

**Oracle Utilities Testing Accelerator
Reference Guide for Oracle Utilities Mobile
Workforce Management/ Oracle Real-Time
Scheduler v2.3.0.3**

Release 6.0.0.0.1

E96258-01

February 2019

E96258-01

Copyright © 2000, 2019 Oracle and/or its affiliates. All rights reserved.

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	i-i
Audience	i-ii
Related Documents	i-ii
Conventions.....	i-ii
Abbreviations	i-ii
Chapter 1	
Component Reference	1-1
Overview	1-2
Components	1-2
Chapter 2	
Function Library Reference	2-1
OUMWMCONNECTEDMCPLIB.....	2-2
OUMWMMCPLIB	2-6
OUMWMHYBRIDLIB	2-11
Chapter 3	
Sample Work Flows	3-1
Sample Flows.....	3-2
Non-MDT Flow	3-2
MDT Flow	3-3
M2 Non-MDT Flow.....	3-3
MDT Flow Using NextGen MCP.....	3-4
MDT Flow Using HybridMCP.....	3-4
MWM_Cloud_Environment_Sanity	3-5
M2 Android-MDT Sanity Flow	3-6
M2 iOS-MDT Sanity Flow	3-13
Executing Sample Flows.....	3-17
Pre-requisites.....	3-17
Setting Up Sample Flows	3-17
Appendix A	
Inbound Web Services	A-1

Preface

Welcome to the Oracle Utilities Testing Accelerator Reference Guide for Oracle Utilities Mobile Workforce Management/ Oracle Real-Time Scheduler v2.3.0.3.

The guide describes the Oracle Utilities Mobile Workforce Management (MWM) v2.3.0.3 and Oracle Real-Time Scheduler (ORS) v2.3.0.3 components and the function libraries used to create those components in Oracle Utilities Testing Accelerator (OUTA) v6.0.0.0.1.

This preface includes the following sections:

- [Audience](#)
- [Related Documents](#)
- [Conventions](#)
- [Abbreviations](#)

Audience

This guide is intended for QA/Test Engineers and Automation Developers to understand the various components and libraries available for them to automate the business test flows for Oracle Utilities Mobile Workforce Management using Oracle Utilities Testing Accelerator for Oracle Utilities Mobile Workforce Management and Oracle Real-Time Scheduler.

Related Documents

For more information, refer to the following Oracle resources.

Release Notes

- *Oracle Utilities Testing Accelerator Release Notes*

Installation and Administration Guide

- *Oracle Utilities Testing Accelerator Installation and Administration Guide*

User and Reference Guides

- *Oracle Utilities Testing Accelerator User's Guide*

See also:

- Oracle Utilities Mobile Workforce Management Documentation Library
- Oracle Real-Time Scheduler Documentation Library

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.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Abbreviations

The following terms are used in this document:

Term	Expanded Form
OUTA	Oracle Utilities Testing Accelerator
MWM	Oracle Utilities Mobile Workforce Management
ORS	Oracle Real-Time Scheduler

Chapter 1

Component Reference

This chapter lists the Oracle Utilities Mobile Workforce Management/Oracle Real-Time Scheduler starter components available to create flows in Oracle Flow Builder for testing the Oracle Utilities Mobile Workforce Management/Oracle Real-Time Scheduler application.

The chapter includes the following sections:

- [Overview](#)
- [Components](#)

Overview

Oracle Utilities Testing Accelerator for Oracle Utilities Mobile Workforce Management/ Oracle Real-Time Scheduler is a test starter pack built on top of Oracle Utilities Testing Accelerator that generates test automation scripts using Oracle Flow Builder.

Oracle Utilities Testing Accelerator for Oracle Utilities Mobile Workforce Management/Oracle Real-Time Scheduler contains out-of-the-box product-specific components used to build new test flows in Oracle Flow Builder to test the Oracle Utilities Mobile Workforce Management/Oracle Real-Time Scheduler applications. These out-of-the-box components correspond to specific business entities, such as business objects, service scripts, or business services used for interfacing with the application. These components can be used as available or can be extended. Create new components to be used to create flows. This starter pack also contains a set of function libraries that can be used for creating custom components.

For detailed information about using these function libraries, refer to [Chapter 2: Function Library Reference](#).

Consider this pack to be a starter kit which can be expanded and built upon. A few sample flows are included as examples.

For information about creating components and flows, refer to the *Oracle Utilities Testing Accelerator User's Guide*.

Components

This section lists the starter components available in Oracle Utilities Mobile Workforce Management.

Pre-requisites: The Inbound Web Service service using the respective business object should be available in the application.

Additional Notes: Failure while creating, reading, or updating the component is logged in the test execution report, thus facilitating debugging/analysis of the problem.

The components are categorized under the following functional areas:

- [Administration Data](#)
- [Connected MCP Processing](#)
- [Contractor Portal](#)
- [Resource Management](#)
- [Scheduling](#)
- [Service Management](#)
- [M2 Activities](#)
- [Hybrid MCP Processing](#)

Administration Data

Component	Description
M1-ActivityType	<p>Used to create, read, or update M1 Activity Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ActivityType business object.</p>
M1-AdditionalUserInfo	<p>Used to maintain additional user information as part of the miscellaneous section on the user portal via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1- AdditionalUserInfo business object.</p>
M1-AlertType	<p>Used to create, read, or update an M1 Alert Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-AlertType business object.</p>
M1-AppointmentBookingGroup	<p>Used to create, read, or update an M1 Appointment Booking Group via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-AppointmentBookingGroup business object.</p>
M1-BreakTaskType	<p>Used to create, read, or update an M1 Break Task Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-BreakTaskType business object.</p>
M1-CapacityType	<p>Used to create, read, or update an M1 Capacity Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CapacityType business object.</p>
M1-ComplexActivityType	<p>Used to create, read, or update an M1 Complex Activity Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ComplexActivityType business object.</p>
M1-CrewHierarchy	<p>Used to create, read, or update an M1-CrewHierarchy via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CrewHierarchy business object.</p>

Component	Description
M1-ContractorType	<p>Used to create, read, or update an M1 Contractor Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ContractorType business object.</p>
M1-CrewShiftType	<p>Used to create, read, or update an M1 Crew Shift Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CrewShiftType business object.</p>
M1-CrewType	<p>Used to create, read, or update an M1 Crew Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CrewType business object.</p>
M1-Deployment	<p>Used to activate or deactivate a deployment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Deployment business object.</p>
M1-DeploymentPart	<p>Used to create, read, or update a deployment part via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DeploymentPart business object.</p>
ATM1MWM2CCBSYNCRQ	Used to change the sync request status to 'send'.
ATM1RetrieveSyncRqId	Used to retrieve the sync request ID request.
M1-DeploymentType	<p>Used to create, read, or update a deployment type via a web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DeploymentType business object.</p>
M1-DeploymentTypeRestfulSvc	<p>Used to create and describe the structure and rules applicable to deployment type definitions supporting the JSON based mobile platform.</p> <p>After creation, the CRUD operations can be performed through these components against the M1- DeploymentTypeRestfulSvc business object.</p>

