

Oracle® Database

SODA for PL/SQL Developer's Guide



Release 18c

E84719-01

February 2018

The Oracle logo, consisting of a solid red square with the word "ORACLE" in white, uppercase, sans-serif font centered within it.

ORACLE®

Oracle Database SODA for PL/SQL Developer's Guide, Release 18c

E84719-01

Copyright © 2018, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Author: Drew Adams

Contributors: Douglas McMahon, Maxim Orgiyen, Srikrishnan Suresh

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

	Preface	
	Audience	vii
	Documentation Accessibility	vii
	Related Documents	vii
	Conventions	viii
1	SODA for PL/SQL Prerequisites	
2	SODA for PL/SQL Overview	
3	Using SODA for PL/SQL	
3.1	Getting Started with SODA for PL/SQL	3-2
3.2	Creating a Document Collection with SODA for PL/SQL	3-4
3.3	Opening an Existing Document Collection with SODA for PL/SQL	3-6
3.4	Checking Whether a Given Collection Exists with SODA for PL/SQL	3-6
3.5	Discovering Existing Collections with SODA for PL/SQL	3-7
3.6	Dropping a Document Collection with SODA for PL/SQL	3-8
3.7	Creating Documents with SODA for PL/SQL	3-9
3.8	Inserting Documents into Collections with SODA for PL/SQL	3-13
3.9	Finding Documents in Collections with SODA for PL/SQL	3-15
3.10	Replacing Documents in a Collection with SODA for PL/SQL	3-16
3.11	Removing Documents from a Collection with SODA for PL/SQL	3-17
3.12	Handling Transactions with SODA for PL/SQL	3-18
4	SODA Collection Configuration Using Custom Metadata	
4.1	Getting the Metadata of an Existing Collection	4-2
4.2	Creating a Collection That Has Custom Metadata	4-3

Index

List of Examples

3-1	Getting Started Run-Through	3-3
3-2	Sample Output for Getting Started Run-Through	3-4
3-3	Creating a Collection That Has the Default Metadata	3-5
3-4	Opening an Existing Document Collection	3-6
3-5	Checking for a Collection with a Given Name	3-6
3-6	Printing the Names of All Existing Collections	3-7
3-7	Dropping a Document Collection	3-8
3-8	Creating a Document with JSON Content	3-11
3-9	Creating a Document with Document Key and JSON Content	3-12
3-10	Inserting a Document into a Collection	3-14
3-11	Inserting a Document into a Collection and Getting the Result Document	3-14
3-12	Finding the Unique Document That Has a Given Document Key	3-15
3-13	Replacing a Document in a Collection, Given Its Key, and Getting the Result Document	3-16
3-14	Removing a Document from a Collection Using a Document Key	3-17
3-15	Transaction Involving SODA Document Insertion and Replacement	3-18
4-1	Getting the Metadata of a Collection	4-2
4-2	Default Collection Metadata	4-2
4-3	Creating a Collection That Has Custom Metadata	4-3

List of Tables

3-1	Getter Methods for Documents (SODA_DOCUMENT_T)	3-10
-----	------------------------------------------------	------

Preface

This document describes how to use Simple Oracle Document Access (SODA) for C.

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documents](#)
- [Conventions](#)

Audience

This document is intended for users of Simple Oracle Document Access (SODA) for PL/SQL.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, see these Oracle resources:

- *Oracle Database Introduction to Simple Oracle Document Access (SODA)* for general information about SODA
- *Oracle Database PL/SQL Language Reference* for general and reference information about the PL/SQL language
- *Oracle as a Document Store* for general information about using JSON data in Oracle Database, including with SODA
- *Oracle Database JSON Developer's Guide* for information about using SQL and PL/SQL with JSON data stored in Oracle Database
- *Oracle Database Error Messages Reference*

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

OTN Registration.

If you already have a user name and password for OTN then you can go directly to the documentation section of the OTN Web site at

OTN Documentation.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
<code>monospace</code>	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

SODA for PL/SQL Prerequisites

SODA for PL/SQL is an integral part of Oracle Database, starting with Release 18c (18.1). The database is the only prerequisite for using SODA for PL/SQL.

2

SODA for PL/SQL Overview

SODA for PL/SQL is a PL/SQL API that implements **Simple Oracle Document Access** (SODA). You can use it with PL/SQL to perform create, read (retrieve), update, and delete (CRUD) operations on documents of any kind, and you can use it to query JSON documents.

SODA is a set of NoSQL-style APIs that let you create and store collections of documents in Oracle Database, retrieve them, and query them, without needing to know Structured Query Language (SQL) or how the data in the documents is stored in the database.

Oracle relational database management system (RDBMS) supports storing and querying JSON data. To access this functionality, you need structured query language (SQL) with special JSON SQL operators. SODA for PL/SQL hides the complexities of SQL/JSON programming.

The remaining topics of this document describe various features of SODA for PL/SQL.

Note:

- This book provides information about using SODA with PL/SQL applications, and it describes all SODA features currently available for use with PL/SQL. To use SODA for PL/SQL you also need to understand SODA generally. For such general information, please consult *Oracle Database Introduction to Simple Oracle Document Access (SODA)*. Some features described in that book are not yet available with SODA for C.
- This book does not provide general information about PL/SQL, including reference information about the SODA for PL/SQL methods and constants. For such information, please consult *Oracle Database PL/SQL Language Reference*.

See Also:

Oracle Database JSON Developer's Guide for information about using SQL and PL/SQL with JSON data stored in Oracle Database

3

Using SODA for PL/SQL

How to access SODA for PL/SQL is described, as well as how to use it to perform create, read (retrieve), update, and delete (CRUD) operations on collections.

(CRUD operations are also called “read and write operations” in this document.)

