

Contents

1 Changes in This Release for JavaScript Developer's Guide

July 2024, Release Update 23.5	1-1
January 2025, Release Update 23.7	1-1
April 2025, Release Update 23.8	1-2

2 Introduction to Oracle Database Multilingual Engine for JavaScript

The Need for a Multilingual Engine	2-2
Overview of JavaScript	2-2
Overview of Multilingual Engine for JavaScript	2-3
JavaScript Implementation Details	2-4
Invoking JavaScript in the Database	2-5
Introduction to Dynamic Execution	2-5
Introduction to MLE Module Calls	2-6
About MLE Execution Contexts	2-8
About Restricted Execution Contexts	2-9
Introduction to Debugging JavaScript Code	2-10

3 MLE JavaScript Modules and Environments

Using JavaScript Modules in MLE	3-1
Managing JavaScript Modules in the Database	3-3
Naming JavaScript Modules	3-3
Creating JavaScript Modules in the Database	3-4
Storing JavaScript Code in Databases Using Single-Byte Character Sets	3-5
Code Analysis	3-5
Preparing JavaScript code for MLE Module Calls	3-6
Additional Options for Providing JavaScript Code to MLE	3-8
Specifying Module Version Information and Providing JSON Metadata	3-9
Drop JavaScript Modules	3-10
Alter JavaScript Modules	3-10
Overview of Built-in JavaScript Modules	3-11
Dictionary Views Related to MLE JavaScript Modules	3-11
USER_SOURCE	3-12

USER_MLE_MODULES	3-13
Specifying Environments for MLE Modules	3-13
Creating MLE Environments in the Database	3-14
Naming MLE Environments	3-14
Creating an Empty MLE Environment	3-15
Creating an Environment as a Clone of an Existing Environment	3-16
Using MLE Environments for Import Resolution	3-16
Providing Language Options	3-18
Dropping MLE Environments	3-19
Modifying MLE Environments	3-19
Altering Language Options	3-20
Modifying Module Imports	3-20
Dictionary Views Related to MLE JavaScript Environments	3-20
USER_MLE_ENVS	3-21
USER_MLE_ENV_IMPORTS	3-21

4 Overview of Dynamic MLE Execution

About Dynamic JavaScript Execution	4-1
Dynamic Execution Workflow	4-2
Providing JavaScript Code Inline	4-2
Loading JavaScript Code from Files	4-3
Returning the Result of the Last Execution	4-6

5 Overview of Importing MLE JavaScript Modules

JavaScript Module Hierarchies	5-2
Resolving Import Names Using MLE Environments	5-2
Export Functionality	5-3
Named Exports	5-3
Default Exports	5-4
Private Identifiers	5-5
Import Functionality	5-5
Module Objects	5-5
Named Imports	5-6
Default Imports	5-7

6 MLE JavaScript Functions

Call Specifications for Functions	6-1
Creating a Call Specification for an MLE Module	6-1
Components of an MLE Call Specification	6-4

MLE Module Clause	6-5
ENV Clause	6-5
SIGNATURE Clause	6-5
Creating an Inline MLE Call Specification	6-7
Components of an Inline MLE Call Specification	6-10
Accessing Built-in Modules Using JavaScript Global Variables	6-11
Choosing Inline Versus Module MLE Call Specifications	6-12
Runtime Isolation for an MLE Call Specification	6-12
Dictionary Views for Call Specifications	6-15
OUT and IN OUT Parameters	6-16

7 Calling PL/SQL and SQL from the MLE JavaScript SQL Driver

Introduction to the MLE JavaScript SQL Driver	7-1
Working with the MLE JavaScript Driver	7-2
Connection Management in the MLE JavaScript Driver	7-3
Introduction to Executing SQL Statements	7-3
Processing Comparison Between node-oracledb and mle-js-oracledb	7-6
Selecting Data Using the MLE JavaScript Driver	7-6
Direct Fetch: Arrays	7-7
Direct Fetch: Objects	7-8
Fetching Rows as ResultSets: Arrays	7-9
Fetching Rows as ResultSets: Iterating Over ResultSet Objects	7-10
Data Modification	7-11
Bind Variables	7-11
Using Bind-by-Name vs Bind-by-Position	7-12
Named Bind Variables	7-12
Positional Bind Variables	7-14
RETURNING INTO Clause	7-15
Batch Operations	7-16
PL/SQL Invocation from the MLE JavaScript SQL Driver	7-18
Error Handling in SQL Statements	7-20
Working with JSON Data	7-25
Using Large Objects (LOB) with MLE	7-30
Writing LOBs	7-30
Reading LOBs	7-31
API Differences Between node-oracledb and mle-js-oracledb	7-32
Synchronous API and Error Handling	7-32
Connection Handling	7-33
Transaction Management	7-34
Type Mapping	7-34
Unsupported Data Types	7-37