Component	Description
M1-Depot	<p>Used to create, read, or update a depot via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Depot business object.</p>
M1-DepotProfile	<p>Used to create, read, or update a depot profile via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DepotProfile business object.</p>
M1-DepotTaskType	<p>Used to create, read, or update a depot task type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DepotTaskType business object.</p>
M1-DispatchArea	<p>Used to create, read, or update an M1 Dispatch Area via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DispatchArea business object.</p>
M1-DispatcherType	<p>Used to create, read, or update an M1 Dispatcher Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DispatcherType business object.</p>
M1-Equipment	<p>Used to create, read, or update an M1 Equipment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Equipment business object.</p>
M1-GeographicArea	<p>Used to create, read, or update an M1 Geographic Area via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-GeographicArea business object.</p>
M1-GlobalConfig	<p>Used to create, read, or update M1 Global Configurations via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-GlobalConfigurations business object.</p>

Component	Description
M1-KPI	<p>Used to create, read, or update an M1 KPI via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-KPI business object.</p>
M1-Location	<p>Used to create, read, or update an M1 Location via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Location business object.</p>
M1-MobileDeviceTerminal	<p>Used to create, read, or update an M1 Mobile Device Terminal via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-MobileDeviceTerminal business object.</p>
M1-MobileWorkerType	<p>Used to create, read, or update an M1 Mobile Worker Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-MobileWorkerType business object.</p>
M1-MobileDeviceTerminalType	<p>Used to create, read, or update an M1 MobileDeviceTerminalType via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-MobileDeviceTerminalType business object.</p>
M1-NPTTaskType	<p>Used to create, read, or update an M1 Non Prd Task Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-NonPrdTaskType business object.</p>
M1-POUTaskType	<p>Used to create, read, or update an M1 POU Task Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-POUTaskType business object.</p>
M1-POUType	<p>Used to create, read, or update an M1 POU Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-POUType business object.</p>

Component	Description
M1-PriorityProfile	<p>Used to create, read, or update an M1 Priority Profile via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-PriorityProfile business object.</p>
M1-ProcedureType	<p>Used to create, read, or update an M1 Procedure Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ProcedureType business object.</p>
M1-RemarkType	<p>Used to create, read, or update an M1 Remark Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-RemarkType business object.</p>
M1-ResTimesheetType	<p>Used to create, read, or update an M1 Resource TimesheetType. via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Resource TimesheetType business object.</p>
M1-Scheduler	<p>Used to create, read, or update an M1 Scheduler via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Scheduler business object.</p>
M1-SchedulerArea	<p>Used to create, read, or update an M1 Scheduler Area via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-SchedulerArea business object.</p>
M1-SchedulerConfig	<p>Used to create, read, or update an M1 Scheduler Config via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-SchedulerConfig business object.</p>
M1-ServiceArea	<p>Used to create, read, or update an M1 Service Area via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ServiceArea business object.</p>

Component	Description
M1-ServiceAreaHierarchy	<p>Used to create, read, or update an M1 Service Area Hierarchy via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ServiceAreaHierarchy business object.</p>
M1-ServiceClass	<p>Used to create, read, or update an M1 Service Class via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ServiceClass business object.</p>
M1-ShiftCostProfile	<p>Used to create, read, or update an M1 Shift Cost Profile via a Web service.</p> <p>The CRUD operations can be performed through these components against the M1-ShiftCostProfile business object.</p>
M1-SpeedProfileGeoArea	<p>Used to create, read, or update an M1-SpeedProfileGeoArea via a Web service.</p> <p>The CRUD operations can be performed through these components against the M1-SpeedProfileGeoArea business object.</p>
M1-Skill	<p>Used to create, read, or update an M1 Skill via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Skill business object.</p>
M1-StatusReason	<p>Used to create, read, or update an M1 Status Reason via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-StatusReason business object.</p>
M1-WorkCalendar	<p>Used to create, read, or update an M1-WorkCalendar via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-WorkCalendar business object.</p>
M1-VehicleType	<p>Used to create, read, or update an M1 Vehicle Type via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-VehicleType business object.</p>

Component	Description
M1-WorkProfile	<p>Used to create, read, or update an M1 Work Profile via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-WorkProfile business object.</p>
M1-WorkerAttribute	<p>Used to create, read, or update an M1 WorkerAttribute via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-WorkerAttribute business object.</p>
M1_BatchExecuteBS	<p>Used to create, read, or update an M1 BatchExecuteBS via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-BatchExecuteBS business object.</p>
M1-AttachmentSizeClass	<p>Used to create, read, or update an M1-AttachmentSizeClass via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-AttachmentSizeClass business object.</p>

Connected MCP Processing

Component	Description
ConnectedMCP_CompleteActivity	<p>Performs the actions of a field worker for completing an M1 Activity using Connected MCP.</p> <p>Note: To be called only after calling the ConnectedMCP_StartActivity component.</p>
ConnectedMCP_CompleteBreak	<p>Performs the actions of a field worker for completing an M1 Break Task using Connected MCP.</p> <p>Note: To be called only after calling the ConnectedMCP_StartBreak component.</p>
ConnectedMCP_CompleteNPT	<p>Performs the actions of a field worker for completing an M1 Non Productive Task using Connected MCP.</p> <p>Note: To be called only after calling the ConnectedMCP_StartNPT component.</p>

Component	Description
ConnectedMCP_CompletePOU	<p>Performs the actions of a field worker for completing an M1 POU task using Connected MCP.</p> <p>Note: To be called only after calling the ConnectedMCP_StartPOU component.</p>
ConnectedMCP_CompleteShift	<p>Performs the actions of a field worker for completing a shift using Connected MCP.</p> <p>Note: To be called only after a shift is in 'start' status.</p>
ConnectedMCP_DispatchedTask	<p>Performs the actions of a field worker to check if a task has been dispatched using Connected MCP. Then, click it.</p>
ConnectedMCP_EnrouteActivity	<p>Performs the actions of a field worker to move the status of a task to 'En-route' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_DispatchedTask component.</p>
ConnectedMCP_EnrouteNPT	<p>Performs the actions of a field worker to move the status of an NPT task to 'En-route' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_DispatchedTask component.</p>
ConnectedMCP_EnroutePOU	<p>Performs the actions of a field worker to move the status of an NPT task to 'En-route' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_DispatchedTask component.</p>
ConnectedMCP_StartActivity	<p>Performs the actions of a field worker to move the status of an activity to 'Started' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_EnrouteActivity component.</p>
ConnectedMCP_StartBreak	<p>Performs the actions of a field worker to move the status of a break task to 'Started' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_DispatchedTask component.</p>
ConnectedMCP_StartCrewShift	<p>Performs the actions of a field worker to start a crew shift using Connected MCP. Then, click it.</p>