- [Getting Started with SODA for PL/SQL](#)
How to access SODA for PL/SQL is described, as well as how to use it to create a database collection, insert a document into a collection, and retrieve a document from a collection.
- [Creating a Document Collection with SODA for PL/SQL](#)
You can use PL/SQL function `DBMS_SODA.create_collection` to create a document collection with the default metadata.
- [Opening an Existing Document Collection with SODA for PL/SQL](#)
You can use PL/SQL function `DBMS_SODA.open_collection` to open an existing document collection.
- [Checking Whether a Given Collection Exists with SODA for PL/SQL](#)
You can use PL/SQL function `DBMS_SODA.open_collection` to check for the existence of a given collection. It returns a SQL `NULL` value if a collection with the specified name does not exist; otherwise, it returns the collection object.
- [Discovering Existing Collections with SODA for PL/SQL](#)
You can use PL/SQL function `DBMS_SODA.list_collection_names` to discover existing collections.
- [Dropping a Document Collection with SODA for PL/SQL](#)
You use PL/SQL function `DBMS_SODA.drop_collection` to drop a document collection.
- [Creating Documents with SODA for PL/SQL](#)
You use a constructor for PL/SQL object type `SODA_DOCUMENT_T` to create SODA documents.
- [Inserting Documents into Collections with SODA for PL/SQL](#)
To insert a document into a collection, you invoke `SODA_COLLECTION_T` method (member function) `insert_one()` or `insert_one_and_get()`. These methods create document keys automatically, unless the collection is configured with client-assigned keys and the input document provides the key.
- [Finding Documents in Collections with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method `find_one()` to find the unique document in a collection that has a given key.
- [Replacing Documents in a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method (member function) `replace_one()` or `replace_one_and_get()` to replace the content of a document in a collection, given its key. The latter method also gets the components of the result document (except for the updated content).

- [Removing Documents from a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method `remove_one()` to remove a document from a collection, given its key.
- [Handling Transactions with SODA for PL/SQL](#)
As usual in PL/SQL and SQL, you can treat individual SODA read and write operations, or groups of them, as a transaction. To commit a transaction, use a `SQL COMMIT` statement. If you want to roll back changes, use a `SQL ROLLBACK` statement.

3.1 Getting Started with SODA for PL/SQL

How to access SODA for PL/SQL is described, as well as how to use it to create a database collection, insert a document into a collection, and retrieve a document from a collection.

Note:

Don't worry if not everything in this topic is clear to you on first reading. The necessary concepts are developed in detail in other topics. This topic should give you an idea of what is involved overall in using SODA.

Follow these steps to get started with SODA for PL/SQL:

1. Ensure that the prerequisites have been met for using SODA for PL/SQL. See [SODA for PL/SQL Prerequisites](#).
2. Identify the database schema (user account) used to store collections, and grant database role `SODA_APP` to that schema:

```
GRANT SODA_APP TO schemaName;
```
3. Use PL/SQL code such as that in [Example 3-1](#) to do the following:
 - a. Create and open a collection (an instance of PL/SQL object type `SODA_COLLECTION_T`), using the default collection configuration (metadata).
 - b. Create a document with particular JSON content, as an instance of PL/SQL object type `SODA_DOCUMENT_T`.
 - c. Insert the document into the collection.
 - d. Get the inserted document back. Its other components, besides the content, are generated automatically.
 - e. Print the unique document key, which is one of the components generated automatically.
 - f. Commit the document insertion.
 - g. Find the document in the collection, by providing its key.
 - h. Print some of the document components: key, content, creation timestamp, last-modified timestamp, and version.
4. Drop the collection, cleaning up the database table that is used to store the collection and its metadata:

```
SELECT DBMS_SODA.drop_collection('myCollectionName') AS drop_status FROM DUAL;
```

▲ Caution:

Do *not* use SQL to drop the database *table* that underlies a collection. Dropping a *collection* involves more than just dropping its database table. In addition to the documents that are stored in its table, a collection has *metadata*, which is also persisted in Oracle Database. Dropping the table underlying a collection does *not* also drop the collection metadata.

✎ Note:

If a PL/SQL subprogram that you write invokes subprograms that are in package `DBMS_SODA`, and if your subprogram has definer (owner) rights, then a database administrator (DBA) must grant role `SODA_APP` to your subprogram. For example, this code grants role `SODA_APP` to procedure `my_soda_proc`, which is owned by database schema (user) `my_db_schema`:

```
GRANT SODA_APP TO PROCEDURE my_db_schema.my_soda_proc;
```

✎ See Also:

Oracle Database Security Guide for information about role `SODA_APP`

Example 3-1 Getting Started Run-Through

```
DECLARE
    collection      SODA_COLLECTION_T;
    document        SODA_DOCUMENT_T;
    foundDocument  SODA_DOCUMENT_T;
    result_document SODA_DOCUMENT_T;
    docKey         VARCHAR2(100);
    status         NUMBER;
BEGIN
    -- Create a collection.
    collection := DBMS_SODA.create_collection('myCollectionName');

    -- The default collection has BLOB content, so create a BLOB-based document.
    document := SODA_DOCUMENT_T(
        b_content => utl_raw.cast_to_raw('{ "name" : "Alexander" }'));

    -- Insert the document and get it back.
    result_document := collection.insert_one_and_get(document);

    -- The result document has auto-generated components, such as key and version,
    -- in addition to the content. Print the auto-generated document key.
    docKey := result_document.get_key;
    DBMS_OUTPUT.put_line('Auto-generated key is ' || docKey);

    -- Commit the insert
    COMMIT;

    -- Find the document in the collection by its key
```

```

foundDocument := collection.find_one(docKey);

-- Get and print some document components: key, content, etc.
DBMS_OUTPUT.put_line('Document components:');
DBMS_OUTPUT.put_line(' Key: ' || foundDocument.get_key);
DBMS_OUTPUT.put_line(' Content: '
                    || utl_raw.cast_to_varchar2(foundDocument.get_blob));
DBMS_OUTPUT.put_line(' Creation timestamp: ' || foundDocument.get_created_on);
DBMS_OUTPUT.put_line(' Last-modified timestamp: '
                    || foundDocument.get_last_modified);
DBMS_OUTPUT.put_line(' Version: ' || foundDocument.get_version);
END;
/

```

Example 3-2 Sample Output for Getting Started Run-Through

[Example 3-1](#) results in output similar to this. The values of the auto-generated components will differ in any actual execution.

```

Auto-generated key is 96F35328CD3B4F96BF3CD01BCE9EBDF5
Document components:
Key: 96F35328CD3B4F96BF3CD01BCE9EBDF5
Content: {"name" : "Alexander"}
Creation timestamp: 2017-09-19T01:05:06.160289Z
Last-modified timestamp: 2017-09-19T01:05:06.160289Z
Version: FD69FB6ACE73FA735EC7922CA4A02DDE0690462583F9EA2AF754D7E342B3EE78

```

3.2 Creating a Document Collection with SODA for PL/SQL

You can use PL/SQL function `DBMS_SODA.create_collection` to create a document collection with the default metadata.

[Example 3-3](#) uses PL/SQL function `DBMS_SODA.create_collection` to create a collection that has the default metadata.

The *default collection metadata* has the following characteristics.

- Each document in the collection has these document components:
 - Key
 - Content
 - Creation timestamp
 - Last-modified timestamp
 - Version
- The collection can store only JSON documents.
- Document keys are automatically generated for documents that you add to the collection.

