR Function

Returns a `SODA_CURSOR_T` object that can be used to iterate over the documents that match the criteria.

Syntax

(Optional) Enter syntax information here.

```
GET_CURSOR ()
 RETURN SODA_Cursor_T;
```

Return Values

This function returns a `SODA_CURSOR_T` object that can be used to iterate over the documents that match the read operation criteria.

Exceptions

`SODA Error`: If an error occurs while fetching the cursor.

GET_DATA_GUIDE Function

This function gets the data guide.

Syntax

```
GET_DATA_GUIDE (
  format    IN PLS_INTEGER DEFAULT 1,
  flag      IN PLS_INTEGER DEFAULT 0)
 RETURN CLOB;
```

Parameters

Table 317-22 GET_DATA_GUIDE Function Parameters

Parameter	Description
`format`	The format of the data guide. This parameter can have one of the following values: • `DBMS_SODA.DATAGUIDE_FORMAT_HIERARCHICAL CONSTANT PLS_INTEGER` := 1; • `DBMS_SODA.DATAGUIDE_FORMAT_FLAT CONSTANT PLS_INTEGER` := 2; The default value is `1`.

Table 317-22 (Cont.) GET_DATA_GUIDE Function Parameters

Parameter	Description
`flag`	This parameter can have one of the following values: • `DBMS_SODA.DATAGUIDE_PRETTY CONSTANT PLS_INTEGER` := 1; • `DBMS_SODA.DATAGUIDE_GEOJSON CONSTANT PLS_INTEGER` := 2; • `DBMS_SODA.DATAGUIDE_GATHER_STATS CONSTANT PLS_INTEGER` := 4; The default value is `0`.

Return Values

This function returns the same `SODA_OPERATION_T` object it was invoked on.

GET_ONE Function

Returns a single `SODA_DOCUMENT_T` object that matches the criteria. Note that, if multiple documents match the criteria, only the first document is returned.

Syntax

```
GET_ONE ()
 RETURN SODA_Document_T;
```

Return Values

The first matching document.

Exceptions

`Error`—If an error occurs while fetching the document.

HINT Function

This function sets the hint attribute of the operation.

Syntax

```
HINT (
     hint         IN VARCHAR2)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-23 HINT Function Parameters

Parameter	Description
`hint`	A hint string in Oracle SQL format, with out the enclosing `/*+` and `*/`.

Return Values

This function returns the same SODA_OPERATION_T object it was invoked on.

KEY Function

Specifies that the document with the specified key should be returned. This causes any previous calls made to this function and KEYS(...), when they appear in the same chain, to be ignored. Returns the same SODA_OPERATION_T object so that further operation criteria can be chained together, if needed.

Syntax

```
KEY (
     key          IN VARCHAR2)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-24 KEY Function Parameters

Parameter	Description
key	The key to be used for the operations.

Return Values

This function returns the same SODA_OPERATION_T object it was invoked on.

KEYS Function

Specifies that documents that match the keys supplied to this function should be returned. This causes any previous calls made to this function and key(...), when they appear in the same chain, to be ignored. Returns the same SODA_OPERATION_T object, so that further operation criteria can be chained together, if needed.

Syntax

```
KEYS (
     key_List   IN SODA_Key_List_T)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-25 KEYS Function Parameters

Parameter	Description
key_List	The parameter is a SODA_Key_List_T which is a list of VARCHAR2 values representing keys. Assuming key_list is a variable of type SODA_Key_List_T, it can be initialized as follows: `key_list := SODA_Key_List_T('key1', 'key2', 'key3', etc);`

Return Values

This function returns the same `SODA_OPERATION_T` object it was invoked on.

LIMIT Function

This function sets a limit on the specified number of documents the operation should return. This setting is only usable for read operations such as `GET_CURSOR`. For write operations, any value set using this method is ignored. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.

Syntax

```
LIMIT (
     limit          IN NUMBER)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-26 LIMIT Function Parameters

Parameter	Description
`limit`	A limit on the number of results returned by read operations.

Return Values

This function returns the same `SODA_OPERATION_T` object it was invoked on.

REMOVE Function

This function removes all of the documents in the collection that match the criteria. Returns the number of documents that was removed.

Syntax

```
REMOVE ()
 RETURN NUMBER;
```

Return Values

This function returns the number of matching documents that were removed in the operation.

Exceptions

`Error`—If an error occurs while removing the documents.

REPLACE_ONE Function

This function replaces a single document in the collection with the specified document. Returns a number that indicates if the document was replaced or not. Currently, before calling this function, you must call the function `KEY`(...) to uniquely identify the document being replaced. Any components set in the input document with the exception of content and media type are not used during the replace. They are ignored.

Syntax

```
REPLACE_ONE (
      document IN SODA_Document_T)
 RETURN NUMBER;
```

Parameters

Table 317-27 REPLACE_ONE Function Parameters

Parameter	Description
`document`	The document object with the new content and media type to be used for replacement.

