Lua Engine Function and DataCallBack Documentation

V1.0

Lua Integration API Documentation

Overview

This document provides a comprehensive guide to the Lua-accessible methods available in the `Engine` class, used primarily for managing and interacting with Packager instances. Each function is designed to be registered and called from Lua scripts, facilitating a dynamic and configurable integration within a software system.

Functions

SetMetadata

Sets metadata for a specific packager.

Parameters:

- `Id`: Integer - ID of the packager.
- `Key`: String - Metadata key.
- `Value`: String - Metadata value.
- `identifier`: String (optional) - Identifier for the action, used for cancellation.

Example:

```
SetMetadata(123, "genre", "jazz")
```

SetMetadataWithDelay

Sets metadata with a specified delay.

Parameters:

- `id`: Integer - ID of the packager.
- `Key`: String - Metadata key.
- `Value`: String - Metadata value.
- `delay`: Integer - Delay in milliseconds before the action is executed.
- `identifier`: String (optional) - Unique identifier for the action.

Example:

```
SetMetadataWithDelay(123, "genre", "rock", 5000)
```

SetMetadataByTime

Sets metadata at a specified due time.

Parameters:

- `id`: Integer - ID of the packager.
- `Key`: String - Metadata key.
- `Value`: String - Metadata value.
- `DueTime`: DateTime - Specific time when the metadata should be set.
- `identifier`: String (optional) - Identifier for the action.

Example:

```
SetMetadataByTime(123, "genre", "pop", DateTime("2024-12-31T23:59:59"))
```

SetMetadataTable

Sets a metadata table for a specific packager.

Parameters:

- `Id`: Integer - ID of the packager.
- `table`: LuaTable - Table containing key-value pairs of metadata.
- `identifier`: String (optional) - Identifier for the action, used for cancellation.

Example:

```
SetMetadataTable(123, {genre = "classical", artist = "Mozart"})
```

SetMetadataTableWithDelay

Sets a metadata table with a delay.

Parameters:

- `Id`: Integer - ID of the packager.
- `table`: LuaTable - Table of metadata.
- `delay`: Integer - Delay in milliseconds before the action is executed.
- `identifier`: String (optional) - Unique identifier for the action.

Example:

```
SetMetadataTableWithDelay(123, {album = "The Dark Side of the Moon"},
10000)
```

SetMetadataTableByTime

Sets a metadata table at a specified due time.

Parameters:

- `Id`: Integer - ID of the packager.
- `table`: LuaTable - Table of metadata.
- `DueTime`: DateTime - Specific time when the metadata should be set.
- `identifier`: String (optional) - Identifier for the action.

Example:

```
SetMetadataTableByTime(123, {track = "Time"},
DateTime("2024-12-31T23:59:59"))
```

CancelSetMetadataWithDelay

Cancels a delayed SetMetadata action.

Parameters:

- `identifier`: String - Identifier of the delayed action to cancel.

Example:

```
CancelSetMetadataWithDelay("12345-identifier")
```

GetGPI

Retrieves the state of GPI pins for a specified port.

Parameters:

- `port`: Integer - Index of the GPI port.

Returns:

- Array of Boolean values representing the state of each pin.

Example:

```
GetGPI(1)
```

GetGPO

Retrieves the state of GPO pins for a specified port.

Parameters:

- `port`: Integer - Index of the GPO port.

Returns:

- Array of Boolean values representing the state of each pin.

Example:

```
GetGPO(2)
```

GetCFGGPO

Retrieves the configuration of a GPO port.

Parameters:

- `port`: Integer - Index of the GPO port.

Returns:

- String - Current route configuration of the port.

Example:

```
GetCFGGPO(3)
```

Log

Logs a message to the system.

Parameters:

- `message`: Object - The message to log.

Example:

```
Log("System initialized successfully.")
```

Email

Sends an email through the system.

Parameters:

- `Subject`: String - Subject of the email.
- `MessageBody`: String - Body of the email.
- `ToAddress`: String - Comma-delimited list of recipient addresses.

Example:

```
Email("Alert", "System error detected.", "admin@example.com")
```

LoadOrCreatePersistantVariable

Loads a persistent variable from the database or creates it if it does not exist.

Parameters:

- `name`: String - Name of the variable.
- `value`: Object (optional) - Value to set if the variable is created.

Returns:

- Object - Value of the persistent variable.

