133 DBMS_MLE

The `DBMS_MLE` package allows users to execute JavaScript code inside the Oracle Database and exchange data seamlessly between PL/SQL and JavaScript. The JavaScript code itself can execute PL/SQL and SQL through built-in JavaScript modules. JavaScript data types are automatically mapped to Oracle Database data types and vice versa.

With the `DBMS_MLE` package, developers can write their data processing logic in JavaScript.

This chapter contains the following topics:

DBMS_MLE Overview

Runtime state for MLE execution is encapsulated in execution contexts that users can explicitly create and destroy. Users can also export values from PL/SQL to MLE, and import values from MLE back into PL/SQL.

The following example captures the typical workflow for MLE execution:

```
set serveroutput on;
declare
  ctx dbms_mle.context_handle_t;
  source clob;
  greeting varchar2(100);
begin
  ctx := dbms_mle.create_context(); -- Create execution context for MLE execution
  dbms_mle.export_to_mle(ctx, 'person', 'World'); -- Export value from PL/SQL
  source := q'~
    var bindings = require("mle-js-bindings");
    var person = bindings.importValue("person"); // Import value previously exported
from PL/SQL
    var greeting = "Hello, " + person + "!";
    bindings.exportValue("greeting", greeting); // Export value to PL/SQL
  ~';
  dbms_mle.eval(ctx, 'JAVASCRIPT', source); -- Evaluate the source code snippet in the
execution context
  dbms_mle.import_from_mle(ctx, 'greeting', greeting); -- Import value previously
exported from MLE
  dbms_output.put_line('Greetings from MLE: ' || greeting);
  dbms_mle.drop_context(ctx); -- Drop the execution context once no longer required
end;
/
```

Executing the above code block produces the following output:

```
Greetings from MLE: Hello, World!
```

DBMS_MLE Security Model

Access to MLE features is protected by database privileges. The user must have the `EXECUTE DYNAMIC MLE` privilege while calling any of its functions and procedures that pertain to MLE execution. In addition to this, the user must also have the `EXECUTE ON JAVASCRIPT` privilege to execute JavaScript code. An `ORA-01031` error is raised if the user calling any of the `DBMS_MLE` subprograms does not have the appropriate privileges. See the summary of `DBMS_MLE` subprograms for the privileges required to call each `DBMS_MLE` subprogram.

Summary of DBMS_MLE Subprograms

This table lists the `DBMS_MLE` subprograms and briefly describes them.

Table 133-1 DBMS_MLE Package Subprograms

Subprogram	Description
CREATE_CONTEXT Function	Creates an MLE context for executing snippets in JavaScript.
DISABLE_CTX_STDERR Procedure	This procedure disables stderr stream of the specified MLE context, so that future writes to stderr are discarded.
DISABLE_CTX_STDOUT Procedure	This procedure disables stdout stream of the specified MLE context, so that future writes to `stdout` are discarded.
DISABLE_DEBUGGING Procedure	This procedure disables any currently enabled debugpoints for the current session.
DISABLE_ICS_STDERR Procedure	This procedure disables the `stderr` stream of the inlined MLE call specification context, so that future writes to `stderr` are discarded for the calling user in the current session.
DISABLE_ICS_STDOUT Procedure	This procedure disables the `stdout` stream of an inlined call specification context, so that future writes to `stdout` are discarded for the calling user in the current session.
DISABLE_STDERR Procedure	This procedure disables the stderr stream of MLE contexts, so that future writes to `stderr` are discarded.
DISABLE_STDOUT Procedure	This procedure disables the stdout stream of MLE contexts, so that future writes to stdout are discarded.
DROP_CONTEXT Procedure	This procedure is used to drop an MLE context that was previously created using the `CREATE_CONTEXT` procedure. After the context is dropped, the context handle is no longer valid and cannot be used anymore.
ENABLE_DEBUGGING Procedure	This procedure enables a set of debugpoints for the current session.
EVAL Procedure	This procedure executes the given JavaScript code within the context identified by the context handle.
EXPORT_TO_MLE Procedure	This procedure allows you to assign the given value, with appropriate conversion, to the named property in the MLE context. The property is created if it is not already present.
GET_AVAILABLE_LANGUAGES Function	This function returns the set of available MLE languages.
GET_CTX_ERROR_STACK Function	This function retrieves the JavaScript stack trace for the most recent application error in the given execution context.
GET_ERROR_STACK Function	This function retrieves the JavaScript stack trace for the most recent application error in the given module (and optional environment) call.