Return Values

This function returns a number—`1` if the document was replaced, `0` otherwise.

Exceptions

`Error`—If an error occurs while updating the collection.

REPLACE_ONE_AND_GET Function

Replaces a single document in the collection with the specified document. Returns a result document if the document was replaced, `NULL` otherwise. Currently, before calling this function, you must call the function `KEY`(...) to uniquely identify the document being replaced. This function is similar to `REPLACE_ONE`. The only difference is that `REPLACE_ONE_AND_GET` also returns the result document with updated components, such as version and last-modified timestamp. The result document does not contain the content component. Any components set in the input document with the exception of content and media type are not used during the replace. They are ignored.

Syntax

```
REPLACE_ONE_AND_GET (
     document         IN SODA_Document_T)
 RETURN SODA_Document_T;
```

Parameters

Table 317-28 REPLACE_ONE_AND_GET Function Parameters

Parameter	Description
`document`	The document object with the new content and media type to be used for replacement.

Return Values

The function returns the result document containing all document components supported by the given collection, with the exception of content. Last-modified and version components, if supported by the given collection, will be updated with new values. If no document in the collection had the supplied key, `NULL` is returned instead of the result document.

Exceptions

`Error`—If an error occurs while updating the collection

SAMPLE Function

This function sets the sampling parameters to be used for the operation.

Syntax

```
SAMPLE (
     pct     IN   DOUBLE,
     seed    IN   NUMBER      DEFAULT NULL,
     method  IN   PLS_INTEFER DEFAULT DBMS_SODA.SAMPLE_ROW)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-29 SAMPLE Function Parameters

Parameter	Description
`pct`	The percentage of the total documents or block count to be included in the sample. The value must be in the range `.000001` to, but not including, `100`. This percentage indicates the probability of each row or each cluster of rows in the case of block sampling, being selected as part of the sample. It does not mean that the database will retrieve exactly the percentage of documents in the collection.
`seed`	Specify this attribute to instruct the database to attempt to return the same sample from one execution to the next. The seed value must be an integer between `0` and `4294967295`. If you omit this attribute, then the resulting sample will change from one execution to the next.
`method`	The type of sampling method to be used. Valid values are: • `DBMS_SODA.SAMPLE_ROW` • `DBMS_SODA.SAMPLE_BLOCK`

Return Values

This function returns the `SELF` operation object.

SKIP Function

This function sets the number of documents that match the operation criteria that will be skipped from the operation result. This setting is only usable for read operations such as `GET_CURSOR`. For write operations, any value set using this method is ignored. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.

Syntax

```
SKIP (
     offset          IN NUMBER)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-30 SKIP Function Parameters

Parameter	Description
`offset`	The number of documents to skip.

Return Values

This function returns the same `SODA_OPERATION_T` object it was invoked on

VERSION Function

This function specifies that only documents with the supplied version should be returned. Typically, this is chained together with `KEY`(...) to implement optimistic locking for write operations such as `REMOVE` and `REPLACE`. Returns the same `SODA_OPERATION_T` object so that further operation criteria can be chained together, if needed.

Syntax

```
VERSION (
     version          IN VARCHAR2)
 RETURN SODA_Operation_T;
```

Parameters

Table 317-31 VERSION Function Parameters

Parameter	Description
`version`	Document version to be used for the operation.

Return Values

This function returns the same `SODA_OPERATION_T` object it was invoked on

SODA_CURSOR_T Type

This `SODA` type is used to represent a result set of documents.

Table 317-32 SODA_CURSOR_T Type Subprograms

Subprogram	Description
CLOSE Function	Closes the cursor.
HAS_NEXT Function	Returns `TRUE`, if the next document is available for the cursor. Otherwise, returns `FALSE`.

Table 317-32 (Cont.) SODA_CURSOR_T Type Subprograms

Subprogram	Description
NEXT Function	Returns the next SODA documented pointed by the cursor.

CLOSE Function

This function closes the cursor.

Syntax

```
CLOSE ()
 RETURN BOOLEAN;
```

Example 317-3 Return Values

This function returns a boolean value.

HAS_NEXT Function

This function returns `TRUE`, if the next document is available for the cursor. Otherwise, returns `FALSE`.

Syntax

```
HAS_NEXT ()
 RETURN BOOLEAN;
```

Return Values

This function returns a boolean value. `TRUE`, if the next document is available for the cursor. Otherwise, returns `FALSE`.

Exceptions

`Error`—If an error occurs while checking if the next document is available.

NEXT Function

This function returns the next `SODA` documented pointed by the cursor.

Syntax

```
NEXT ()
 RETURN SODA_Document_T;
```

Return Values

This function returns the next `SODA` documented pointed by the cursor. Returns `NULL` when the `HAS_NEXT function` returns `FALSE`.

Exceptions

`Error`—If an error occurs while getting the next document.