Example:

```
LoadOrCreatePersistantVariable("last_checked", DateTime.Now)
```

UpdatePersistantVariable

Updates a persistent variable.

Parameters:

- `name`: String - Name of the variable.
- `newValue`: String - New value for the variable.

Example:

```
UpdatePersistantVariable("last_checked", "2024-12-31T23:59:59")
```

DeletePersistantVariable

Deletes a persistent variable.

Parameters:

- `name`: String - Name of the variable to delete.

Example:

```
DeletePersistantVariable("obsolete_variable")
```

Lua Built-in Functions Documentation

Basic Functions

print(...)

Prints its arguments with a conversion to strings as necessary.

Parameters:

- `...`: Variable arguments to be printed.

Example:

```
print("Hello", "world")
```

tonumber(e)

Attempts to convert its argument to a number.

Parameters:

- `e`: Expression to convert.

Returns:

- Number or nil if conversion fails.

Example:

```
local num = tonumber("123")
```

tostring(e)

Converts its argument to a string.

Parameters:

- `e`: Expression to convert.

Returns:

- String representation of the input.

Example:

```
local str = tostring(123)
```

type(v)

Returns the type of its argument as a string.

Parameters:

- `v`: Variable to check the type.

Returns:

- String describing the type of `v`.

Example:

```
print(type("Hello"))  -- Outputs "string"
```

String Manipulation

string.upper(s)

Converts all lowercase letters in a string to uppercase.

Parameters:

- `s`: String to convert.

Returns:

- Converted string.

Example:

```
print(string.upper("hello"))  -- Outputs "HELLO"
```

string.lower(s)

Converts all uppercase letters in a string to lowercase.

Parameters:

- `s`: String to convert.

Returns:

- Converted string.

Example:

```
print(string.lower("HELLO"))  -- Outputs "hello"
```

string.gsub(s, pattern, repl [, n])

Returns a copy of `s` with occurrences of the pattern replaced by `repl`.

Parameters:

- `s`: Source string.

- `pattern`: Pattern to search for.
- `repl`: Replacement string.
- `n`: Maximum number of replacements (optional).

Returns:

- Modified string.

Example:

```
print(string.gsub("banana", "a", "o"))  -- Outputs "bonono"
```

string.find(s, pattern [, init [, plain]])

Searches for the first occurrence of the pattern.

Parameters:

- `s`: Source string.
- `pattern`: Pattern to search for.
- `init`: Initial position to start the search (optional).
- `plain`: If true, turns off the pattern matching facilities (optional).

Returns:

- Start and end indices of the found occurrence, or nil if not found.

Example:

```
print(string.find("hello world", "world"))  -- Outputs "7 11"
```

string.reverse(s)

Reverses a string.

Parameters:

- `s`: String to reverse.

Returns:

- Reversed string.

Example:

```
print(string.reverse("hello"))  -- Outputs "olleh"
```

Mathematical Functions

math.abs(x)

Returns the absolute value of `x`.

Parameters:

- `x`: Number.

Returns:

- Absolute value of `x`.

Example:

```
print(math.abs(-5))  -- Outputs "5"
```

math.floor(x)

Returns the largest integer less than or equal to `x`.

Parameters:

- `x`: Number.

Returns:

- Floor value of `x`.

Example:

```
print(math.floor(3.7))  -- Outputs "3"
```

math.ceil(x)

Returns the smallest integer greater than or equal to `x`.

Parameters:

- `x`: Number.

Returns:

- Ceiling value of `x`.

Example:

```
print(math.ceil(3.7))  -- Outputs "4"
```

math.random([m [, n]])

Generates a pseudo-random number.

Parameters:

- `m`: Lower limit (optional).
- `n`: Upper limit (optional).

Returns:

- Random number.

Example:

```
print(math.random(1, 10))  -- Outputs a random number between 1 and 10
```

math.sin(x), math.cos(x), math.tan(x)

Trigonometric functions returning sine, cosine, and tangent of `x` (in radians).

Parameters:

- `x`: Angle in radians.

Returns:

- Trigonometric value.

Examples:

```
print(math.sin(math.pi/2))  -- Outputs "1"
print(math.cos(math.pi))  -- Outputs "-1"
```

```
print(math.tan(0))  -- Outputs "0"
```

Input/Output Functions

io.read(...)

Reads input according to the given formats.