Table 133-1 (Cont.) DBMS_MLE Package Subprograms

Subprogram	Description
IMPORT_FROM_MLE Procedure	This procedure retrieves the value of the named property from the MLE context and converts it to the requested PL/SQL type.
PARSE_DEBUG_OUTPUT Function	Given a BLOB containing MLE debug output in the Java Heap Dump format, this function returns a textual representation of debug output.
SET_CTX_STDERR Procedure	This procedure redirects the stderr stream of the MLE context to the given `CLOB`.
SET_CTX_STDERR_TO_DBMS_OUTPUT Procedure	This procedure redirects the stderr stream of the MLE context to `DBMS_OUTPUT`.
SET_CTX_STDOUT Procedure	This procedure redirects the stdout stream of the MLE context to the given `CLOB`.
SET_CTX_STDOUT_TO_DBMS_OUTPUT Procedure	This procedure redirects the stdout stream of the MLE context to `DBMS_OUTPUT`.
SET_ICS_STDERR Procedure	This procedure redirects the `stderr` stream of the inlined MLE call specification context to the given `CLOB` for the calling user in the current session.
SET_ICS_STDERR_TO_DBMS_OUTPUT Procedure	This procedure redirects the `stderr` stream of the inlined MLE call specification context to `DBMS_OUTPUT` for the calling user in the current session.
SET_ICS_STDOUT Procedure	This procedure redirects the `stdout` stream of an inlined MLE call specification context in the current session to the given `CLOB`.
SET_ICS_STDOUT_TO_DBMS_OUTPUT Procedure	This procedure redirects the `stdout` stream of the inlined call specification context to `DBMS_OUTPUT` for the calling user in the current session.
SET_STDERR Procedure	This procedure redirects the stderr stream of MLE contexts to the given `CLOB`.
SET_STDERR_TO_DBMS_OUTPUT Procedure	This procedure redirects the stderr stream of MLE contexts to `DBMS_OUTPUT`.
SET_STDOUT Procedure	This procedure redirects the stdout stream of MLE contexts to the given `CLOB`.
SET_STDOUT_TO_DBMS_OUTPUT Procedure	This procedure redirects the stdout stream of MLE contexts to `DBMS_OUTPUT`.

CREATE_CONTEXT Function

Creates an MLE context for executing snippets in JavaScript. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.CREATE_CONTEXT
   RETURN CONTEXT_HANDLE_T;

DBMS_MLE.CREATE_CONTEXT(
   environment      IN      VARCHAR2)
RETURN CONTEXT_HANDLE_T;
```

Parameters

Table 133-2   CREATE_CONTEXT Function Parameters

Parameter	Description
`environment`	The case-sensitive schema name of the MLE environment that configures the context. Optional.

Return Values

This function returns a handle that uniquely identifies a context within a session for use in subsequent operations, such as `EXPORT_TO_MLE` and `EVAL`.

Usage Notes

A context has a lifetime limited to the session in which it was created. When a client session is terminated, all its contexts are dropped. All MLE contexts created in a session are also dropped when the session state is reset, for example, by calling `DBMS_SESSION.RESET_PACKAGE`. JavaScript code is evaluated in the context using the user, roles, and schema that are in effect at the time of context creation.

The function may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04105`: if the environment does not exist.

See Also:

Oracle Database JavaScript Developer's Guide for more details about privileges required to execute JavaScript code

DISABLE_CTX_STDERR Procedure

This procedure disables `stderr` stream of the specified MLE context, so that future writes to `stderr` are discarded. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.DISABLE_CTX_STDERR(
      context_handle            IN            context_handle_t);
```