Component	Description
ConnectedMCP_StartNPT	<p>Performs the actions of a field worker to move the status of an activity to 'Started' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_ EnrouteNPT component.</p>
ConnectedMCP_StartPOU	<p>Performs the actions of a field worker to move the status of a POU to 'Started' using Connected MCP. Then, click it.</p> <p>Note: To be called only after calling the ConnectedMCP_ EnroutePOU component.</p>

Contractor Portal

Component	Description
M1-Capacity	<p>Used to create, read, or update an M1 Capacity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Capacity business object.</p>
M1-CapacityTemplate	<p>Used to create, read, or update an M1 Capacity Template via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CapacityTemplate business object.</p>
M1-CapacityWeeklyTemplate	<p>Used to create, read, or update an M1-CapacityWeeklyTemplate via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CapacityWeeklyTemplate business object.</p>
M1-Company	<p>Used to create, read, or update an M1 Company via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Company business object.</p>
M1-Contractor	<p>Used to create, read, or update an M1 Contractor via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Contractor business object.</p>
M1-ContractorEligibility	<p>Used to create, read, or update an M1 Contractor Eligibility via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ContractorEligibility business object.</p>

Resource Management

Component	Description
M1-BreakTask	<p>Used to create, read, or update an M1 Break Task via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-BreakTask business object.</p>

Component	Description
M1-CommonShiftWeeklyTemplate	<p>Used to create, read, or update an M1 Common Shift Weekly Template via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CommonShiftWeeklyTemplate business object.</p>
M1-CrewShftCloseOpenAllocate	<p>Used to close or open a shift using M1-CrShfClAl via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CrShfClAl service script.</p>
M1-CrewShift	<p>Used to create, read, or update an M1 Crew Shift via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CrewShift business object.</p>
M1-CrewShiftTemplate	<p>Used to create, read, or update an M1 Crew Shift Template via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-CrewShiftTemplate business object.</p>
M1-DeactivateMobileWorker	<p>Used to deactivate an active mobile worker via a Web service.</p> <p>After creation, the mobile worker can be deactivated. The business object used for this component is M1-MobileWorker.</p>
M1-DeactivateVehicle	<p>Used to deactivate an active vehicle via a Web service.</p> <p>After creation, the vehicle can be deactivated. The business object used for this component is M1-Vehicle.</p>
M1-DeactivateCrew	<p>Used to deactivate an active crew via a Web service.</p> <p>After creation, crew can be deactivated. The business object used for this component is M1-Crew.</p>
M1-DeleteResourceLeavePOU	Used for deleting M1 Resource Leave POU
M1-DepotRelatedShift	<p>Used to create, read, or update an M1 Depot Related Shift via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DepotRelatedShift business object.</p>

Component	Description
M1-DepotSinglePersonCrew	<p>Used to create, read, or update an M1 Depot Single Person Crew via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DepotSinglePersonCrew business object.</p>
M1-Dispatcher	<p>Used to create, read, or update an M1 Dispatcher via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Dispatcher business object.</p>
M1-DispatcherShift	<p>Used to create, read, or update an M1 Dispatcher Shift via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DispatcherShift business object.</p>
M1-DispatcherShiftLite	<p>Used to retrieve dispatcher shift information when we only need base elements.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-DispatcherShiftLite business object.</p>
M1-ForceLogOffCrewShift	Used to force logging off an MDT crew shift
M1-GetShiftCapabilities	Used to retrieve the capabilities of a crew shift
M1-HazardArea	<p>Used to create, read, or update an M1 Hazard Area via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-HazardArea business object.</p>
M1-MailingList	<p>Used to create, read, or update an M1 Mailing List via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-MailingList business object.</p>
M1-MainMail	Used to create M1 MainMail
M1-MobileWorker	<p>Used to create, read, or update an M1 Mobile Worker via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-MobileWorker business object.</p>

Component	Description
M1-NonProductiveTask	<p>Used to create, read, or update an M1 NonPrdTask via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-NonPrdTask business object.</p>
M1-POUTask	<p>Used to create, read, or update an M1 POU Task via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-POUTask business object.</p>
M1-RealPOU	<p>Used to create, read, or update an M1 Real POU via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-RealPOU business object.</p>
M1-ResourceTimesheet	<p>Used to create, read, or update an M1 Res Time Sheet via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ResTimesheet business object.</p>
M1-Resourceleave	<p>Used to create, read, or update an M1 Resource Leave via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ResourceLeave business object.</p>
M1-ResourceLite	<p>Used to retrieve dispatcher information when we only need base elements, however it also includes other elements which may only be relevant for other Resource business objects (Crews, Mobile Workers, Vehicles, etc).</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ResourceLite business object.</p>
M1-ShiftWeeklyTemplate	<p>Used to create, read, or update an M1 Shift Weekly Template via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-ShiftWeeklyTemplate business object.</p>

Component	Description
M1-SimpleProcedure	<p>Used to create, read, or update an M1 Simple Procedure via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-SimpleProcedure business object.</p>
M1-SinglePersonCrew	<p>Used to create, read, or update an M1 Single Person Crew via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-SinglePersonCrew business object.</p>
M1-SubscriptionWeeklyTemplate	<p>Used to create, read, or update an M1 Subscription Weekly Template via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-SubscriptionWeeklyTemp business object.</p>
M1-TemplateMail	<p>Used to create, read, or update an M1 Template Mail via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-TemplateMail business object.</p>
M1-TemplatePOU	<p>Used to create, read, or update an M1 Template POU via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-TemplatePOU business object.</p>
M1-Vehicle	<p>Used to create, read, or update an M1 Vehicle via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-Vehicle business object.</p>
M1-VehicleLeaveArea	<p>Used to create, read, or update an M1 Vehicle Leave Area via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-VehicleLeaveArea business object.</p>
M1-getPOUTaskIdForPOUId	Used to retrieve the POU task ID (the task scheduled to a crew shift) for a given real POU ID.
M1-getdeactivateMWId	Used to retrieve the mobile worker ID for the given user whose BO status is 'Active'.
M1-getdeactivateVehicleId	Used to retrieve the vehicle ID for the given user whose BO status is 'Active'.

Component	Description
M1-getdeactivateCrewId	Used to retrieve the crew ID for the given user whose BO status is 'Active'.