Parameters:

- `...`: Formats specifying what to read.

Returns:

- Read input.

Example:

```
local input = io.read("*line")  -- Reads a line from standard input
```

io.write(...)

Writes values to standard output.

Parameters:

- `...`: Values to write.

Example:

```
io.write("Hello", " ", "world")
```

io.open(filename [, mode])

Opens a file.

Parameters:

- `filename`: Path to the file.
- `mode`: Mode of opening the file (optional).

Returns:

- File handle and an error message if applicable.

Example:

```
local file, err = io.open("test.txt", "r")
```

Coroutine Functions

coroutine.create(f)

Creates a new coroutine.

Parameters:

- `f`: Function to execute in the coroutine.

Returns:

- Coroutine handle.

Example:

```
local co = coroutine.create(function () print("Hello, coroutine!") end)
```

coroutine.resume(co [, val1, ...])

Starts or continues the execution of a coroutine.

Parameters:

- `co`: Coroutine to resume.
- `val1, ...`: Values to pass to the coroutine.

Returns:

- True if no errors, false and the error message otherwise.

Example:

```
coroutine.resume(co)
```

coroutine.yield(...)

Suspends the execution of the calling coroutine.

Parameters:

- `...`: Values to return on suspension.

Example:

```
coroutine.yield()
```

Lua Script Callback Usage for Different Data Types

This guide expands on the previous explanations, focusing on how each data type is passed to Lua functions.. Each Lua function should be prepared to handle the specific format and structure of the data as described below.

Bytes (DataReceivedBytes)

Usage: The function receives the raw byte array along with its length. The dictionary may contain metadata about the data.

Lua Function Example for Bytes with Header Check

```
function handleBytesWithHeader(dataDictionary, dataChannelId, bytes,
length)
    -- Define the header you're looking for
    local header = {0xAA, 0xBB, 0xCC}  -- Example header bytes

    -- Function to compare the header with the beginning of the bytes
array
    local function checkHeader(bytes, header)
        for i = 1, #header do
            if bytes[i] ~= header[i] then
                return false
            end
        end
        return true
    end

    -- Call the function to check for the header
    if checkHeader(bytes, header) then
```

```
        print("Header matched in data channel", dataChannelId)
        -- Process bytes following the header
    else
        print("Header did not match in data channel", dataChannelId)
    end
end
```

JSON (DataReceivedJson)

Usage: JSON data is parsed into a Lua dictionary using a nested dictionary structure to maintain JSON hierarchy.

Example JSON Object

```
{
  "libraryName": "Central City Library",
  "address": "123 Library St, Central City",
  "books": [
    {
      "title": "The Great Gatsby",
      "author": "F. Scott Fitzgerald",
      "publishedYear": 1925,
      "genre": "Fiction",
      "reviews": [
        {
          "reviewer": "John Doe",
          "rating": 5,
          "comment": "A timeless masterpiece that captures the essence
of the American Jazz Age."
        },
        {
          "reviewer": "Jane Smith",
          "rating": 4,
          "comment": "Beautifully written, though a bit melancholic."
        }
      ]
    },
    {
      "title": "1984",
      "author": "George Orwell",
      "publishedYear": 1949,
      "genre": "Dystopian Fiction",
      "reviews": [
        {
```

```
          "reviewer": "Alice Johnson",
          "rating": 5,
          "comment": "A chilling prediction of the future, still
relevant today."
        },
        {
          "reviewer": "Bob Brown",
          "rating": 4,
          "comment": "A profound book on surveillance and individuality.
A bit heavy at times."
        }
      ]
    }
  ]
}
```

Example LUA Function using JSON Object

```
function handleJson(dataDictionary, dataChannelId, jsonObject)
    -- Print library information
    print("Received JSON data on channel", dataChannelId)
    print("Library Name:", jsonObject["libraryName"])
    print("Library Address:", jsonObject["address"])

    -- Access the 'books' array
    local books = jsonObject["books"]
    if books then
        print("Number of books in the library:", #books)
        for i, book in ipairs(books) do
            -- Print book details
            print("Book " .. i .. ":")
            print("  Title:", book["title"])
            print("  Author:", book["author"])
            print("  Published Year:", book["publishedYear"])
            print("  Genre:", book["genre"])

            -- Access the 'reviews' array for each book
            local reviews = book["reviews"]
            if reviews then
                print("  Number of reviews:", #reviews)
                for j, review in ipairs(reviews) do
                    -- Print review details
                    print("  Review " .. j .. ":")
```

```
                    print("    Reviewer:", review["reviewer"])
                    print("    Rating:", review["rating"])
                    print("    Comment:", review["comment"])
                end
            else
                print("  No reviews available for this book.")
            end
        end
    else
        print("No books found in the library JSON.")
    end
end
```

XML (DataReceivedXml)

Usage: XML data is converted to a JSON object using `JsonConvert.SerializeXmlNode`, making it accessible as a dictionary in Lua.