The default collection configuration is recommended in most cases, but collections are highly configurable. When you create a collection you can specify things such as the following:

- Storage details, such as the name of the table that stores the collection and the names and data types of its columns.
- The presence or absence of columns for creation timestamp, last-modified timestamp, and version.

- Whether the collection can store only JSON documents.
- Methods of document key generation, and whether document keys are client-assigned or generated automatically.
- Methods of version generation.

This configurability also lets you map a new collection to an existing database table.

To configure a collection in a nondefault way, supply custom collection metadata, expressed in JSON, as the second argument to `DBMS_SODA.create_collection`.

If you do not care about the details of collection configuration then pass only the collection name to `DBMS_SODA.create_collection` — no second argument. That creates a collection with the default configuration.

If a collection with the same name already exists then it is simply opened and its handle is returned. If custom metadata is provided and it does not match the metadata of the existing collection then the collection is not opened and an error is raised. (To match, all metadata fields must have the same values.)

Note:

Unless otherwise stated, the remainder of this documentation assumes that a collection has the default configuration.

See Also:

- *Oracle Database Introduction to Simple Oracle Document Access (SODA)* for information about the default naming of a collection table
- *Oracle Database PL/SQL Packages and Types Reference* for information about PL/SQL function `DBMS_SODA.create_collection`

Example 3-3 Creating a Collection That Has the Default Metadata

This example creates collection `myCollectionName` with the default metadata.

```
DECLARE
    collection SODA_Collection_T;
BEGIN
    collection := DBMS_SODA.create_collection('myCollectionName');
END;
/
```

Related Topics

- [Getting the Metadata of an Existing Collection](#)
You use `SODA_COLLECTION_T` method `get_metadata()` to get all of the metadata for a collection, as a JSON document.
- [Creating a Collection That Has Custom Metadata](#)
To create a document collection that has custom metadata, you pass its metadata, as JSON data, to PL/SQL function `DBMS_SODA.create_collection`.

- **Checking Whether a Given Collection Exists with SODA for PL/SQL**
You can use PL/SQL function `DBMS_SODA.open_collection` to check for the existence of a given collection. It returns a SQL `NULL` value if a collection with the specified name does not exist; otherwise, it returns the collection object.

3.3 Opening an Existing Document Collection with SODA for PL/SQL

You can use PL/SQL function `DBMS_SODA.open_collection` to open an existing document collection.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information about PL/SQL function `DBMS_SODA.open_collection`

Example 3-4 Opening an Existing Document Collection

This example uses PL/SQL function `DBMS_SODA.open_collection` to open the collection named `myCollectionName` and returns a `SODA_COLLECTION_T` instance that represents this collection. If the value returned is `NULL` then there is no existing collection named `myCollectionName`.

```
DECLARE
    collection  SODA_COLLECTION_T;
BEGIN
    collection := DBMS_SODA.open_collection('myCollectionName');
END;
/
```

3.4 Checking Whether a Given Collection Exists with SODA for PL/SQL

You can use PL/SQL function `DBMS_SODA.open_collection` to check for the existence of a given collection. It returns a SQL `NULL` value if a collection with the specified name does not exist; otherwise, it returns the collection object.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information about PL/SQL function `DBMS_SODA.open_collection`

Example 3-5 Checking for a Collection with a Given Name

This example uses `DBMS_SODA.open_collection` to try to open an existing collection named `myCollectionName`. It prints a message if no such collection exists.


```

DECLARE
    collection SODA_COLLECTION_T;
BEGIN
    collection := DBMS_SODA.open_collection('myCollectionName');
    IF collection IS NULL THEN
        DBMS_OUTPUT.put_line('Collection does not exist');
    END IF;
END;
/

```

Related Topics

- [Creating a Document Collection with SODA for PL/SQL](#)
You can use PL/SQL function `DBMS_SODA.create_collection` to create a document collection with the default metadata.

3.5 Discovering Existing Collections with SODA for PL/SQL

You can use PL/SQL function `DBMS_SODA.list_collection_names` to discover existing collections.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information about PL/SQL function `DBMS_SODA.list_collection_names`

Example 3-6 Printing the Names of All Existing Collections

This example uses PL/SQL function `DBMS_SODA.list_collection_names` to obtain a list of the collection names. It then iterates over that list, printing out the names.

```

DECLARE
    coll_list SODA_COLLNAME_LIST_T;
BEGIN
    coll_list := DBMS_SODA.list_collection_names;
    DBMS_OUTPUT.put_line('Number of collections: ' || to_char(coll_list.count));
    DBMS_OUTPUT.put_line('Collection List: ');
    IF (coll_list.count > 0) THEN
        -- Loop over the collection name list
        FOR i IN
            coll_list.first .. coll_list.last
        LOOP
            DBMS_OUTPUT.put_line(coll_list(i));
        END LOOP;
    ELSE
        DBMS_OUTPUT.put_line('No collections found');
    END IF;
END;
/

```

3.6 Dropping a Document Collection with SODA for PL/SQL

You use PL/SQL function `DBMS_SODA.drop_collection` to drop a document collection.

Caution:

Do *not* use SQL to drop the database *table* that underlies a collection. Dropping a *collection* involves more than just dropping its database table. In addition to the documents that are stored in its table, a collection has *metadata*, which is also persisted in Oracle Database. Dropping the table underlying a collection does *not* also drop the collection metadata.

Note:

Commit all writes to a collection *before* using `DBMS_SODA.drop_collection`. For the drop to succeed, all uncommitted writes to the collection must first be either committed or rolled back — you must explicitly use SQL `COMMIT` or `ROLLBACK`. Otherwise, an exception is raised.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information about PL/SQL function `DBMS_SODA.drop_collection`

Example 3-7 Dropping a Document Collection

This example uses PL/SQL function `DBMS_SODA.drop_collection` to drop collection `myCollectionName`.

If the collection cannot be dropped because of uncommitted write operations then an exception is thrown. If the collection is dropped successfully, the returned status is 1; otherwise, the status is 0. In particular, if a collection with the specified name does not exist, the returned status is 0 — no exception is thrown.

```
DECLARE
    status NUMBER := 0;
BEGIN
    status := DBMS_SODA.drop_collection('myCollectionName');
END;
/
```

Related Topics

- [Handling Transactions with SODA for PL/SQL](#)
As usual in PL/SQL and SQL, you can treat individual SODA read and write operations, or groups of them, as a transaction. To commit a transaction, use a SQL `COMMIT` statement. If you want to roll back changes, use a SQL `ROLLBACK` statement.