Parameters

Table 133-3   DISABLE_CTX_STDERR Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges

DISABLE_CTX_STDOUT Procedure

This procedure disables `stdout` stream of the specified MLE context, so that future writes to `stdout` are discarded. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.DISABLE_CTX_STDOUT(
      context_handle            IN            context_handle_t);
```

Parameters

Table 133-4 DISABLE_CTX_STDOUT Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges

DISABLE_DEBUGGING Procedure

This procedure disables any currently enabled debugpoints for the current session. Post-execution debugging allows you to collect runtime state to be used for analysis after the program has been run. Post-execution debugging can only be applied to MLE code deployed in modules as opposed to code deployed using dynamic execution.

Syntax

```
DBMS_MLE.DISABLE_DEBUGGING();
```

Usage Notes

This procedure has no effect if no debugpoints are currently enabled. Debugging can be enabled again with a subsequent call to `DBMS_MLE.ENABLE_DEBUGGING`.

See Also:

Oracle Database JavaScript Developer's Guide for more information about post-execution debugging with MLE

DISABLE_ICS_STDERR Procedure

This procedure disables the `stderr` stream of the inlined MLE call specification context, so that future writes to `stderr` are discarded for the calling user in the current session. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.DISABLE_ICS_STDERR(
      name      IN      VARCHAR2);
```

Parameters

Table 133-5 DISABLE_ICS_STDERR Procedure Parameters

Parameter	Description
`name`	The name of the inlined MLE call specification.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04043`: if the call specification does not exist.

DISABLE_ICS_STDOUT Procedure

This procedure disables the `stdout` stream of an inlined call specification context, so that future writes to `stdout` are discarded for the calling user in the current session. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.DISABLE_ICS_STDOUT(
      name      IN      VARCHAR2);
```

Parameters

Table 133-6 DISABLE_ICS_STDOUT Procedure Parameters

Parameter	Description
`name`	The name of the inlined MLE call specification.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.

- `ORA-04043`: if the call specification does not exist.

DISABLE_STDERR Procedure

This procedure disables the `stderr` stream of MLE contexts, so that future writes to `stderr` are discarded. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.DISABLE_STDERR();

DBMS_MLE.DISABLE_STDERR (
      module_name      IN      VARCHAR2);

DBMS_MLE.DISABLE_STDERR (
      module_name      IN      VARCHAR2,
      env_name         IN      VARCHAR2);
```

Parameters

Table 133-7 DISABLE_STDERR Function Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment.

Usage Notes

When called without parameters, this procedure applies to all existing contexts and contexts created in the future. Otherwise, only the context associated with the given module (or module and environment combination) is affected.

If no environment is specified, the context defined by the given module and the built-in environment is used.

Any output that was buffered so far gets flushed to the pre-existing sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04103`: if the module does not exist
- `ORA-04105`: if the environment does not exist

DISABLE_STDOUT Procedure

This procedure disables the stdout stream of MLE contexts, so that future writes to `stdout` are discarded. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.DISABLE_STDOUT();

DBMS_MLE.DISABLE_STDOUT(
      module_name       IN      VARCHAR2);

DBMS_MLE.DISABLE_STDOUT(
      module_name       IN      VARCHAR2,
      env_name          IN      VARCHAR2);
```

Parameters

Table 133-8 DISABLE_STDOUT Function Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment.

Usage Notes

When called without parameters, this procedure applies to all existing contexts and contexts created in the future. Otherwise, only the context associated with the given module (or module and environment combination) is affected.

If no environment is specified, the context defined by the given module and the built-in environment is used.

Any output that was buffered so far gets flushed to the pre-existing sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04103`: if the module does not exist
- `ORA-04105`: if the environment does not exist

DROP_CONTEXT Procedure

This procedure is used to drop an MLE context that was previously created using the `CREATE_CONTEXT` function. After the context is dropped, the context handle is no longer valid and cannot be used anymore. You need the `EXECUTE DYNAMIC MLE` privilege to execute this procedure.

Syntax