Scheduling

Component	Description
M1-AllocateActivityToAShift	Used to manually allocate an activity to a crew shift.
M1-AppointmentBookingRequest	Used to trigger an M1 Appointment Booking Request.
M1-CheckIfActivityIsScheduled	Verifies if an activity is scheduled to a crew shift.
M1-DailySpeedTemplate	Used to create, read, or update an M1 Daily Speed Template via a Web service. After creation, the CRUD operations can be performed through these components against the M1-DailySpeedTemplate business object.
ATM2ManualAssignActtoShift	Used to assign activity to the shift.
M1-FixToCrew	Used to fix an activity to be scheduled to a particular crew
M1-SchedulerRegistryRead	Used to retrieve the scheduler registry details for an entity
M1-SpeedProfileTemplate	Used to create, read, or update an M1 Speed Profile Template via a Web service. After creation, the CRUD operations can be performed through these components against the M1-SpeedProfileTemplate business object.
M1-TaskScheduleDetails	Used for reading an M1 Task Schedule entity.
M1-UnAssignActivity	Used to unassign an activity from the crew shift.
M1-WaitForTime	Used to wait for certain amount of time. The unit of measure is minutes.

Service Management

Component	Description
M1-Activity	Used to create, read, or update an M1 Activity via a Web service. After creation, the CRUD operations can be performed through these components against the M1-Activity business object.

Component	Description
M1-Assignment	Used to create, read, or update an M1 Assignment via a Web service. After creation, the CRUD operations can be performed through these components against the M1-Assignment business object.
M1-ComplexActivity	Used to create, read, or update an M1 Complex Activity via a Web service. After creation, the CRUD operations can be performed through these components against the M1-ComplexActivity business object.
M1-DeferActivity	Used to defer an activity
M1-DepotRelatedActivity	Used to create, read, or update an M1 Depot Related Activity via a Web service. After creation, the CRUD operations can be performed through these components against the M1-DepotRelatedActivity business object.
M1-DepotTask	Used to create, read, or update an M1 Depot Task via a Web service. After creation, the CRUD operations can be performed through these components against the M1-DepotTask business object.
M1-GetAssignmentIdForTaskId	Used to retrieve the assignment ID of a dispatched task
M1-GetDepotTaskId	Used to retrieve the depot task ID associated to a given depot related or a pre-requisite activity.
M1-GetDeploymentId	Used to retrieve the deployment ID for the given deployment type.
M1-OverrideTimeWindow	Used to create, read, or update an M1 Override Time Window via a Web service. After creation, the CRUD operations can be performed through these components against the M1-OverrideTimeWindow business object.
M1-ActivityCommitByCont	Used to commit the work done by the contractors via a Web service.
M1-ActivityCompleteByCont	Used to complete the work done by Contractors via Web service.
M1-DepotTransferActivityLoad	Used to create the depot transfer activity load via Web service.
M1-DepotTransferActivityUnload	Used to create the depot transfer activity un-load via Web service.

Component	Description
M1-DepotTransferAssignmentLoad	Used to create the depot transfer assignment load via Web service.
M1-DepotTransferAssignmentUnld	Used to create the depot transfer assignment un-load via Web service.
M1-LoadCategory	Used to create the load category via Web service.
M1-Bay	Used to create the bays via Web service.
M1-DepotTransferActivityType	Used to create the depot transfer activity type via Web service.
M1-DepotTransferActivity	Used to create the depot transfer activity via Web service.
M1-DepotTransferShiftType	Used to create the depot transfer shift type via Web service.
M1-DepotTransferShift	Used to create the depot transfer shift via Web service.

M2 Activities

Component	Description
M2-InstallMeterActivity	<p>Used to create, read, or update an M2 Install Meter Activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M1-InstallMeterActivity business object.</p>
M2-InstallMeterAssignment	<p>Used to create, read, or update an M2 Install Meter Assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-InstallMeterAssignment business object.</p>
M2-BasicItemActivity	<p>Used to create, read, or update an M2 Basic Item Activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-BasicItemActivity business object.</p>
M2-BasicItemAssignment	<p>Used to create, read, or update a basic item assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-BasicItemAssignment business object.</p>

Component	Description
M2- ConnectSPActivity	<p>Used to create, read, or update an M2 Connect Service Point Activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- ConnectSPActivity business object.</p>
M2-ConnectSPAssignment	<p>Used to create, read, or update a connect service point assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-ConnectSPAssignment business object.</p>
M2-CutNonPayItemActivity	<p>Used to create, read, or update an M2 Cut For Non Payment Item Activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- CutNonPayItemActivity business object.</p>
M2-CutNonPayItemAssignment	<p>Used to create, read, or update a cut for non payment item assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-CutNonPayItemAssignment business object.</p>
M2-DisconnectMeterActivity	<p>Used to create, read, or update a disconnect meter activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- DisconnectMeterActivity business object.</p>
M2-DisconnectMeterAssignment	<p>Used to create, read, or update a disconnect meter assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-DisconnectMeterAssignment business object.</p>
M2-DisconnectSPActivity	<p>Used to create, read, or update a disconnect service point activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-DisconnectSPActivity business object.</p>
M2-DisconnectSPAssignment	<p>Used to create, read, or update a disconnect service point assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- DisconnectSPAssignment business object.</p>

Component	Description
M2-ExchangeItemActivity	<p>Used to create, read, or update an exchange item activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- ExchangeItemActivity business object.</p>
M2-ExchangeItemAssignment	<p>Used to create, read, or update an exchange item assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- ExchangeItemAssignment business object.</p>
M2-InstallItemActivity	<p>Used to create, read, or update an install item activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- InstallItemActivity business object.</p>
M2-InstallItemAssignment	<p>Used to create, read, or update an install item assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-InstallItemAssignment business object.</p>
M2-MeterReadActivity	<p>Used to create, read, or update a meter read activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- MeterReadActivity business object.</p>
M2-MeterReadAssignment	<p>Used to create, read, or update a meter read assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-MeterReadAssignment business object.</p>
M2-RemoveItemActivity	<p>Used to create, read, or update a remove item activity via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2- RemoveItemActivity business object.</p>
M2-RemoveItemAssignment	<p>Used to create, read, or update a remove item assignment via a Web service.</p> <p>After creation, the CRUD operations can be performed through these components against the M2-RemoveItemAssignment business object.</p>

Component	Description
M2-RemoveMeterActivity	Used to create, read, or update a remove meter activity via a Web service. After creation, the CRUD operations can be performed through these components against the M2- RemoveMeterActivity business object.
M2-RemoveMeterAssignment	Used to create, read, or update a remove meter assignment via a Web service. After creation, the CRUD operations can be performed through these components against the M2-RemoveMeterAssignment business object.
M2-MaintUtilityActByHost	Used to create an M2 Activity from an external system

Hybrid MCP Processing

Component	Description
HybridMCP_StartShift	Performs the actions of a field worker to start a crew shift using HybridMCP. Then, clicks it.
HybridMCP_TaskDispatched	Verifies whether task (activity, break, NPT, or POU) is dispatched or not. Then, clicks it from the Task List screen.
HybridMCP_EnrouteActivity	Performs the actions of a field worker to move the status of a task to 'En-route' using HybridMCP. Then, clicks it. To be called only after calling the HybridMCP_DispatchedTask component.
HybridMCP_CompleteActivity_InstallMeter	Completes the Install Meter activity.
HybridMCP_CompleteActivity_RemoveMeter	Completes the Remove Meter activity.
HybridMCP_addUnRelatedPickUp Activity	Creates unrelated pick activity using appium.
HybridMCP_activityWithRelated PickUp	Creates related pick up activity using appium.
HybridMCP_StartActivity	Performs the actions of a field worker to move the status of an activity to 'Started' using HybridMCP. Then, clicks it. To be called only after calling the HybridMCP_EnrouteActivity component.