XML Example Data

```
<library>
    <libraryName>Central City Library</libraryName>
    <address>123 Library St, Central City</address>
    <books>
        <book title="The Great Gatsby" author="F. Scott Fitzgerald">
            <publishedYear>1925</publishedYear>
            <genre>Fiction</genre>
            <reviews>
                <review>
                    <reviewer>John Doe</reviewer>
                    <rating>5</rating>
                    <comment>A timeless masterpiece that captures the
essence of the American Jazz Age.</comment>
                </review>
                <review>
                    <reviewer>Jane Smith</reviewer>
                    <rating>4</rating>
                    <comment>Beautifully written, though a bit
melancholic.</comment>
                </review>
            </reviews>
        </book>
```

```
        <book title="1984" author="George Orwell">
            <publishedYear>1949</publishedYear>
            <genre>Dystopian Fiction</genre>
            <reviews>
                <review>
                    <reviewer>Alice Johnson</reviewer>
                    <rating>5</rating>
                    <comment>A chilling prediction of the future, still
relevant today.</comment>
                </review>
                <review>
                    <reviewer>Bob Brown</reviewer>
                    <rating>4</rating>
                    <comment>A profound book on surveillance and
individuality. A bit heavy at times.</comment>
                </review>
            </reviews>
        </book>
    </books>
</library>
```

Handling JSON Data Derived from XML

```
function handleXml(dataDictionary, dataChannelId, xmlObject)
    -- Print library information
    print("Received JSON data on channel", dataChannelId)
    print("Library Name:", xmlObject.libraryName)
    print("Library Address:", xmlObject.address)

    -- Access the 'books' array
    local books = xmlObject.books
    if books then
        print("Number of books in the library:", #books)
        for i, book in ipairs(books) do
            -- Print book details with attributes accessed
            print("Book " .. i .. ":")
            print("  Title:", book["@Title"])  -- Access title as an
attribute
            print("  Author:", book["@Author"])  -- Access author as an
attribute
            print("  Published Year:", book.publishedYear)
```

```
            print("  Genre:", book.genre)

            -- Access the 'reviews' array for each book
            local reviews = book.reviews
            if reviews then
                print("  Number of reviews:", #reviews)
                for j, review in ipairs(reviews) do
                    -- Print review details
                    print("  Review " .. j .. ":")
                    print("    Reviewer:", review.reviewer)
                    print("    Rating:", review.rating)
                    print("    Comment:", review.comment)
                end
            else
                print("  No reviews available for this book.")
            end
        end
    else
        print("No books found in the library JSON.")
    end
end
```

UTF-8 String (DataReceivedUtf8)

Usage: UTF-8 data is passed directly as a string. Any related data or metadata is passed in a dictionary.

Lua Function for Handling and Processing UTF-8 Strings

```
function handleUtf8(dataDictionary, dataChannelId, utf8String)
    print("Received UTF-8 string on channel", dataChannelId)

    -- Example processing: Assuming utf8String is a CSV line of values
    local values = {}
    for value in utf8String:gmatch("[^,]+") do
        table.insert(values, value:trim())  -- Trim function to remove
any spaces around values
    end

    -- Display the parsed values
    if #values > 0 then
        print("Parsed values from the UTF-8 string:")
        for i, v in ipairs(values) do
            print(i .. ": " .. v)
```

```
        end
    else
        print("No values parsed from the UTF-8 string.")
    end

    -- Further processing: For example, checking if data meets certain
conditions
    if #values > 0 and values[1] == "ExpectedHeader" then
        print("Data matches expected header, proceeding with
processing...")
        -- Additional logic to process the data
    else
        print("Data does not match expected criteria.")
    end
end

-- Helper function to trim strings
function string.trim(self)
    return self:match("^%s*(.-)%s*$")
end
```