- [Inserting Documents into Collections with SODA for PL/SQL](#)
To insert a document into a collection, you invoke `SODA_COLLECTION_T` method (member function) `insert_one()` or `insert_one_and_get()`. These methods create document keys automatically, unless the collection is configured with client-assigned keys and the input document provides the key.
- [Replacing Documents in a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method (member function) `replace_one()` or `replace_one_and_get()` to replace the content of a document in a collection, given its key. The latter method also gets the components of the result document (except for the updated content).

3.7 Creating Documents with SODA for PL/SQL

You use a constructor for PL/SQL object type `SODA_DOCUMENT_T` to create SODA documents.

SODA for PL/SQL represents a document using an instance of PL/SQL object type `SODA_DOCUMENT_T`. This object is a *carrier* of document content and other document components, such as the document key.

Here is an example of the *content* of a JSON document:

```
{ "name" :    "Alexander",  
  "address" : "1234 Main Street",  
  "city" :    "Anytown",  
  "state" :   "CA",  
  "zip" :    "12345"  
}
```

A document has these **components**:

- Key
- Content
- Creation time stamp
- Last-modified time stamp
- Version
- Media type ("application/json" for JSON documents)

You *create a document* by invoking one of the `SODA_DOCUMENT_T` constructors. The constructors differ according to the content type of the documents they create: `VARCHAR2`, `CLOB`, or `BLOB`.

You can write a document of a given content type only to a collection whose content column has been defined for documents of that type. For example, you can write (insert or replace) only a document with content type `BLOB` to a collection whose `contentColumn` has a `sqlType` value of `BLOB`. (`BLOB` is the default content type for a collection.)

There are different ways to invoke a document constructor:

- You can provide the document *key*, as the first argument.

In a collection, each document must have a key. You must provide the key when you create the document *only* if you expect to insert the document into a collection

that does *not* automatically generate keys for inserted documents. By default, collections are configured to automatically generate document keys.

- You *must* provide the document *content*. If you also provide the document key then the content is the second argument to the constructor.

If you provide only the content then you must specify both the formal and actual content parameters, separated by the association arrow (`=>`): `v_content => actual`, `c_content => actual`, or `b_content => actual`, for content of type `VARCHAR2`, `CLOB`, or `BLOB`, respectively.

- You can provide the document media type, which defaults to `"application/json"`. Unless you provide all of the parameters (key, content, and media type) you must specify both the formal and actual media-type parameters, , separated by the association arrow (`=>`): `media_type => actual`.

Parameters that you do not provide explicitly default to `NULL`.

Providing only the content parameter can be useful for creating documents that you insert into a collection that automatically generates document keys. Providing only the key and content can be useful for creating documents that you insert into a collection that has client-assigned keys. Providing (the content and) the media type can be useful for creating *non*-JSON documents (using a media type other than `"application/json"`).

However you invoke a `SODA_DOCUMENT_T` constructor, doing so sets the components that you provide (the content, possibly the key, and possibly the media type) to the values you provide for them. And it sets the values of the creation time stamp, last-modified time stamp, and version to a SQL `NULL` value.

Object type `SODA_DOCUMENT_T` provides **getter methods** (also known as *getters*), which each retrieve a particular component from a document. (Getter `get_data_type()` actually returns information about the content component, rather than the component itself.)

Table 3-1 Getter Methods for Documents (SODA_DOCUMENT_T)

Getter Method	Description
<code>get_created_on()</code>	Get the <i>creation time stamp</i> for the document, as a <code>VARCHAR2</code> value.
<code>get_key()</code>	Get the unique <i>key</i> for the document, as a <code>VARCHAR2</code> value.
<code>get_last_modified()</code>	Get the <i>last-modified time stamp</i> for the document, as a <code>VARCHAR2</code> value.
<code>get_media_type()</code>	Get the <i>media type</i> for the document, as a <code>VARCHAR2</code> value.
<code>get_version()</code>	Get the document <i>version</i> , as a <code>VARCHAR2</code> value.
<code>get_blob()</code>	Get the document <i>content</i> , as a <code>BLOB</code> value. The document content must be <code>BLOB</code> data, or else an error is raised.
<code>get_clob()</code>	Get the document <i>content</i> , as a <code>CLOB</code> value. The document content must be <code>CLOB</code> data, or else an error is raised.

Table 3-1 (Cont.) Getter Methods for Documents (SODA_DOCUMENT_T)

Getter Method	Description
<code>get_varchar2()</code>	Get the document <i>content</i> , as a <code>VARCHAR2</code> value. The document content must be <code>VARCHAR2</code> data, or else an error is raised.
<code>get_data_type()</code>	Get the data type of the document content, as a <code>PLS_INTEGER</code> value. The value is <code>DBMS_SODA.DOC_VARCHAR2</code> for <code>VARCHAR2</code> content, <code>DBMS_SODA.DOC_BLOB</code> for <code>BLOB</code> content, and <code>DBMS_SODA.DOC_CLOB</code> for <code>CLOB</code> content.

Immediately after you create a document, the getter methods return these values:

- Values provided to the constructor
- "application/json", for method `get_media_type()`, if the media type was not provided
- `NULL` for other components

Each content storage data type has an associated content-component getter method. You must use the getter method that is appropriate to each content storage type: `get_varchar2()` for `VARCHAR2` storage, `get_clob()` for `CLOB` storage, and `get_blob()` for `BLOB` storage. Otherwise, an error is raised.

Example 3-8 creates a `SODA_DOCUMENT_T` instance, providing only content. The media type defaults to "application/json", and the other document components default to `NULL`.

Example 3-9 creates a `SODA_DOCUMENT_T` instance, providing the document key and content. The media type defaults to "application/json", and the other document components default to `NULL`.

See Also:

- *Oracle Database Introduction to Simple Oracle Document Access (SODA)* for an overview of SODA documents
- *Oracle Database Introduction to Simple Oracle Document Access (SODA)* for restrictions that apply for SODA documents
- *Oracle Database PL/SQL Packages and Types Reference* for information about object type `SODA_DOCUMENT_T` constructors and getter methods

Example 3-8 Creating a Document with JSON Content

This example uses `SODA_DOCUMENT_T` constructors to create three documents, one of each content type. The example provides only the document content (which is the same for each).

The content parameter is different in each case; it specifies the SQL data type to use to store the content. The first document creation here uses content parameter

`v_content`, which specifies `VARCHAR2` content; the second uses parameter `c_content`, which specifies `CLOB` content; the third uses parameter `b_content`, which specifies `BLOB` content.