```
DBMS_MLE.DROP_CONTEXT(
   context_handle   IN      context_handle_t);
```

Parameters

Table 133-9 DROP_CONTEXT Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.

Usage Notes

The procedure may raise `ORA-01031` error if the caller does not have sufficient privileges.

ENABLE_DEBUGGING Procedure

This procedure enables a set of debugpoints for the current session. Post-execution debugging allows you to collect runtime state to be used for analysis after the program has been run. Post-execution debugging can only be applied to MLE code deployed in modules as opposed to code deployed using dynamic execution.

Syntax

```
DBMS_MLE.ENABLE_DEBUGGING(
   debugspec          IN         JSON,
   sink               OUT        NOCOPY BLOB);
```

Parameters

Table 133-10   ENABLE_DEBUGGING Procedure Parameters

Parameter	Description
`debugspec`	The debug specification as a JSON document that identifies the debugging information to be collected.
`sink`	The sink to log debug output to.

Usage Notes

Calling this procedure multiple times in the same session replaces any existing set of debugpoints.

All enabled debugpoints are automatically disabled once the session ends.

The procedure may raise the following errors:

- `ORA-04103`: if the module does not exist.
- `ORA-04162`: if an attempt is made to debug an MLE built-in module.
- `ORA-04164`: if the caller is missing the `COLLECT DEBUG INFO` privilege on the module.
- `ORA-04165`: if the provided debug specification is invalid.

See Also:

Oracle Database JavaScript Developer's Guide for more information about post-execution debugging with MLE

EVAL Procedure

This procedure executes the given JavaScript code within the context identified by the context handle.

The evaluated code has access to all previous modifications to the state of the context, including variables defined by code previously evaluated in the context and values exported through `EXPORT_TO_MLE()`. The evaluated code can also import MLE built-in modules such as the MLE SQL driver.

You need the `EXECUTE DYNAMIC MLE` privilege to execute this procedure. It also requires the `EXECUTE ON JAVASCRIPT` privilege.

Syntax

```
DBMS_MLE.EVAL(
   context_handle   IN         context_handle_t,
   language_id      IN         language_t,
   source           IN         CLOB,
   result           IN OUT     NOCOPY CLOB CHARACTER SET ANY_CS,
   options          IN         VARCHAR2          DEFAULT NULL,
   source_name      IN         VARCHAR2          DEFAULT NULL);

DBMS_MLE.EVAL(
   context_handle   IN         context_handle_t,
   language_id      IN         language_t,
   source           IN         VARCHAR2,
   result           IN OUT     NOCOPY CLOB CHARACTER SET ANY_CS,
   options          IN         VARCHAR2          DEFAULT NULL,
   source_name      IN         VARCHAR2          DEFAULT NULL);
```

Parameters

Table 133-11   EVAL Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context.
`language_id`	The language of the provided source code. The value `'JAVASCRIPT'` must be provided as of Oracle 23ai.
`source`	The source code to be executed.
`result`	A buffer to which the result of the evaluation of the source code is appended. Optional.
`options`	Reserved for future use. Optional.
`source_name`	A name for the provided source code that is used to identify the snippet in stack traces. Optional.

Usage Notes

When specifying the optional `source_name` parameter, the `options` parameter must be defined as either `NULL` or `""`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04108`: if the current container, the current user, or the currently enabled roles are different from those in effect at the time of context creation.
- `ORA-04153`: if the context handle is invalid.
- `ORA-04161`: if the source code being evaluated throws an exception.

EXPORT_TO_MLE Procedure

This procedure allows you to assign the given value, with appropriate conversion, to the named property in the MLE context. The property is created if it is not already present. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      BINARY_INTEGER);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      BINARY_DOUBLE);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      BINARY_FLOAT);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      BLOB);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      BOOLEAN);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      CLOB CHARACTER SET ANY_CS);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      DATE);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      DSINTERVAL_UNCONSTRAINED);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      JSON);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      NUMBER);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      TIMESTAMP_TZ_UNCONSTRAINED);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      TIMESTAMP_UNCONSTRAINED);

DBMS_MLE.EXPORT_UROWID (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      UROWID);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      VARCHAR2 CHARACTER SET ANY_CS);

DBMS_MLE.EXPORT_TO_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      YMINTERVAL_UNCONSTRAINED);

DBMS_MLE.EXPORT_CHAR (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      CHAR CHARACTER SET ANY_CS);

DBMS_MLE.EXPORT_RAW (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   property_value   IN      RAW);
```