Axia Data (AxiaDataReceived)

Usage: Custom data specific to Axia, represented as a list of `ResponseObject`. Each object includes a response type and response data.

```
public enum ResponseType
{
    VER =0,
    SET =1,
    SRC =2,
    DST =3,
    MTR =4,
    GPO =5,
    GPI =6,
    LVL =7,
    ERROR =8,
    CFGGPO =9,
    DEC =10,
    ENC =11,
    TZD =12
}
```

Lua Function for Handling Axia Data

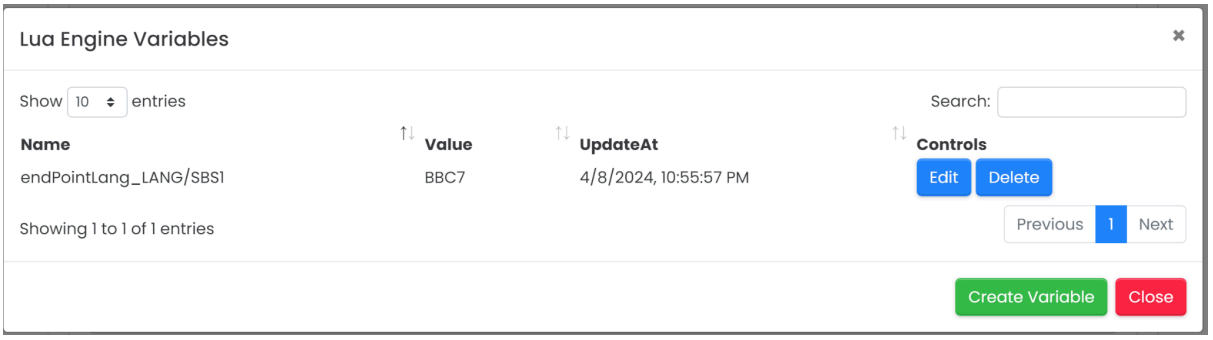
```
function AxiaRx(source, Obj)
    -- Loop through each object in the Axia data
    for i = 0, Obj.Count - 1 do
        local axiaObject = Obj[i]

        -- Check if the object is a GPO and its port matches specific
criteria
        if axiaObject.Type == 5 then  -- Assuming Type 5 is GPO
            -- Loop through each pin in the GPO (assuming there are 5
pins)
            for pinIndex = 0, 4 do
                -- Check each studio port to see if it matches the
current GPO port and pin
                for _, studio in pairs(studioPort) do
                    local portNumber = studio[1]
                    local pinNumber = studio[2]

                    if portNumber == axiaObject.Response.PORT and
pinNumber == pinIndex + 1 then
                        local pinState =
axiaObject.Response.Pins[pinIndex]
                        studio[3] = pinState  -- Update the studio
port's state based on the pin state

                        -- Log the pin state
                        Log("Port: " .. tostring(portNumber) .. " Pin: "
.. tostring(pinNumber) .. " State: " .. tostring(pinState))
                    end
                end
            end
        end
    end
end
```

Example Lua Functions

Reading and Writing from the persistent Datastore

```
-- Function to handle JSON data for configuration settings
function handleConfigData(dataDictionary, dataChannelId, jsonConfig)
    print("Received configuration data on channel", dataChannelId)

    -- Load or create persistent variables based on received JSON
    if jsonConfig.theme then
        loadOrCreatePersistentVariable("theme", jsonConfig.theme)
    end

    if jsonConfig.language then
        loadOrCreatePersistentVariable("language", jsonConfig.language)
    end

    if jsonConfig.notificationsEnabled ~= nil then  -- Check for boolean
as well, hence nil check
        loadOrCreatePersistentVariable("notificationsEnabled",
tostring(jsonConfig.notificationsEnabled))
    end

    -- Example of conditionally deleting a variable based on special
instruction in JSON
    if jsonConfig.deleteSetting and jsonConfig.deleteSetting ~= "" then
        deletePersistentVariable(jsonConfig.deleteSetting)
        print("Requested to delete setting:", jsonConfig.deleteSetting)
    end

    -- Optionally update existing settings if specified
    if jsonConfig.updateSettings and type(jsonConfig.updateSettings) ==
"table" then
        for key, value in pairs(jsonConfig.updateSettings) do
            updatePersistentVariable(key, value)
            print("Updated setting", key, "to", value)
        end
    end
end
```