Component	Description
HybridMCP_CompleteActivity	<p>Performs the actions of a field worker for completing an M1 Activity using HybridMCP.</p> <p>To be called only after calling the HybridMCP_StartActivity component.</p>
HybridMCP_StartBreak	<p>Performs the actions of a field worker to move the status of a break task to 'Started' using HybridMCP. Then, clicks it.</p> <p>To be called only after calling the HybridMCP_DispatchedTask component.</p>
HybridMCP_CompleteBreak	<p>Performs the actions of a field worker for completing an M1 Break Task using HybridMCP.</p> <p>To be called only after calling the HybridMCP_StartBreak component.</p>
HybridMCP_EnrouteNPT	<p>Performs the actions of a field worker to move the status of an NPT task to 'En-route' using HybridMCP. Then, clicks it.</p> <p>To be called only after calling the HybridMCP_DispatchedTask component.</p>
HybridMCP_StartNPT	<p>Performs the actions of a field worker to move the status of an activity to 'Started' using HybridMCP. Then, clicks it.</p> <p>To be called only after calling the HybridMCP_EnrouteNPT component.</p>
HybridMCP_CompleteNPT	<p>Performs the actions of a field worker for completing an M1 Non Productive Task using HybridMCP.</p> <p>To be called only after calling the HybridMCP_StartNPT component.</p>
HybridMCP_EnroutePOU	<p>Performs the actions of a field worker to move the status of an NPT task to 'En-route' using HybridMCP. Then, clicks it.</p> <p>To be called only after calling the HybridMCP_DispatchedTask component.</p>
HybridMCP_StartPOU	<p>Performs the actions of a field worker to move the status of a POU to 'Started' using HybridMCP. Then, clicks it.</p> <p>To be called only after calling the HybridMCP_EnroutePOU component.</p>

Component	Description
HybridMCP_CompletePOU	<p>Performs the actions of a field worker for completing an M1 POU task using ConnectedMCP.</p> <p>To be called only after calling the HybridMCP_StartPOU component.</p>
HybridMCP_CompleteShift	<p>Performs the actions of a field worker for completing a shift using HybridMCP.</p> <p>To be called only after a shift is in 'start' status.</p>
HybridMCP_getCurrentTime	<p>Returns the current date and time. It is designed for entering the current date and time value in the MDT Tag while logging into the HybridMCP Shift.</p>
Webdriver_AndroidLogin	<p>Performs the crew shift login on an Android emulator or android device.</p> <p>To be called only after configuring an Android device or emulator. More details can be found on Sample flows Section for Android flow.</p>
Webdriver_completeBreak	<p>Performs the actions of a field worker for completing an M1 Break Task using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_startBreak component.</p>
Webdriver_completeM1Activity	<p>Performs the actions of a field worker for completing an M1 Activity using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_startM1Activity component.</p>
Webdriver_completeM2Activity	<p>Performs the actions of a field worker for completing an M2 Activity using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_startM2Activity component.</p>
Webdriver_completeNPT	<p>Performs the actions of a field worker for completing an Non Productive Task using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_startNPTcomponent.</p>
Webdriver_completePOU	<p>Performs the actions of a field worker for completing an M1 POU task using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_completePOU component.</p>

Component	Description
Webdriver_endShift	<p>Performs the actions of a field worker for completing a shift using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after a shift is in 'start' status.</p>
Webdriver_gotoTaskHome	<p>Performs the actions of a field worker for navigating to the task list home screen using Web Driver on an Android/iOS emulator or device.</p>
Webdriver_iOSLogin	<p>Performs the crew shift login on an iOS emulator.</p> <p>To be called only after configuring an iOS emulator. More details can be found on Sample flows Section for iOS flow.</p>
Webdriver_selectDeployment	<p>Performs the actions of a field worker for selecting the active deployment before the crew shift starts, using Web Driver on an Android/iOS emulator or device.</p>
Webdriver_startBreak	<p>Performs the actions of a field worker to move the status of a break task to 'Started' using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_waitForTaskDispatch component.</p>
Webdriver_startM1Activity	<p>Performs the actions of a field worker to move the status of an M1activity to 'Started' using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_startM1Activity component.</p>
Webdriver_startM2Activity	<p>Performs the actions of a field worker to move the status of an M2activity to 'Started' using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_startM2Activity component.</p>
Webdriver_startNPT	<p>Performs the actions of a field worker to move the status of the NPT to 'Started' using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_waitForTaskDispatch component.</p>
Webdriver_startPOU	<p>Performs the actions of a field worker to move the status of the POU to 'Started' using Web Driver on an Android/iOS emulator or device.</p> <p>To be called only after calling the Webdriver_waitForTaskDispatch component.</p>

Component	Description
Webdriver_startShift	Performs the actions of a field worker to move the status of the Crew shift to 'Started' using Web Driver on an Android/iOS emulator or device.
Webdriver_waitForTaskDispatch	Performs the actions of a field worker to verify all the assigned tasks are dispatched in the MDT using Web Driver on an Android/iOS emulator or device.

Chapter 2

Function Library Reference

This chapter lists the Oracle Utilities Mobile Workforce Management function libraries and functions available to create components and flows in Oracle Utilities Testing Accelerator for testing the Oracle Utilities Mobile Workforce Management application.

- [OUMWMCONNECTEDMCPLIB](#)
- [OUMWMMCLIB](#)
- [OUMWMHYBRIDLIB](#)

OUMWMCONNECTEDMCPLIB

The OUMWMCONNECTEDMCPLIB library contains functions that have been/can be used for creating Oracle Utilities Mobile Workforce Management components used to create the Oracle Utilities Mobile Workforce Management flows.

The library includes the Oracle Utilities Mobile Workforce Management specific functions that use the underlying framework of core functions. These functions interface with the database and read information, such as scheduling information of tasks, etc.

The library uses the UI element's xpath values specified in the MWM_ConnectedMCP_REPOSITORY.properties file in the **etc** directory of the repository.

Note: The private functions in this library are used to implement the public functions. The private functions cannot be used directly in the Oracle Flow Builder scripts.

invokeApp

Launches the Web browser, navigates to the URL provided, and then checks for the object in the Web page.

If the object is found, the function returns “Y”, else returns “N”.