Parameters

Table 133-12 EXPORT_TO_MLE Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.
`property_name`	The name of the variable to be set. If the `property_name` value is `NULL` or an empty string, `ORA-04157` error is thrown.
`property_value`	The value to which the variable should be set.

Usage Notes

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.

- `ORA-04157`: if the value of the passed property_name is `NULL` or an empty string.
- `ORA-04108`: if the current container, the current user, or the currently enabled roles are different from those in effect at the time of context creation.
- `ORA-04153`: if the context handle is invalid.

GET_AVAILABLE_LANGUAGES Function

This function returns the set of available MLE languages.

Syntax

```
DBMS_MLE.GET_AVAILABLE_LANGUAGES()
  RETURN languages_t;
```

Return Values

A set of available MLE languages as a table of language identifiers as they can be used as an argument to `DBMS_MLE.EVAL()`.

GET_CTX_ERROR_STACK Function

This function retrieves the JavaScript stack trace for the most recent application error in the given execution context. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.GET_CTX_ERROR_STACK(
   context_handle           IN            context_handle_t)
RETURN error_frames_t;
```

Parameters

Table 133-13 GET_CTX_ERROR_STACK Function Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.

Return Values

A collection of error stack frames, each of type error_frame_t. An empty collection is returned if there is no error stack to report.

Usage Notes

The Function may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04153`: if the context handle is invalid.

GET_ERROR_STACK Function

This function retrieves the JavaScript stack trace for the most recent application error in the given module (and optional environment) call. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.GET_ERROR_STACK(
   module_name            IN            VARCHAR2,
   env_name               IN            VARCHAR2 DEFAULT '')
RETURN error_frames_t;
```

Parameters

Table 133-14 GET_ERROR_STACK Function Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment. Optional.

Return Values

A collection of error stack frames, each of type error_frame_t. An empty collection is returned if there is no error stack to report.

Usage Notes

The Function may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04170`: if the module name or environment name is invalid

IMPORT_FROM_MLE Procedure

This procedure retrieves the value of the named property from the MLE context and converts it to the requested PL/SQL type. You need the `EXECUTE DYNAMIC MLE` privilege to execute this procedure.

Syntax

```
DBMS_MLE.IMPORT_FROM_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   target           OUT     NOCOPY BINARY_INTEGER);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   target           OUT     NOCOPY BINARY_DOUBLE);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   target            OUT     NOCOPY BINARY_FLOAT);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY BLOB);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY BOOLEAN);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY CLOB CHARACTER SET ANY_CS);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY DATE);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY DSINTERVAL_UNCONSTRAINED);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     JSON);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY NUMBER);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY TIMESTAMP_TZ_UNCONSTRAINED);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY TIMESTAMP_UNCONSTRAINED);

DBMS_MLE.IMPORT_UROWID (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY UROWID);

DBMS_MLE.IMPORT_FROM_MLE (
   context_handle    IN      context_handle_t,
   property_name     IN      VARCHAR2,
   target            OUT     NOCOPY VARCHAR2 CHARACTER SET ANY_CS);
```

```
DBMS_MLE.IMPORT_FROM_MLE (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   target           OUT     NOCOPY YMINTERVAL_UNCONSTRAINED);

DBMS_MLE.IMPORT_CHAR (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   target           OUT     CHAR CHARACTER SET ANY_CS);

DBMS_MLE.IMPORT_RAW (
   context_handle   IN      context_handle_t,
   property_name    IN      VARCHAR2,
   target           OUT     RAW);
```