Persistent variables can be accessed from the web interface



Reading and Writing from memory

```
-- In-memory table to simulate persistent storage
local configStore = {}

-- Function to handle JSON data for configuration settings using
in-memory storage
function handleConfigData(dataDictionary, dataChannelId, jsonConfig)
    print("Received configuration data on channel", dataChannelId)

    -- Set or update configuration settings in the in-memory store
    if jsonConfig.theme then
        configStore.theme = jsonConfig.theme
        print("Set theme to:", jsonConfig.theme)
    end

    if jsonConfig.language then
        configStore.language = jsonConfig.language
        print("Set language to:", jsonConfig.language)
    end

    if jsonConfig.notificationsEnabled ~= nil then  -- Handle boolean
values directly
        configStore.notificationsEnabled =
jsonConfig.notificationsEnabled
        print("Set notificationsEnabled to:",
tostring(jsonConfig.notificationsEnabled))
    end

    -- Example of conditionally deleting a variable based on special
instruction in JSON
    if jsonConfig.deleteSetting and
configStore[jsonConfig.deleteSetting] then
```

```
        configStore[jsonConfig.deleteSetting] = nil
        print("Deleted setting:", jsonConfig.deleteSetting)
    end

    -- Optionally update existing settings if specified in a sub-table
    if jsonConfig.updateSettings and type(jsonConfig.updateSettings) ==
"table" then
        for key, value in pairs(jsonConfig.updateSettings) do
            configStore[key] = value
            print("Updated setting", key, "to", value)
        end
    end
end
```

Setting Metadata from Zetta Now Playing Output

```
packagerId=1;
function isempty(s)
    return s ==nil or s =='' or s =='null'
end

function ZettaLiveMetadata(obj,source,objectXml,stupidNumber)
    if(isempty(objectXml.ZettaClipboard)
        or isempty(objectXml.ZettaClipboard.LogEvents)
        or isempty(objectXml.ZettaClipboard.LogEvents.LogEvent))
    then
        Log('Zetta now playing missing data1')
            return
    end
      logevent=nil
      if(not(isempty(objectXml.ZettaClipboard.LogEvents.LogEvent[1])))
      then
            logevent=objectXml.ZettaClipboard.LogEvents.LogEvent[1]
        Log('Index 1 logevent')
      else
            logevent=objectXml.ZettaClipboard.LogEvents.LogEvent
        Log('logevent raw')
      end

      if(isempty(logevent.AssetEvent)
      or isempty(logevent.AssetEvent.Asset)
      or isempty(logevent.AssetEvent.Asset.Artist))
      then
```

```
          Log('Zetta now playing missing asset data2')
            return;
      end
      Log("Gets Here")

SetMetadataTable(packagerId,{["Artist"]=logevent.AssetEvent.Asset.Artist
["@Name"],["Title"]=logevent.AssetEvent.Asset["@Title"],})


end
```