After creating each document, the example uses getter methods to get the document content. Note that the getter method that is appropriate for each content storage type is used: `get_blob()` for `BLOB` content, and so on.

The document with content type `BLOB` would be appropriate for writing to the collection created in [Example 3-3](#), because that collection (which has the default metadata) accepts documents with (only) `BLOB` content. The other two documents would not be appropriate for that collection; trying to insert them would raise an error.

```
DECLARE
    v_doc SODA_DOCUMENT_T;
    b_doc SODA_DOCUMENT_T;
    c_doc SODA_DOCUMENT_T;
BEGIN
    - Create VARCHAR2 document
    v_doc := SODA_DOCUMENT_T(v_content => '{"name" : "Alexander"}');
    DBMS_OUTPUT.put_line('Varchar2 Doc content: ' || v_doc.get_varchar2);

    - Create BLOB document
    b_doc := SODA_DOCUMENT_T(
        b_content => utl_raw.cast_to_raw('{"name" : "Alexander"}'));
    DBMS_OUTPUT.put_line('Blob Doc content: ' ||
        utl_raw.cast_to_varchar2(b_doc.get_blob));

    - Create CLOB document
    c_doc := SODA_DOCUMENT_T(c_content => '{"name" : "Alexander"}');
    DBMS_OUTPUT.put_line('Clob Doc content: ' || c_doc.get_clob);
END;
/
```

Example 3-9 Creating a Document with Document Key and JSON Content

This example is similar to [Example 3-8](#), but it provides the document key (`myKey`) as well as the document content.

```
DECLARE
    v_doc SODA_DOCUMENT_T;
    b_doc SODA_DOCUMENT_T;
    c_doc SODA_DOCUMENT_T;
BEGIN
    - Create VARCHAR2 document
    v_doc := SODA_DOCUMENT_T('myKey' , v_content => '{"name" : "Alexander"}');
    DBMS_OUTPUT.put_line('Varchar2 Doc key: ' || v_doc.get_key);
    DBMS_OUTPUT.put_line('Varchar2 Doc content: ' || v_doc.get_varchar2);

    - Create BLOB document
    b_doc := SODA_DOCUMENT_T('myKey' ,
        b_content => utl_raw.cast_to_raw('{"name" :
"Alexander"}'));
    DBMS_OUTPUT.put_line('Blob Doc key: ' || b_doc.get_key);
    DBMS_OUTPUT.put_line('Blob Doc content: ' ||
        utl_raw.cast_to_varchar2(b_doc.get_blob));

    - Create CLOB document
    c_doc := SODA_DOCUMENT_T('myKey' , c_content => '{"name" : "Alexander"}');
    DBMS_OUTPUT.put_line('Clob Doc key: ' || c_doc.get_key);
    DBMS_OUTPUT.put_line('Clob Doc content: ' || c_doc.get_clob);
```

```
END;
/
```

Related Topics

- [Inserting Documents into Collections with SODA for PL/SQL](#)
To insert a document into a collection, you invoke `SODA_COLLECTION_T` method (member function) `insert_one()` or `insert_one_and_get()`. These methods create document keys automatically, unless the collection is configured with client-assigned keys and the input document provides the key.
- [Finding Documents in Collections with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method `find_one()` to find the unique document in a collection that has a given key.
- [Replacing Documents in a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method (member function) `replace_one()` or `replace_one_and_get()` to replace the content of a document in a collection, given its key. The latter method also gets the components of the result document (except for the updated content).
- [Removing Documents from a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method `remove_one()` to remove a document from a collection, given its key.

3.8 Inserting Documents into Collections with SODA for PL/SQL

To insert a document into a collection, you invoke `SODA_COLLECTION_T` method (member function) `insert_one()` or `insert_one_and_get()`. These methods create document keys automatically, unless the collection is configured with client-assigned keys and the input document provides the key.

Both method `insert_one()` and method `insert_one_and_get()` insert a document into a collection and automatically set the values of the creation time stamp, last-modified time stamp, and version (if the collection is configured to include these components and to generate the version automatically, as is the case by default).

When you insert a document, any document components that currently have `NULL` values (as a result of creating the document without providing those component values) are updated to have appropriate, automatically generated values. Thereafter, other SODA operations on a document can automatically update the last-modified timestamp and version components.

In addition to inserting the document, `insert_one_and_get` returns a result document, which contains the generated document components, such as the key, and which does not contain the content of the inserted document.

Note:

If the collection is configured with client-assigned document keys (which is not the default case), and the input document provides a key that identifies an existing document in the collection, then these methods throw an exception.

 **See Also:**

- *Oracle Database PL/SQL Packages and Types Reference* for information about SODA_COLLECTION_T method `insert_one()`
- *Oracle Database PL/SQL Packages and Types Reference* for information about SODA_COLLECTION_T method `insert_one_and_get()`
- *Oracle Database PL/SQL Packages and Types Reference* for information about SODA_DOCUMENT_T getter methods

Example 3-10 Inserting a Document into a Collection

This example creates a document and inserts it into a collection using SODA_COLLECTION_T method `insert_one()`.

```

DECLARE
    collection SODA_COLLECTION_T;
    document SODA_DOCUMENT_T;
    status NUMBER;
BEGIN
    -- Open the collection
    collection := DBMS_SODA.open_collection('myCollectionName');
    document := SODA_DOCUMENT_T(
        b_content => utl_raw.cast_to_raw('{ "name" : "Alexander" }'));

    -- Insert a document
    status := collection.insert_one(document);
END;
/

```

Example 3-11 Inserting a Document into a Collection and Getting the Result Document

This example creates a document and inserts it into a collection using method `insert_one_and_get()`. It then gets (and prints) each of the generated components from the result document (which contains them). To obtain the components it uses SODA_DOCUMENT_T methods `get_key()`, `get_created_on()`, `get_last_modified()`, and `get_version()`.

```

DECLARE
    collection SODA_COLLECTION_T;
    document SODA_DOCUMENT_T;
    ins_doc SODA_DOCUMENT_T;
BEGIN
    -- Open the collection
    collection := DBMS_SODA.open_collection('myCollectionName');
    document := SODA_DOCUMENT_T(
        b_content => utl_raw.cast_to_raw('{ "name" : "Alexander" }'));
    ins_doc := collection.insert_one_and_get(document);

    -- Insert the document and get its components
    IF ins_doc IS NOT NULL THEN
        DBMS_OUTPUT.put_line('Inserted document components:');
        DBMS_OUTPUT.put_line('Key: ' || ins_doc.get_key);
        DBMS_OUTPUT.put_line('Creation timestamp: ' || ins_doc.get_created_on);
        DBMS_OUTPUT.put_line('Last modified timestamp: '
            || ins_doc.get_last_modified);
    END IF;
END;
/