Parameters

Table 133-15 IMPORT_FROM_MLE Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.
`property_name`	The name of the property to be retrieved. If the `property_name` is `NULL` or an empty string, `ORA-04157` error is thrown.
`target`	A PL/SQL variable into which the retrieved property is stored.

Usage Notes

The `IMPORT_FROM_MLE` procedure may throw the following exceptions:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04108`: if the current container, the current user, or the currently enabled roles are different from those in effect at the time of context creation.
- `ORA-04153`: if the context handle is invalid.
- `ORA-04157`: if the value of the passed property_name is `NULL` or an empty string.
- `ORA-04205`: the value cannot be converted to the target PL/SQL type.
- `ORA-06502`: the buffer of the PL/SQL variable is too small to hold the retrieved value.

PARSE_DEBUG_OUTPUT Function

Given a BLOB containing MLE debug output in the Java Heap Dump format, returns a textual representation of debug output.

Syntax

```
DBMS_MLE.PARSE_DEBUG_OUTPUT(
   debugoutput         IN         BLOB)
RETURN JSON;
```

Parameters

Table 133-16 PARSE_DEBUG_OUTPUT Function Parameters

Parameter	Description
`debugoutput`	MLE debug output in the Java Heap Dump format.

Return Values

The function returns a JSON representation of the debug information. The output is an array of `DebugPointData` objects.

Usage Notes

The procedure may raise the following errors:

- `ORA-04163`: if the input is not in the Java Heap Dump format.
- `ORA-04166`: if the debug output is invalid.

See Also:

Oracle Database JavaScript Developer's Guide for more information about analyzing debug output

SET_CTX_STDERR Procedure

This procedure redirects the `stderr` stream of the MLE context to the given `CLOB`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_CTX_STDERR(
      context_handle  IN            context_handle_t,
      sink            IN OUT        NOCOPY CLOB CHARACTER SET ANY_CS);
```

Parameters

Table 133-17 SET_CTX_STDERR Procedure Parameters

Parameter	Description
`context_handle`	Handle to an MLE context in the current session.
`sink`	The CLOB sink to redirect stderr to. Providing a `NULL` value will result in `ORA-06530` error.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to the new sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04153`: if the context handle is invalid.
- `ORA-06530`: if the sink is `NULL`.

SET_CTX_STDERR_TO_DBMS_OUTPUT Procedure

This procedure redirects the `stderr` stream of the MLE context to `DBMS_OUTPUT`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_CTX_STDERR_TO_DBMS_OUTPUT(
   context_handle      IN      context_handle_t);
```

Parameters

Table 133-18 SET_CTX_STDERR_TO_DBMS_OUTPUT Procedure Parameters

Parameter	Description
`context_handle`	Handle to an MLE context in the current session.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to `DBMS_OUTPUT`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04153`: if the context handle is invalid.