Example Data

```
<?xml version="1.0" encoding="ASCII"?>
<ZettaClipboard Version="0.0.0" MessageType="0">
  <Station StationID="3" Name="[SYD] 104.1 2Day FM" Active="true" Comment="SYD
SVC 02" StationRole="PRIMARY"
UniversalIdentifier="18554857-15fe-4754-b52e-30d127d2c324" CallLetters="2DAY"
EnterpriseStationId="NSW"/>
  <LogEvents>
    <LogEvent UniversalIdentifier="7b1aa478-a232-ef11-81cb-00155ddb18e4"
LogEventID="131213889" LogGroupID="11552071" Sequence="13"
LogEventEntryTypeID="106" Description="Fortnight - Taylor Swift ft. Post Malone"
AirStarttime="2024-06-28T05:34:55Z" AirStarttimeLocal="28/06/2024 3:34:55 PM"
AirStoptime="2024-06-28T05:38:44Z" AirStoptimeLocal="28/06/2024 3:38:44 PM"
EntryValue1="0" EntryData="" ChainType="1" StatusCode="1" EditCode="0"
ErrorCode="0" TimingType="2" IsFixed="false" DynamicType="1"
IsChildEvent="false">
      <LogGroup LogGroupID="11552071" LogGroupTypeID="101" LogID="248919"
Sequence="15" Hour="15">
        <LogHeader LogID="248919" LogTypeID="0"
Date="2024-06-28T00:00:00.0000000+10:00"/>
      </LogGroup>
      <AssetEvent>
        <Asset AssetID="4972940" AssetTypeID="1" Title="Fortnight"
AssetTypeName="Song" PrimaryResourceID="11757145" Comment="Zetta Api: SCA
Melbourne" infoCreatedDate="2024-04-11T04:48:42.6470000+00:00"
infoModifiedDate="2024-06-19T02:47:44.8330000+00:00" OverrideChainType="0"
PlaybackMethodID="0" BedTypeID="0" IgnorePitching="false"
ThirdPartyID="BNE0000000573"
UniversalIdentifier="bf0cac47-aaed-791f-94f7-9df34bbaa55b" Label="" Publisher=""
ISRC="" MetadataScope="1">
          <Resource ResourceID="11757145" ResourceTypeID="1"
ResourceFile="\\ZETSRV-001\Content_Store\11\757\11757145~001_Taylor Swift Post
Malone_Fortnight_USUG12401028.wav" Length="228.9655"
CheckSumMD5="-8075287915822506372" Year="0" FileSize="40389590" StartOffset="0"
infoCreatedDate="0001-01-01T00:00:00.0000000+00:00"
```

```
infoModifiedDate="0001-01-01T00:00:00.0000000+00:00"
UniversalIdentifier="f338224a-1d15-4b7c-84a7-0b01b9cc23c2"
FileMD5="65B723D6F2133F29A7F9782D1D6BA758" DatabaseRootMeanSquare="8706"
DatabaseStandardPeakMean="27379" DatabaseStandardPeak="24352"
DatabaseAbsolutePeak="32766" DatabaseLoudnessBs1770rev0="-9.9863"
DatabaseLoudnessBs1770rev3="-9.8317">
            <Transition ResourceLength="228.9655" Filename="11757145~001_Taylor
Swift Post Malone_Fortnight_USUG12401028.wav" PlaybackMode="OnAir"
FileMD5="65B723D6F2133F29A7F9782D1D6BA758">
              <PointOfInterestMarker Position="0" Enabled="true"
Duration="3.436" Value="0" Details="" POIType="6"/>
              <PointOfInterestMarker Position="152.312" Enabled="false"
Duration="30" Value="0" POIType="2"/>
              <PointOfInterestMarker Position="223.565" Enabled="true"
Duration="0" Value="5.4005" Details="" POIType="8"/>
              <VolumeMarker Position="0" Level="1"/>
              <VolumeMarker Position="228.9655" Level="1"/>
            </Transition>
          </Resource>
          <Artist ArtistID="19603" ArtistTypeID="0" Name="Taylor Swift ft. Post
Malone"/>
          <Album AlbumID="16743" Name="THE TORTURED POETS DEPARTMENT"/>
          <Participant ParticipantId="4662" Name="Taylor Swift" RoleId="7"/>
          <Participant ParticipantId="7165" Name="Post Malone" RoleId="7"/>
          <Participant ParticipantId="30443" Name="Taylor Swift ft. Post Malone"
RoleId="7"/>
          <AssetAttribute AttributeTypeID="1" AttributeTypeName="Mood"
AttributeValueID="2" AttributeValueName="2 - Feel Good"/>
          <StationSpecific StationSpecificId="24079091"
UniversalIdentifier="cf4bdbda-1cba-4699-924d-32593cee9ceb" IsActive="true"
Station="[SYD] 104.1 2Day FM" StationID="3"
StationUniversalIdentifier="18554857-15fe-4754-b52e-30d127d2c324">
            <Category CategoryID="2400" Name="R1" AssetTypeID="1"
CategoryGroup="R Extra" LongName="Extra"/>
          </StationSpecific>
        </Asset>