```



```

        DBMS_OUTPUT.put_line('Version: ' || ins_doc.get_version);
    END IF;
END;
/

```

Related Topics

- [Handling Transactions with SODA for PL/SQL](#)
As usual in PL/SQL and SQL, you can treat individual SODA read and write operations, or groups of them, as a transaction. To commit a transaction, use a SQL `COMMIT` statement. If you want to roll back changes, use a SQL `ROLLBACK` statement.
- [Dropping a Document Collection with SODA for PL/SQL](#)
You use PL/SQL function `DBMS_SODA.drop_collection` to drop a document collection.
- [Replacing Documents in a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method (member function) `replace_one()` or `replace_one_and_get()` to replace the content of a document in a collection, given its key. The latter method also gets the components of the result document (except for the updated content).

3.9 Finding Documents in Collections with SODA for PL/SQL

You can use `SODA_COLLECTION_T` method `find_one()` to find the unique document in a collection that has a given key.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_COLLECTION_T` method `find_one()`
- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_DOCUMENT_T` getter methods
- *Oracle Database SQL Language Reference* for information about SQL/JSON function `json_query`

Example 3-12 Finding the Unique Document That Has a Given Document Key

This example uses `SODA_COLLECTION_T` method `find_one()` to find the unique document whose key is "key1".

It then uses `SODA_DOCUMENT_T` methods `get_key()`, `get_blob()`, `get_created_on()`, `get_last_modified()`, and `get_version()`, to get the document metadata, which it passes to SQL/JSON function `json_query` to pretty-print (using keyword `PRETTY`).

```

DECLARE
    collection SODA_COLLECTION_T;
    document SODA_DOCUMENT_T;
BEGIN
    - Open the collection
    collection := DBMS_SODA.open_collection('myCollectionName');

```

```

- Find a document using a key
document := collection.find_one('key1');

IF document IS NOT NULL THEN
    DBMS_OUTPUT.put_line('Document components:');
    DBMS_OUTPUT.put_line('Key: ' || document.get_key);
    DBMS_OUTPUT.put_line('Content: ' ||
        utl_raw.cast_to_varchar2(document.get_blob));
    DBMS_OUTPUT.put_line('Creation timestamp: ' || document.get_created_on);
    DBMS_OUTPUT.put_line('Last modified timestamp: ' ||
        document.get_last_modified);
    DBMS_OUTPUT.put_line('Version: ' || document.get_version);
END IF;
END;
/

```

3.10 Replacing Documents in a Collection with SODA for PL/SQL

You can use `SODA_COLLECTION_T` method (member function) `replace_one()` or `replace_one_and_get()` to replace the content of a document in a collection, given its key. The latter method also gets the components of the result document (except for the updated content).

In addition to replacing the content, these methods update the values of the last-modified timestamp and the version. Replacement does *not* change the document key or the creation timestamp.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_COLLECTION_T` method `replace_one()`
- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_COLLECTION_T` method `replace_one_and_get()`
- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_DOCUMENT_T` getter methods

Example 3-13 Replacing a Document in a Collection, Given Its Key, and Getting the Result Document

This example replaces a document in a collection, given its key. It then gets (and prints) the key and the generated components from the result document. To obtain the components it uses `SODA_DOCUMENT_T` methods `get_key()`, `get_created_on()`, `get_last_modified()`, and `get_version()`.

```

DECLARE
    collection SODA_COLLECTION_T;
    document SODA_DOCUMENT_T;
    rep_doc SODA_DOCUMENT_T;
BEGIN
    collection := DBMS_SODA.open_collection('myCollectionName');

```

```

document := SODA_DOCUMENT_T(
    b_content => utl_raw.cast_to_raw('{"name" : "Sriky"}'));
rep_doc := collection.replace_one_and_get('k1', document);

IF rep_doc IS NOT NULL THEN
    DBMS_OUTPUT.put_line('Document components:');
    DBMS_OUTPUT.put_line('Key: ' || rep_doc.get_key);
    DBMS_OUTPUT.put_line('Creation timestamp: ' || rep_doc.get_created_on);
    DBMS_OUTPUT.put_line('Last modified timestamp: ' ||
        rep_doc.get_last_modified);
    DBMS_OUTPUT.put_line('Version: ' || rep_doc.get_version);
END IF;
END;
/

```

Related Topics

- [Handling Transactions with SODA for PL/SQL](#)
As usual in PL/SQL and SQL, you can treat individual SODA read and write operations, or groups of them, as a transaction. To commit a transaction, use a SQL `COMMIT` statement. If you want to roll back changes, use a SQL `ROLLBACK` statement.
- [Dropping a Document Collection with SODA for PL/SQL](#)
You use PL/SQL function `DBMS_SODA.drop_collection` to drop a document collection.
- [Inserting Documents into Collections with SODA for PL/SQL](#)
To insert a document into a collection, you invoke `SODA_COLLECTION_T` method (member function) `insert_one()` or `insert_one_and_get()`. These methods create document keys automatically, unless the collection is configured with client-assigned keys and the input document provides the key.

3.11 Removing Documents from a Collection with SODA for PL/SQL

You can use `SODA_COLLECTION_T` method `remove_one()` to remove a document from a collection, given its key.

See Also:

Oracle Database PL/SQL Packages and Types Reference for information about `SODA_COLLECTION_T` method `remove_one()`

Example 3-14 Removing a Document from a Collection Using a Document Key

This example removes the document whose document key is "key1". The removal status (1 if the document was removed; 0 if not) is returned and printed.

```

DECLARE
    collection SODA_COLLECTION_T;
    status     NUMBER;
BEGIN
    collection := DBMS_SODA.open_collection('myCollectionName');
    status := collection.remove_one('key1');

```

```

        DBMS_OUTPUT.put_line('Status: ' || status);
    END;
/

```

3.12 Handling Transactions with SODA for PL/SQL

As usual in PL/SQL and SQL, you can treat individual SODA read and write operations, or groups of them, as a transaction. To commit a transaction, use a SQL `COMMIT` statement. If you want to roll back changes, use a SQL `ROLLBACK` statement.

SODA operations `DBMS_SODA.create_collection` and `DBMS_SODA.drop_collection` do *not* automatically commit before or after they perform their action. This differs from the behavior of SQL DDL statements, which commit both before and after performing their action.