SET_CTX_STDOUT Procedure

This procedure redirects the `stdout` stream of the MLE context to the given CLOB. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_CTX_STDOUT(
   context_handle      IN context_handle_t,
   sink                IN OUT NOCOPY CLOB CHARACTER SET ANY_CS);
```

Parameters

Table 133-19 SET_CTX_STDOUT Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.
`sink`	The `CLOB` sink to redirect `stdout` to. Providing a `NULL` value will result in an `ORA-06530` error.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to the new sink.

This procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04153`: if the context handle is invalid.
- `ORA-06530`: if the sink is `NULL`.

SET_CTX_STDOUT_TO_DBMS_OUTPUT Procedure

This procedure redirects the `stdout` stream of the MLE context to `DBMS_OUTPUT`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_CTX_STDOUT_TO_DBMS_OUTPUT(
   context_handle      IN      context_handle_t);
```

Parameters

Table 133-20   SET_CTX_STDOUT_TO_DBMS_OUTPUT Procedure Parameters

Parameter	Description
`context_handle`	The handle to an MLE context in the current session.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to `DBMS_OUTPUT`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04153`: if the context handle is invalid.

SET_ICS_STDERR Procedure

This procedure redirects the `stderr` stream of the inlined MLE call specification context to the given `CLOB` for the calling user in the current session. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_ICS_STDERR(
      name            IN            VARCHAR2,
      sink            IN OUT        NOCOPY CLOB CHARACTER SET ANY_CS);
```

Parameters

Table 133-21 SET_ICS_STDERR Procedure Parameters

Parameter	Description
`name`	The name of the inlined MLE call specification.
`sink`	The `CLOB` to redirect stderr to.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to the new sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04043`: if the call specification does not exist.
- `ORA-06530`: if the sink is `NULL`.

SET_ICS_STDERR_TO_DBMS_OUTPUT Procedure

This procedure redirects the `stderr` stream of the inlined MLE call specification context to `DBMS_OUTPUT` for the calling user in the current session. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_ICS_STDERR_TO_DBMS_OUTPUT(
   name       IN      VARCHAR2);
```

Parameters

Table 133-22 SET_ICS_STDERR_TO_DBMS_OUTPUT Procedure Parameters

Parameter	Description
`name`	The name of the inlined MLE call specification.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to `DBMS_OUTPUT`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04043`: if the call specification does not exist.

SET_ICS_STDOUT Procedure

This procedure redirects the `stdout` stream of an inlined MLE call specification context in the current session to the given `CLOB`.

Syntax

```
DBMS_MLE.SET_ICS_STDOUT(
    name          IN         VARCHAR2,
    sink          IN         OUT NOCOPY CLOB CHARACTER SET ANY_CS);
```

Parameters

Table 133-23 SET_ICS_STDOUT Procedure Parameters

Parameter	Description
`name`	The name of the inlined MLE call specification.
`sink`	The `CLOB` sink to redirect `stdout` to. Providing a `NULL` value will result in an `ORA-06530` error.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to the new sink.

This procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04043`: if the call specification does not exist.
- `ORA-06530`: if the sink is `NULL`.

SET_ICS_STDOUT_TO_DBMS_OUTPUT Procedure

This procedure redirects the `stdout` stream of the inlined call specification context to `DBMS_OUTPUT` for the calling user in the current session. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_ICS_STDOUT_TO_DBMS_OUTPUT(
   name       IN      VARCHAR2);
```

Parameters

Table 133-24 SET_ICS_STDOUT_TO_DBMS_OUTPUT Procedure Parameters

Parameter	Description
`name`	The name of the inlined call specification.

Usage Notes

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to `DBMS_OUTPUT`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04043`: if the call specification does not exist.

SET_STDERR Procedure

This procedure redirects the `stderr` stream of MLE contexts to the given `CLOB`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_STDERR(
      sink            IN OUT       NOCOPY CLOB CHARACTER SET ANY_CS);

DBMS_MLE.SET_STDERR(
      module_name     IN           VARCHAR2,
      sink            IN OUT       NOCOPY CLOB CHARACTER SET ANY_CS);

DBMS_MLE.SET_STDERR(
      module_name     IN           VARCHAR2,
      env_name        IN           VARCHAR2,
      sink            IN OUT       NOCOPY CLOB CHARACTER SET ANY_CS);
```

Parameters

Table 133-25 SET_STDERR Procedure Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment.
`sink`	The CLOB to redirect `stdout` to.