      </AssetEvent>
    </LogEvent>
    <LogEvent UniversalIdentifier="7c1aa478-a232-ef11-81cb-00155ddb18e4"
LogEventID="131213890" LogGroupID="11552071" Sequence="14"
LogEventEntryTypeID="106" Description="Hot In Herre [Radio Edit] - Nelly"
AirStarttime="2024-06-28T05:38:38Z" AirStarttimeLocal="28/06/2024 3:38:38 PM"
AirStoptime="2024-06-28T05:42:26Z" AirStoptimeLocal="28/06/2024 3:42:26 PM"
EntryValue1="0" EntryData="" ChainType="1" StatusCode="1" EditCode="0"
ErrorCode="0" TimingType="2" IsFixed="false" DynamicType="1"
IsChildEvent="false">
      <LogGroup LogGroupID="11552071" LogGroupTypeID="101" LogID="248919"
Sequence="15" Hour="15">
        <LogHeader LogID="248919" LogTypeID="0"
Date="2024-06-28T00:00:00.0000000+10:00"/>
      </LogGroup>
```

```
      <AssetEvent>
        <Asset AssetID="29514" AssetTypeID="1" Title="Hot In Herre [Radio Edit]"
AssetTypeName="Song" PrimaryResourceID="36595"
infoCreatedDate="2016-10-28T00:25:40.2330000+00:00"
infoModifiedDate="2024-06-26T04:27:50.6500000+00:00" OverrideChainType="0"
PlaybackMethodID="0" BedTypeID="0" IgnorePitching="false"
ThirdPartyID="GOS0000007120"
UniversalIdentifier="5ba45ac2-38dd-49e4-912b-46d3e854cdf1" Label="" Publisher=""
Mix="2001" ISRC="" MetadataScope="1">
          <Resource ResourceID="36595" ResourceTypeID="1"
ResourceFile="\\ZETSRV-001\Content_Store\36\36595~GOS0000007120.wav"
Length="229.7828" CheckSumMD5="7038448669228539083" Year="0" FileSize="44118342"
StartOffset="0" infoCreatedDate="0001-01-01T00:00:00.0000000+00:00"
infoModifiedDate="0001-01-01T00:00:00.0000000+00:00"
UniversalIdentifier="84276b37-be65-47c4-8b0e-975bfd0e8e24"
FileMD5="CD3E524CF840898D151DEDB761AEB10E" DatabaseRootMeanSquare="2399"
DatabaseStandardPeakMean="9285" DatabaseStandardPeak="8800"
DatabaseAbsolutePeak="10974" DatabaseLoudnessBs1770rev0="-21.2935"
DatabaseLoudnessBs1770rev3="-21.2097">
            <Transition ResourceLength="229.7828"
Filename="36595~GOS0000007120.wav" PlaybackMode="OnAir"
FileMD5="CD3E524CF840898D151DEDB761AEB10E">
              <PointOfInterestMarker Position="0" Enabled="true" Duration="0.46"
Value="0" Details="" POIType="13"/>
              <PointOfInterestMarker Position="0" Enabled="true"
Duration="2.235" Value="0" Details="" POIType="6"/>
              <PointOfInterestMarker Position="0" Enabled="true"
Duration="11.734" Value="0" Details="" POIType="6"/>
              <PointOfInterestMarker Position="0" Enabled="true"
Duration="29.357" Value="0" Details="" POIType="6"/>
              <PointOfInterestMarker Position="66.187" Enabled="false"
Duration="31.304" Value="0" POIType="2"/>
              <PointOfInterestMarker Position="227.591" Enabled="true"
Duration="0" Value="2.1918" Details="" POIType="8"/>
              <PointOfInterestMarker Position="227.804" Enabled="true"
Duration="1.9788" Value="0" Details="" POIType="14"/>
              <VolumeMarker Position="0" Level="1"/>
              <VolumeMarker Position="229.7828" Level="1"/>
            </Transition>
          </Resource>
          <Artist ArtistID="2" ArtistTypeID="0" Name="Nelly"/>
          <Album AlbumID="10644" Name="Hot In Herre [Single] [Europe]"/>
          <AssetAttribute AttributeTypeID="1" AttributeTypeName="Mood"
AttributeValueID="4" AttributeValueName="4 - Feel Amazing "/>
          <AssetAttribute AttributeTypeID="7" AttributeTypeName="Sound Code"
AttributeValueName="RAP "/>
          <StationSpecific StationSpecificId="693768"
UniversalIdentifier="7103d17a-50aa-4219-90e7-256ff30f1411" IsActive="true"
Station="[SYD] 104.1 2Day FM" StationID="3"
StationUniversalIdentifier="18554857-15fe-4754-b52e-30d127d2c324">
            <Category CategoryID="2383" Name="21" AssetTypeID="1"
```

```
CategoryGroup="21 - 96-03" LongName="21 - 96-03"/>
          </StationSpecific>
        </Asset>
      </AssetEvent>
    </LogEvent>

  </LogEvents>
</ZettaClipboard>
```