The file containing the `log_module_message` procedure appears to be a part of a logging system for an Oracle database application. This procedure is responsible for recording messages related to application module activities, including errors, warnings, and notifications. Here's a high-level summary of its functionalities:

1. **Purpose**: To log messages associated with specific modules of an application, detailing the message content, related package or procedure name, severity, and other contextual information.

2. **Autonomous Transaction**: The procedure executes independently of the main transaction, ensuring that the logs are recorded even if the main transaction fails or is rolled back.

3. **Validation**: It performs checks on the input parameters to ensure they meet certain criteria, such as length restrictions and predefined acceptable values. Any validation failure triggers an error.

4. **Debugging Support**: It can check for a debugging parameter to decide whether to record certain types of messages.

5. **Logging Mechanism**: It logs messages into the `xxmx_module_messages` table, including a unique message ID, relevant application and module information, and the message text. The severity level of the message determines how it is logged.

6. **Exception Handling**: The procedure handles exceptions by logging them appropriately and using the `RAISE_APPLICATION_ERROR` function to notify the calling procedure of the issue. It distinguishes between a custom `ModuleError` and other Oracle errors.

7. **Transaction Commitment**: It commits the transaction after inserting the log record to ensure that the log entry is saved.

8. **Example Usage**: The procedure is designed to be used within the application code, potentially across different modules, to consistently log messages about system behavior, especially for monitoring and debugging purposes.

In summary, the `log_module_message` procedure is a centralized logging utility for an Oracle application, aimed at capturing a wide range of messages with consistent formatting and storage, facilitating easier monitoring and troubleshooting of the application.

log_module_message

log_module_message Procedure Documentation

Overview

The `log_module_message` procedure is used to log messages for different phases and severity levels of a module within an application. It supports dynamic error handling, logging to a database table, and raises application errors when necessary.

Procedure Signature

```
PROCEDURE log_module_message
    (
     pt_i_ApplicationSuite         IN xxmx_module_messages.application_suite%TYPE,
     pt_i_Application              IN xxmx_module_messages.application%TYPE,
     pt_i_BusinessEntity           IN xxmx_module_messages.business_entity%TYPE,
     pt_i_SubEntity                IN xxmx_module_messages.sub_entity%TYPE,
     pt_i_Phase                    IN xxmx_module_messages.phase%TYPE,
     pt_i_Severity                 IN xxmx_module_messages.severity%TYPE,
     pt_i_PackageName              IN xxmx_module_messages.package_name%TYPE,
     pt_i_ProcOrFuncName           IN xxmx_module_messages.proc_or_func_name%TYPE,
     pt_i_ProgressIndicator        IN xxmx_module_messages.progress_indicator%TYPE,
     pt_i_ModuleMessage            IN xxmx_module_messages.module_message%TYPE,
     pt_i_OracleError              IN xxmx_module_messages.oracle_error%TYPE
    )
```

Parameters

- `pt_i_ApplicationSuite`: The application suite for which the message is being logged.
- `pt_i_Application`: The specific application within the suite.
- `pt_i_BusinessEntity`: The business entity relevant to the message.
- `pt_i_SubEntity`: The sub-entity relevant to the message.
- `pt_i_Phase`: The phase of the application where the message is generated.
- `pt_i_Severity`: The severity level of the message.
- `pt_i_PackageName`: The package name of the module logging the message.
- `pt_i_ProcOrFuncName`: The procedure or function name within the package.
- `pt_i_ProgressIndicator`: A variable to track progress or status.
- `pt_i_ModuleMessage`: The message to be logged.
- `pt_i_OracleError`: The Oracle error code, if applicable.

Autonomous Transaction

```
PRAGMA AUTONOMOUS_TRANSACTION;
```

The procedure runs as an autonomous transaction, which means it will not be affected by the current transaction context.

Constants and Variables

- `ct_ProcOrFuncName`: A constant holding the name of the current procedure `log_module_message`.
- `v_debug_message`: A variable to store the debug message status.
- `e_ModuleError`: A user-defined exception.

Procedure Logic

Validation

The procedure starts with several validation checks:

1. It checks whether the length of `pt_i_ApplicationSuite` and `pt_i_Application` parameters does not exceed 4 characters, raising `e_ModuleError` if the condition is violated.

2. It validates if the `pt_i_Phase` parameter is one of the allowed values: 'CORE', 'EXTRACT', 'TRANSFORM', 'EXPORT', or 'VALIDATE', raising `e_ModuleError` if not.

3. It checks if the `pt_i_Severity` parameter is either 'NOTIFICATION', 'WARNING', or 'ERROR', also raising `e_ModuleError` if the value is outside this range.

Debug Parameter Check

It retrieves a parameter value for debugging purposes using `xxmx_utilities_pkg.get_single_parameter_value` function.

Message Logging

Depending on the `pt_i_Severity` level and the debug parameter, an INSERT statement logs the message into the `xxmx_module_messages` table with relevant information including a unique message ID, application information, and timestamps.

Commit Transaction

After the insert operation, a `COMMIT` statement is executed to save the changes made by the autonomous transaction.

Exception Handling

Module Error

If `e_ModuleError` is raised, an error message is constructed and inserted into the `xxmx_module_messages` table, then a `COMMIT` is executed followed by `RAISE_APPLICATION_ERROR` to report the error with an appropriate message.

Other Errors

For any other errors, the procedure captures the Oracle error message, logs it into the `xxmx_module_messages` table, commits the change, and raises an application error.

Example Usage

The procedure is intended to be called from various procedures within the application whenever there is a need to log a message to the `xxmx_module_messages` table, including during error handling.

```
BEGIN
    log_module_message(
        pt_i_ApplicationSuite      => 'ACCT',
        pt_i_Application           => 'PAYR',
        pt_i_BusinessEntity        => 'HR',
        pt_i_SubEntity             => 'PAYROLL',
        pt_i_Phase                 => 'TRANSFORM',
        pt_i_Severity              => 'ERROR',
        pt_i_PackageName           => 'payroll_pkg',
        pt_i_ProcOrFuncName        => 'calculate_salaries',
        pt_i_ProgressIndicator     => 'STEP1',
        pt_i_ModuleMessage         => 'Invalid salary data.',
        pt_i_OracleError           => 'ORA-01400'
    );
END;
```

This call to `log_module_message` would attempt to log an error message from the `payroll_pkg.calculate_salaries` procedure during the 'TRANSFORM' phase, with the progress indicator 'STEP1'. If any of the validation checks fail or an exception is raised, it would log an appropriate error message and raise an application error for the calling procedure to handle.