One consequence of this is that, before a SODA collection can be dropped, any outstanding write operations to it must be explicitly committed or rolled back — you must explicitly use SQL `COMMIT` or `ROLLBACK`. This is because `DBMS_SODA.drop_collection` does not itself issue commit before it performs its action. In this, the behavior of `DBMS_SODA.drop_collection` differs from that of a SQL `DROP TABLE` statement.

See Also:

- *Oracle Database SQL Language Reference* for information about the SQL `COMMIT` statement
- *Oracle Database SQL Language Reference* for information about the SQL `ROLLBACK` statement
- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_COLLECTION_T` method `insert_one()`

Example 3-15 Transaction Involving SODA Document Insertion and Replacement

This example shows the use of SQL `COMMIT` and `ROLLBACK` statements in an anonymous PL/SQL block. It opens a SODA collection, inserts a document, and then replaces its content. The combination of the document insertion and document content replacement operations is *atomic*: a single transaction.

```

DECLARE
    collection SODA_COLLECTION_T;
    status NUMBER;
BEGIN
    collection := DBMS_SODA.open_collection('myCollectionName');
    status := collection.insert_one(
        SODA_Document_T(
            b_content =>
utl_raw.cast_to_raw('{"a":"aval","b":"bval","c":"cval"}'));
    status := collection.replace_one('key1',
SODA_DOCUMENT_T('{"x":"xval","y":"yval"}'));
    -- Commit the transaction
    COMMIT;
    DBMS_OUTPUT.put_line('Transaction is committed');
    -- Catch exceptions and roll back if an error is raised

```

```
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.put_line (SQLERRM);
    ROLLBACK;
    DBMS_OUTPUT.put_line('Transaction has been rolled back');
END;
/
```

Related Topics

- [Dropping a Document Collection with SODA for PL/SQL](#)
You use PL/SQL function `DBMS_SODA.drop_collection` to drop a document collection.
- [Inserting Documents into Collections with SODA for PL/SQL](#)
To insert a document into a collection, you invoke `SODA_COLLECTION_T` method (member function) `insert_one()` or `insert_one_and_get()`. These methods create document keys automatically, unless the collection is configured with client-assigned keys and the input document provides the key.
- [Replacing Documents in a Collection with SODA for PL/SQL](#)
You can use `SODA_COLLECTION_T` method (member function) `replace_one()` or `replace_one_and_get()` to replace the content of a document in a collection, given its key. The latter method also gets the components of the result document (except for the updated content).

4

SODA Collection Configuration Using Custom Metadata

SODA collections are highly configurable. You can customize collection metadata, to obtain different behavior from that provided by default.

Note:

Although you can customize collection metadata to obtain different behavior from that provided by default, Oracle recommends *against* this unless you have a compelling reason. Customizing collection metadata requires familiarity with Oracle Database concepts, such as SQL data types. Because SODA collections are implemented on top of Oracle Database tables (or views), many collection configuration components are related to the underlying table configuration.

- [Getting the Metadata of an Existing Collection](#)
You use `SODA_COLLECTION_T` method `get_metadata()` to get all of the metadata for a collection, as a JSON document.
- [Creating a Collection That Has Custom Metadata](#)
To create a document collection that has custom metadata, you pass its metadata, as JSON data, to PL/SQL function `DBMS_SODA.create_collection`.

See Also:

- *Oracle Database Introduction to Simple Oracle Document Access (SODA)* for general information about SODA document collections and their metadata
- *Oracle Database Introduction to Simple Oracle Document Access (SODA)* for reference information about collection metadata components

4.1 Getting the Metadata of an Existing Collection

You use `SODA_COLLECTION_T` method `get_metadata()` to get all of the metadata for a collection, as a JSON document.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_COLLECTION_T` method `get_metadata()`
- *Oracle Database SQL Language Reference* for information about SQL/JSON function `json_query`

Example 4-1 Getting the Metadata of a Collection

This example shows the result of invoking `SODA_COLLECTION_T` method `get_metadata()` on the collection with the default configuration that was created using [Example 3-3](#). (It also uses SQL/JSON function `json_query`, with keyword `PRETTY`, to pretty-print the JSON data obtained.)

```
DECLARE
    collection SODA_COLLECTION_T;
BEGIN
    collection := DBMS_SODA.open_collection('myCollectionName');
    IF collection IS NULL THEN
        DBMS_OUTPUT.put_line('Collection does not exist');
    ELSE
        DBMS_OUTPUT.put_line('Metadata: '
            || json_query(collection.get_metadata, '$' PRETTY));
    END IF;
END;
/
```

Example 4-2 Default Collection Metadata

```
{
  "schemaName" : "mySchemaName",
  "tableName" : "myTableName",
  "keyColumn" :
  {
    "name" : "ID",
    "sqlType" : "VARCHAR2",
    "maxLength" : 255,
    "assignmentMethod" : "UUID"
  },
  "contentColumn" :
  {
    "name" : "JSON_DOCUMENT",
    "sqlType" : "BLOB",
    "compress" : "NONE",
    "cache" : true,
    "encrypt" : "NONE",
    "validation" : "STANDARD"
  },
  "versionColumn" :
```

```

    {
      "name" : "VERSION",
      "method" : "SHA256"
    },
    "lastModifiedColumn" :
    {
      "name" : "LAST_MODIFIED"
    },
    "creationTimeColumn" :
    {
      "name" : "CREATED_ON"
    },
    "readOnly" : false
  }

```

4.2 Creating a Collection That Has Custom Metadata

To create a document collection that has custom metadata, you pass its metadata, as JSON data, to PL/SQL function `DBMS_SODA.create_collection`.

The optional second argument to PL/SQL function `DBMS_SODA.create_collection` is a SODA **collection specification**. It is JSON data that specifies the metadata for the new collection.

If a collection with the same name already exists then it is simply opened and its handle is returned. If the custom metadata provided does not match the metadata of the existing collection then the collection is not opened and an error is raised. (To match, all metadata fields must have the same values.)