Usage Notes

When called without parameters, this procedure applies to all existing contexts and contexts created in the future. Otherwise, only the context associated with the given module (or module and environment combination) is affected.

If no environment is specified, the context defined by the given module and the built-in environment is used.

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to the new sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges.
- `ORA-04103`: if the module does not exist.
- `ORA-04105`: if the environment does not exist.
- `ORA-06530`: if the sink is `NULL`.

SET_STDERR_TO_DBMS_OUTPUT Procedure

This procedure redirects the `stderr` stream of MLE contexts to `DBMS_OUTPUT`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_STDERR_TO_DBMS_OUTPUT();

DBMS_MLE.SET_STDERR_TO_DBMS_OUTPUT(
   module_name          IN         VARCHAR2);

DBMS_MLE.SET_STDERR_TO_DBMS_OUTPUT(
   module_name          IN         VARCHAR2,
   env_name             IN         VARCHAR2);
```

Parameters

Table 133-26 SET_STDERR_TO_DBMS_OUTPUT Function Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment.

Usage Notes

When called without parameters, this procedure applies to all existing contexts and contexts created in the future. Otherwise, only the context associated with the given module (or module and environment combination) is affected.

If no environment is specified, the context defined by the given module and the built-in environment is used.

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to `DBMS_OUTPUT`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04103`: if the module does not exist
- `ORA-04105`: if the environment does not exist

SET_STDOUT Procedure

This procedure redirects the `stdout` stream of MLE contexts to the given `CLOB`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_STDOUT(
   sink           IN OUT   NOCOPY CLOB CHARACTER SET ANY_CS);
```

```
DBMS_MLE.SET_STDOUT(
   module_name    IN       VARCHAR2,
   sink           IN OUT   NOCOPY CLOB CHARACTER SET ANY_CS);

DBMS_MLE.SET_STDOUT(
   module_name    IN       VARCHAR2,
   env_name       IN       VARCHAR2,
   sink           IN OUT   NOCOPY CLOB CHARACTER SET ANY_CS);
```

Parameters

Table 133-27 SET_STDOUT Procedure Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment.
`sink`	The CLOB to redirect stdout to.

Usage Notes

When called without parameters, this procedure applies to all existing contexts and contexts created in the future. Otherwise, only the context associated with the given module (or module and environment combination) is affected.

If no environment is specified, the context defined by the given module and the built-in environment is used.

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to the new sink.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04103`: if the module does not exist
- `ORA-04105`: if the environment does not exist
- `ORA-06530`: if the sink is `NULL`.

SET_STDOUT_TO_DBMS_OUTPUT Procedure

This procedure redirects the `stdout` stream of MLE contexts to `DBMS_OUTPUT`. You need the `EXECUTE DYNAMIC MLE` privilege to execute this function.

Syntax

```
DBMS_MLE.SET_STDOUT_TO_DBMS_OUTPUT();

DBMS_MLE.SET_STDOUT_TO_DBMS_OUTPUT(
      module_name      IN      VARCHAR2);

DBMS_MLE.SET_STDOUT_TO_DBMS_OUTPUT(
      module_name      IN      VARCHAR2,
      env_name         IN      VARCHAR2);
```

Parameters

Table 133-28 SET_STDOUT_TO_DBMS_OUTPUT Function Parameters

Parameter	Description
`module_name`	The name of the MLE module.
`env_name`	The name of the MLE environment.

Usage Notes

When called without parameters, this procedure applies to all existing contexts and contexts created in the future. Otherwise, only the context associated with the given module (or module and environment combination) is affected.

If no environment is specified, the context defined by the given module and the built-in environment is used.

Any output that was buffered so far gets flushed to the pre-existing sink before redirecting to `DBMS_OUTPUT`.

The procedure may raise the following errors:

- `ORA-01031`: if the caller does not have sufficient privileges
- `ORA-04103`: if the module does not exist
- `ORA-04105`: if the environment does not exist