Miscellaneous Features Not Available with the MLE JavaScript SQL Driver	7-37
Introduction to the PL/SQL Foreign Function Interface	7-38
Object Resolution Using FFI	7-39
Provide Arguments to a Subprogram Using FFI	7-42

8 Working with SODA Collections in MLE JavaScript Code

High-Level Introduction to Working with SODA for In-Database JavaScript	8-2
SODA Objects	8-3
Using SODA for In-Database JavaScript	8-4
Getting Started with SODA for In-Database JavaScript	8-6
Creating a Document Collection with SODA for In-Database JavaScript	8-8
Opening an Existing Document Collection with SODA for In-Database JavaScript	8-9
Checking Whether a Given Collection Exists with SODA for In-Database JavaScript	8-9
Discovering Existing Collections with SODA for In-Database JavaScript	8-10
Dropping a Document Collection with SODA for In-Database JavaScript	8-11
Creating Documents with SODA for In-Database JavaScript	8-12
Inserting Documents into Collections with SODA for In-Database JavaScript	8-14
Saving Documents into Collections with SODA for In-Database JavaScript	8-15
SODA for In-Database JavaScript Read and Write Operations	8-15
Finding Documents in Collections with SODA for In-Database JavaScript	8-17
Replacing Documents in a Collection with SODA for In-Database JavaScript	8-22
Removing Documents from a Collection with SODA for In-Database JavaScript	8-24
Indexing the Documents in a Collection with SODA for In-Database JavaScript	8-25
Getting a Data Guide for a Collection with SODA for In-Database JavaScript	8-27
Handling Transactions with SODA for In-Database JavaScript	8-29
Creating Call Specifications Involving the SODA API	8-29

9 Post-Execution Debugging of MLE JavaScript Modules

Specifying Debugpoints	9-2
Debugpoint Locations	9-2
Debugpoint Actions	9-2
Debugpoint Conditions	9-4
Managing Debugpoints	9-4
Debugging Security Considerations	9-6
COLLECT DEBUG INFO Privilege for MLE Modules	9-6
Analyzing Debug Output	9-7
Textual Representation of Debug Output	9-7
Analyzing Debug Output Using Developer Tools	9-10
Error Handling in MLE	9-10
Errors in Callouts	9-13

Accessing stdout and stderr from JavaScript	9-13
Accessing stdout and stderr for MLE Modules	9-13
Accessing stdout and stderr for Dynamic MLE	9-15

10 MLE Security

System and Object Privileges Required for Working with JavaScript in MLE	10-1
Necessary Privileges for the Execution of JavaScript Code	10-2
Necessary Privileges for Using the NoSQL API	10-2
Necessary Privileges for Creating MLE Schema Objects	10-3
Necessary Privileges for Creating MLE Modules and Environments in ANY Schema	10-3
Necessary Privileges for Post-Execution Debugging	10-4
Security Considerations for MLE	10-4
MLE_PROG_LANGUAGES Initialization Parameter	10-5
Execution Contexts	10-5
Runtime State Isolation	10-6
Database Security Model	10-8
Considerations for Using MLE Call Specifications and Modules from Different Schemas	10-9
Auditing MLE Operations in Oracle Database	10-10
JavaScript Security Best Practices	10-10
Using Bind Variables for Security and Performance	10-10
Generic Database and PL/SQL Specific Security Considerations	10-12
Supply Chain Security	10-13
Software Bill of Material	10-14
Using the Database to Store State	10-14
Disabling Multilingual Runtime	10-16
MLE Security Examples	10-16
Business Logic Stored in MLE Modules	10-16
Generic Data Processing Libraries	10-18
Generic Libraries in Business Logic	10-19

A MLE Type Conversions

MLE JavaScript Support for JSON	A-4
MLE JavaScript Support for the VECTOR Data Type	A-6

Index