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* for information about PL/SQL function `DBMS_SODA.create_collection`
- *Oracle Database PL/SQL Packages and Types Reference* for information about `SODA_COLLECTION_T` method `get_metadata()`
- *Oracle Database SQL Language Reference* for information about SQL/JSON function `json_query`

Example 4-3 Creating a Collection That Has Custom Metadata

This example creates a collection with custom metadata that specifies two metadata columns, named `KEY` (for document keys), and `JSON` (for document content type JSON). The key assignment method is `CLIENT`, and the content-column SQL data type is `VARCHAR2`. The example uses `SODA_COLLECTION_T` method `get_metadata()` to get the complete metadata from the newly created collection, which it passes to SQL/JSON function `json_query` to pretty-print (using keyword `PRETTY`).

```

DECLARE
  collection SODA_COLLECTION_T;
  metadata VARCHAR2(4000) :=
    '{"keyColumn" : {"name" : "KEY", "assignmentMethod": "CLIENT" },
     "contentColumn" : { "name" : "JSON", "sqlType": "VARCHAR2" } }';
BEGIN
  collection := DBMS_SODA.create_collection('myCustomCollection', metadata);

```



```
DBMS_OUTPUT.put_line('Collection specification: ' ||  
                      json_query(collection.get_metadata, '$' PRETTY));  
END  
/
```

This is the pretty-printed output. The values of fields for `keyColumn` and `contentColumn` that are not specified in the collection specification are defaulted. The values of fields other than those provided in the collection specification (`keyColumn` and `contentColumn`) are also defaulted. The value of field `tableName` is defaulted from the collection name. The value of field `schemaName` is the database schema (user) that is current when the collection is created.

```
Collection specification: {  
  "schemaName" : "mySchemaName",  
  "tableName" : "myCustomCollection",  
  "keyColumn" :  
  {  
    "name" : "KEY",  
    "sqlType" : "VARCHAR2",  
    "maxLength" : 255,  
    "assignmentMethod" : "CLIENT"  
  },  
  "contentColumn" :  
  {  
    "name" : "JSON",  
    "sqlType" : "VARCHAR2",  
    "maxLength" : 4000,  
    "validation" : "STANDARD"  
  },  
  "readOnly" : false  
}
```

Related Topics

- [Creating a Document Collection with SODA for PL/SQL](#)
You can use PL/SQL function `DBMS_SODA.create_collection` to create a document collection with the default metadata.

Index

C

- collection configuration, [4-1](#)
- collection metadata
 - custom, [4-3](#)
 - getting, [4-2](#)
- collections
 - checking existence, [3-6](#)
 - creating, [3-4](#)
 - with custom metadata, [4-3](#)
 - discovering, [3-7](#)
 - dropping, [3-8](#)
 - opening, [3-6](#)
 - during creation, [3-4](#)
- committing a transaction, [3-18](#)
- components of SODA documents, [3-9](#)
- create_collection function
 - transaction handling, [3-18](#)
- creating collections, [3-4](#)
 - with custom metadata, [4-3](#)
- creating documents, [3-9](#)

D

- database role SODA_APP, [3-2](#)
- DBMS_SODA package subprograms
 - create_collection
 - example, [3-4](#)
 - transaction handling, [3-18](#)
 - drop_collection
 - example, [3-8](#)
 - transaction handling, [3-18](#)
 - list_collection_names
 - example, [3-7](#)
 - open_collection
 - example, [3-6](#)
- DBMS_SODA.DOC_BLOB constant, [3-9](#)
- DBMS_SODA.DOC_CLOB constant, [3-9](#)
- DBMS_SODA.DOC_VARCHAR2 constant, [3-9](#)
- deleting collections
 - See dropping collections
- deleting documents from collections
 - See removing documents from collections
- discovering collections
 - checking existence, [3-6](#)

- discovering collections (*continued*)
 - listing, [3-7](#)
- documents
 - components, [3-9](#)
 - creating, [3-9](#)
 - finding in collections, [3-15](#)
 - inserting into collections, [3-13](#)
 - metadata, [3-9](#)
 - removing from collections, [3-17](#)
 - replacing in collections, [3-16](#)
- drop_collection function
 - example, [3-8](#)
 - transaction handling, [3-18](#)
- dropping collections, [3-8](#)

E

- existing collection, checking for, [3-6](#)

F

- find_one() SODA_COLLECTION_T method, [3-15](#)
- finding documents in collections, [3-15](#)

G

- get_blob() SODA_DOCUMENT_T method, [3-9](#)
- get_clob() SODA_DOCUMENT_T method, [3-9](#)
- get_created_on() SODA_DOCUMENT_T method, [3-9](#)
- get_data_type() SODA_DOCUMENT_T method, [3-9](#)
- get_key() SODA_DOCUMENT_T method, [3-9](#)
- get_last_modified() SODA_DOCUMENT_T method, [3-9](#)
- get_media_type() SODA_DOCUMENT_T method, [3-9](#)
- get_metadata() SODA_COLLECTION_T method, [4-2](#)
- get_varchar2() SODA_DOCUMENT_T method, [3-9](#)
- get_version() SODA_DOCUMENT_T method, [3-9](#)
- getter methods, document, [3-9](#)
- getting collection metadata, [4-2](#)

getting document components, [3-9](#)

H

handling transactions, [3-18](#)

I

insert_one_and_get() SODA_COLLECTION_T method, [3-13](#)

insert_one() SODA_COLLECTION_T method, [3-13](#)

inserting documents into collections, [3-13](#)

L

list_collection_names function
example, [3-7](#)

listing collections, [3-7](#)

M

metadata of collections
getting, [4-2](#)

metadata of documents
getting, [3-9](#)

O

open_collection function
example, [3-6](#)

opening collections, [3-6](#)
during creation, [3-4](#)

P

prerequisites for using SODA for PL/SQL, [1-1](#)

R

remove_one() SODA_COLLECTION_T method, [3-17](#)

removing documents from collections, [3-17](#)

replace_one_and_get() SODA_COLLECTION_T method, [3-16](#)

replace_one() SODA_COLLECTION_T method, [3-16](#)

replacing documents in collections, [3-16](#)

role SODA_APP, [3-2](#)

rolling back a transaction, [3-18](#)

S

SODA_APP database role, [3-2](#)

SODA_COLLECTION_T methods

find_one(), [3-15](#)

get_metadata(), [4-2](#)

insert_one_and_get(), [3-13](#)

insert_one(), [3-13](#)

remove_one(), [3-17](#)

replace_one_and_get(), [3-16](#)

replace_one(), [3-16](#)

SODA_DOCUMENT_T methods

get_blob(), [3-9](#)

get_clob(), [3-9](#)

get_created_on(), [3-9](#)

get_data_type(), [3-9](#)

get_key(), [3-9](#)

get_last_modified(), [3-9](#)

get_media_type(), [3-9](#)

get_varchar2(), [3-9](#)

get_version(), [3-9](#)

SODA_DOCUMENT_T object type and
constructors, [3-9](#)

T

transaction handling, [3-18](#)