Example:

```
loginToMDT(String <URL>,String <objectToCheck>)
```

Input Parameters: String, String

Return Type: String

loginToMDT

Internally calls the [invokeApp](#) function with the MCP URL that is connected.

This function inputs the User ID and password, and logs into the MCP application that is connected. It returns “Y” on the successful login, else returns “N”.

If the URL parameter is specified as “NA”, the connected MCP URL is formulated using the application URL specified in the configuration.properties file by appending “/mobility/Login.jsp” to that URL. If the connected MCP URL is specified to this function, then it is used instead.

Example:

```
loginToMDT(String <username>, String <password>, String  
<connectedMCPURL>)
```

Input Parameters: String, String, String

Return Type: String

downloadDeployment

Internally calls the [loginToMDT](#) function and waits for the deployment to download.

This function verifies if the **Start Shift** page is displayed within a minute. If the page is displayed, it returns “Y”, else it returns “N”.

If the URL parameter is specified as “NA”, the connected MCP URL is formulated using the application URL specified in the configuration.properties file by appending “/mobility/Login.jsp” to that URL. If the connected MCP URL is specified to this function, then it is used instead.

Example:

```
downloadDeployment(String <username>, String <password>, String  
<connectedMCPURL>)
```

Input Parameters: String, String, String
Return Type: String

startShift

Internally calls the [downloadDeployment](#) function and waits for the crew shift to start.

This function verifies if the Open Task List page is displayed. If the page is displayed, it returns “Y”, else it returns “N”. The loginError should be set as “N” for the crew shift to start successfully.

The function also verifies the negative scenarios where an error message is expected to be displayed during the shift start. The error message verification can be triggered by setting the loginError flag as “Y” as the input. If the loginError is set as “Y”, the function checks for an error message to be displayed, returns the error message, and then logs off the crew shift. If no error message is retrieved, it then returns a blank string.

If the URL parameter is specified as “NA”, the connected MCP URL is formulated using the application URL specified in the configuration.properties file by appending “/mobility/Login.jsp” to that URL. If the connected MCP URL is specified to this function, then it is used instead.

Example:

```
startShift(String <username>, String <password>, String
<connectedMCPURL>, String <loginError>)
```

Input Parameters: String, String, String, String
Return Type: String

verifyTaskDispatched

Verifies if a task has been dispatched for a given Task ID and then clicks on the task to open the UI.

This function returns a “Y” if the task has been dispatched. It checks for a task to be dispatched to the connected MCP within five minutes. If it is not dispatched, it returns a “N”.

Example:

```
verifyTaskDispatched(String <tasked>)
```

Input Parameters: String
Return Type: String

enrouteActivity

Performs the actions of a field worker and moves the task to “Enroute” status using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign an activity, it has to be called after calling the [verifyTaskDispatched](#) function.

Example:

```
enrouteActivity ()
```

Input Parameters: <none>
Return Type: String

startActivityFromEnroute

Performs the actions of a field worker and moves the task to “Started” status using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign an activity, it has to be called after calling the [enrouteActivity](#) function.

Example:

```
startActivityFromEnroute ()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

completeActivity

Performs the actions of a field worker and moves the task to “Completed” status using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign an activity, it has to be called after calling the [startActivityFromEnroute](#) function.

Example:

```
completeActivity ()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

startBreak

Performs the actions of a field worker and moves the break task status to “Started” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a break task, it has to be called after calling the [verifyTaskDispatched](#) function.

Example:

```
startBreak ()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

completeBreak

Performs the actions of a field worker and moves the break task status to “Completed” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a break task, it has to be called after calling the [startBreak](#) function.

Example:

```
completedBreak ()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

enrouteNPT

Performs the actions of a field worker and moves the non productive task (NPT) status to “Enroute” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a non productive task, it has to be called after calling the [verifyTaskDispatched](#) function.

Example:

```
enrouteNPT ()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

startNPT

Performs the actions of a field worker and moves the non productive task (NPT) status to “Started” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a non productive task, it has to be called after calling the [enrouteNPT](#) function.

Example:

```
startNPT ()
```

Input Parameters: <none>

Return Type: String

enroutePOU

Performs the actions of a field worker and moves the period of unavailability (POU) task status to “Enroute” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a POU task, it has to be called after calling the [verifyTaskDispatched](#) function.

Example:

```
enroutePOU ()
```

Input Parameters: <none>

Return Type: String

startPOU

Performs the actions of a field worker and moves the period of unavailability (POU) task status to “Started” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a POU task, it has to be called after calling the [enroutePOU](#) function.

Example:

```
startPOU ()
```

Input Parameters: <none>

Return Type: String

completePOU

Performs the actions of a field worker and moves the period of unavailability (POU) task status to “Completed” using the connected MCP.

This function returns “Y” on success, else returns “N”. To assign a POU task, it has to be called after calling the [startPOU](#) function.

Example:

```
completePOU ()
```

Input Parameters: <none>

Return Type: String

endShift

Completes/ends the crew shift by performing the actions of a field worker to complete the current day’s crew shift using the connected MCP.

This function also closes the connected MCP application. It returns “Y” on success, else returns “N”.

Example:

```
endShift()
```

Input Parameters: <none>

Return Type: String

OUMWMMCLIB

The OUMWMMCLIB library contains functions that have been/can be used for creating Oracle Utilities Mobile Workforce Management components. These components are later used to create the Oracle Utilities Mobile Workforce Management flows.

The library includes the Oracle Utilities Mobile Workforce Management specific functions that use the underlying framework of core functions. These functions interface with the database and read information, such as scheduling information of tasks, etc.

The library uses the UI elements' xpath values specified in the MCP.properties file added as an asset to this functional library.

The library includes three new functions (hybridMCP_startShift, hybridMCP_verifyTaskDispatched, hybridMCP_endShift) to support the HybridMCP MDT flow.

Note: The private functions in this library are used to implement the public functions. The private functions cannot be used directly in the Oracle Flow Builder scripts.

startShift

Internally calls the downloadDeployment function and waits for the crew shift to start. This Function invokes the application in Chrome browser using the URL specified in config.properties file. The URL is provided as a value to the variable gStrNewMCPURL in config.properties file.

The function fetches this URL and invokes the application, and then logs in to the application fetching user credentials from config.properties file. On successful login, it checks for the available deployments. It selects the required deployment and waits for it to be downloaded.

Once the deployment is downloaded, it starts the crew shift and waits for the tasks to get dispatched in The **Open Tasks List** screen. If in case, the user tries to launch the application for an already started shift, it directly logs into the application and waits for the activities to get dispatched in the Open Task List screen. (No start shift step is performed as it is referring to already started shift.)

The function also verifies the negative scenarios where an error message is expected to be displayed when user login fails for any of the reasons, such as invalid user credentials, etc.

Example:

```
startShift(String <username>, String <password>, String HardwareId,
String <deploymentType>)
```

Input Parameters: String, String, String, String

Return Type: String

verifyTaskDispatched

Verifies if a task has been dispatched for a given Task ID for a specified amount of time maxTimeToCheck, and then clicks on the task in UI to perform actions on it.

This function returns a "Y" if the task has been dispatched. It checks for a task to be dispatched to the new MCP browser application within specified amount of time in minutes. If it is not dispatched, it returns a "N".

Example:

```
verifyTaskDispatched(String <taskId>,String maxTimeToCheck)
```

Input Parameters: String,String

Return Type: String

enrouteActivity

Performs the actions of a field worker and moves the task to "Enroute" status using the Next Gen MCP_ EnrouteActivity component.

This function returns "Y" on success, else returns "N". To assign an activity, it has to be called after calling the verifyTaskDispatched function.

Example:

```
enrouteActivity ()
```

Input Parameters: <none>

Return Type: String

startActivityFromEnroute

Performs the actions of a field worker and moves the task to "Started" status using the NextGenMCP_StartActivity component.

This function returns "Y" on success, else returns "N". To assign an activity, it has to be called after calling the enrouteActivity function.

Example:

```
startActivityFromEnroute ()
```

Input Parameters: <none>

Return Type: String

completeActivity

Performs the actions of a field worker and moves the task to "Completed" status using the Next Gen MCP_ CompleteActivity component

This function returns "Y" on success, else returns "N". To assign an activity, it has to be called after calling the startActivityFromEnroute function.

Example:

```
completeActivity ()
```

Input Parameters: <none>

Return Type: String

startBreak

Performs the actions of a field worker and moves the break task status to "Started" using the Next Gen MCP_ StartBreak component

This function returns "Y" on success, else returns "N". To assign a break task, it has to be called after calling the verifyTaskDispatched function.

Example:

```
startBreak ()
```

Input Parameters: <none>

Return Type: String

completeBreak

Performs the actions of a field worker and moves the break task status to "Completed" using the Next Gen MCP_CompleteBreak component

This function returns "Y" on success, else returns "N". To assign a break task, it has to be called after calling the startBreak function.

Example:

```
completedBreak ()
```

Input Parameters: <none>

Return Type: String

enrouteNPT

Performs the actions of a field worker and moves the non productive task (NPT) status to "Enroute" using the Next Gen MCP_EnrouteNPT component.

This function returns "Y" on success, else returns "N". To assign a non productive task, it has to be called after calling the verifyTaskDispatched function.

Example:

```
enrouteNPT ()
```

Input Parameters: <none>

Return Type: String

startNPT

Performs the actions of a field worker and moves the non productive task (NPT) status to "Started" using the Next Gen MCP_StartNPT Component

This function returns "Y" on success, else returns "N". To assign a non productive task, it has to be called after calling the enrouteNPT function.

Example:

```
startNPT ()
```

Input Parameters: <none>

Return Type: String

completeNPT

Performs the actions of a field worker and moves the period of unavailability (POU) task status to "Completed" using the Next Gen MCP_CompleteNPT component.

This function returns "Y" on success, else returns "N". To assign a POU task, it has to be called after calling the startPOU function.

Example:

```
completeNPT ()
```

Input Parameters: <none>

Return Type: String

enroutePOU

Performs the actions of a field worker and moves the period of unavailability (POU) task status to "Enroute" using the Next Gen MCP_EnroutePOU component.

This function returns "Y" on success, else returns "N". To assign a POU task, it has to be called after calling the verifyTaskDispatched function.

Example:

```
enroutePOU ()
```

Input Parameters: <none>

Return Type: String

startPOU

Performs the actions of a field worker and moves the period of unavailability (POU) task status to “Started” using the Next Gen MCP_StartPOU component.

This function returns “Y” on success, else returns “N”. To assign a POU task, it has to be called after calling the enroutePOU function.

Example:

```
startPOU ()
```

Input Parameters: <none>

Return Type: String

completePOU

Performs the actions of a field worker and moves the period of unavailability (POU) task status to “Completed” using the Next Gen MCP_CompletePOU component.

This function returns “Y” on success, else returns “N”. To assign a POU task, it has to be called after calling the startPOU function.

Example:

```
completePOU()
```

Input Parameters: <none>

Return Type: String

endShift

Completes/ends the crew shift by performing the actions of a field worker to complete the current day's crew shift using the Next Gen MCP_CompleteShift component.

This function also closes the Next Gen MCP application. It returns “Y” on success, else returns “N”.

Example:

```
endShift ()
```

Input Parameters: <none>

Return Type: String

CheckforAnActivityInCompletionScreen

Navigates to the **Activity Completion** screen and verifies if the activity with specified Activity ID is in completed state for a time specified as maxTimetoCheck in minutes.

If found, it returns “true”, else “false”.

Example:

```
CheckforAnActivityInCompletionScreen(String taskID, String  
maxTimetoCheck)
```

Input Parameters: <String,String>

Return Type: boolean

moveToOpenTasksAfterTaskComplete

Performs actions such as navigating back to open the task list screen from the **Activity Completion** screen. It returns TRUE on successful navigation to open the task list screen. Else, it returns false.

Example:

```
moveToOpenTasksAfterTaskComplete ()
```

Input Parameters: <none>

Return Type: Boolean

hybridMCP_startShift

Works similar to the [startShift](#) function. This function is designed to handle the Hybrid MCP browser, where it additionally enters the MDT tag. It internally calls the [downloadDeployment](#) function and waits for the crew shift to start. It invokes the application in the Google Chrome browser using the URL specified in config.properties file. The URL is provided as a value to the variable gStrNewMCPURL in the config.properties file.

This function fetches the URL and invokes the application, and then logs in to the application fetching user credentials from the config.properties file. On successful login, it checks for the available deployments. It selects the required deployment and waits for it to be downloaded.

After the deployment is downloaded, it starts the crew shift and waits for the tasks to get dispatched in **The Open Tasks List** screen. If in case, the user tries to launch the application for an already started shift, it directly logs into the application and waits for the activities to get dispatched in the **Open Task List** screen.(No start shift step is performed as it is referring to already started shift.)

The function also verifies the negative scenarios where an error message is expected to be displayed when user login fails for any of the reasons, such as invalid user credentials.

Example:

```
startShift(String <username>, String <password>, String HardwareId,  
String <MDTTag> String <deploymentType>)
```

Input Parameters: String, String, String, String, String.

Return Type: String

hybridMCP_verifyTaskDispatched

This function works similar to the [verifyTaskDispatched](#) function, except that it is designed to handle the HybridMCP browser. It verifies if a task has been dispatched for a given Task ID for a specified amount of time maxTimeToCheck, and then clicks on the task in the UI to perform actions on it.

This function returns a “Y” if the task has been dispatched. It checks for a task to be dispatched to the new MCP browser application within the specified time (in minutes). If it is not dispatched, it returns a “N”.

Example:

```
verifyTaskDispatched(String <taskId>,StringmaxTimeToCheck)
```

Input Parameters: String,String

Return Type: String

hybridMCP_endShift

This function works similar to the [verifyTaskDispatched](#) function, except that it is designed to handle HybridMCP browser. It completes/ends the crew shift by performing the actions of a field worker to complete the current day's crew shift using the HybridMCP_ CompleteShift component.

This function also closes the HybridMCP application. It returns “Y” on success, else returns “N”.

Example:

```
endShift ()
```

Input Parameters: <none>

Return Type: String

OUMWMHYBRIDLIB

The OUMWMHYBRIDLIB library includes functions that are/ can be used to create Oracle Utilities Mobile Workforce Management Mobile web driver related components. These components are later used to create the Oracle Utilities Mobile Workforce Management flows.

The library includes the Oracle Utilities Mobile Workforce Management web driver specific functions that use mobile specific activities.

Note: The private functions in this library are used to implement the public functions. The private functions cannot be used directly in the Oracle Utilities Testing Accelerator scripts.

AndroidMCP_startRemoteDriver

This function is used with the Android-MDT Flow. It is an initial step required to connect the android mobile device/ emulator that is running with the Appium server using the following capabilities:

- MWM Application Package Name
- MWM Application Activity Name
- Android OS Version(gStrAndroidOSVersion)
- Android Device Name(gStrAndroidDeviceName)
- Android Device UID(gStrAndroidDeviceUID)
- Android MWM Build Path(gStrAndroidBuildPath)
- Android Node URL(gStrAndroidMCPRemoteNodeURL)

Using these capabilities, the Appium server chooses the particular Android device/emulator for automation. The emulator should be running so that it automatically installs the MWM-debug build, located in the defined build-path folder. Finally, it automatically launches the Oracle Utilities Mobile Workforce Management application.

Note: All variables mentioned with the capabilities should be defined in configuration.properties file.

Example:

```
AndroidMCP_startRemoteDriver ()
```

Input Parameters: <none>

Return Type: String

iosMCP_startRemoteDriver

This method is used with the iOS-MDT Flow. This is the initial step required to connect the iOS mobile Simulator running with the Appium server (MAC) using the following capabilities:

- iOS X-Code Version(gStrIOSVersion)
- iOS Simulator Name(gStrIOSDeviceName)
- iOS Device UID(gStrIOSDeviceUID): <OPTIONAL>

- iOS Build Path(gStrIosBuildPath)
- iOS Node URL(gStrIosMCPRemoteNodeURL)

Using these capabilities, the Appium server chooses the particular iOS Simulator for automation. The Simulator automatically invokes; it automatically installs the MWM-debug build located in the defined build-path folder. Finally, it automatically launches the Oracle Utilities Mobile Workforce Management application.

Note: All variables mentioned with the capabilities should be defined in configuration.properties file.

Example:

```
IosMCP_startRemoteDriver()
```

Input Parameters: <none>
Return Type: String

MobileMCP_switchToDebugContextNodeURL

After the Oracle Utilities Mobile Workforce Management application is successfully launched, this method handles the Application Context-Mode and switches the context from Native-View to Debug-View (Web-View).

The method is used to handle the automation on OUML code-level, as both Android and IOS applications have same OUML code. So it is easier to create and maintain the same functional flow for both application types (Android / iOS).

Note: This method is common for both MDT flows.

Example:

```
MobileMCP_switchToDebugContextNodeURL()
```

Input Parameters: <none>
Return Type: String

MobileMCP_selectTerms

When the Oracle Utilities Mobile Workforce Management application is launched, the Terms and Conditions need to be accepted. This method just presses the **Accept** button.

Note: This method is common for both MDT flows.

Example:

```
MobileMCP_selectTerms()
```

Input Parameters: <none>
Return Type: String

MobileMCP_applicationUrl

This function inputs the application URL with the username and password. It takes the following variables from the “configuration.properties” file:

- gStrMobileDeploymentURL
- gStrApplicationUserName
- gStrApplicationUserPassword

After providing input to these three fields, the function clicks the **Download** button that triggers the download app bundle process.

Example:

```
MobileMCP_applicationUrl()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

MobileMCP_loginPage

This function performs the login action by entering the MDT-Tag, application username, and password, and then clicking the **Login** button.

Example:

```
MobileMCP_loginPage()
```

```
Input Parameters: <String>
```

```
Return Type: String
```

HybridMCP_selectDeployment

After successful login, the **Select Deployment** page appears that requires a human intervention to select a proper deployment from the available list. This method simulates this action by selecting a valid REST-DEPLOYMENT.

Example:

```
HybridMCP_selectDeployment()
```

```
Input Parameters: <String>
```

```
Return Type: String
```

HybridMCP_startShift

After selecting deployment, this method starts the crew shift in MDT by clicking **Start**.

Example:

```
HybridMCP_startShift()
```

```
Input Parameters: <none>
```

```
Return Type: String
```

HybridMCP_waitForAllTaskDispatch

This method enables wait on the **Task - Home** page till all tasks are dispatched in the MDT. Pass "Number of Tasks" as an argument and also the max time to wait for all tasks to dispatch. If it crosses the max time, the test fails.

Example:

```
HybridMCP_waitForAllTaskDispatch()
```

```
Input-Parameters:<String, String>
```

```
Return Type: String
```

HybridMCP_gotoTaskHome

After the completion of all tasks, go to the **Task - Home** page to see the remaining task list. This method simulates this action. It opens **Menu** and clicks **Home** from the list.

Example:

```
HybridMCP_gotoTaskHome()
```

```
Input Parameters:<String>
```

```
Return Type: String
```

HybridMCP_startBreak

On the **Task - Home** page, this method finds the break-task by the Task ID, and then selects the particular task. On that task action page, it clicks **Start**.

Example:

```
HybridMCP_startBreak()
```

Input Parameters:<String, SString>

Return Type: String

HybridMCP_completeBreak

After starting break-task, this method completes it by clicking **Complete**.

Example:

```
HybridMCP_completeBreak()
```

Input Parameters:<None>

Return Type: String

HybridMCP_startNonProductiveTask

On the **Task - Home** page, this method finds the NPT-task by the Task ID, and then selects the particular task. When on the task action page, it clicks **Start**.

Example:

```
HybridMCP_startNonProductiveTask()
```

Input Parameters:<String, String>

Return Type: String

HybridMCP_completeNonProductiveTask

After starting break-task, this method completes the NPT-task by clicking **Complete**.

Example:

```
HybridMCP_completeNonProductiveTask()
```

Input Parameters:<None>

Return Type: String

HybridMCP_startPOU

On the **Task-Home** page, this method finds the POU-task by the Task ID and then selects the particular task. When on the task action page, it clicks **Start**.

Example:

```
HybridMCP_startPOU()
```

Input Parameters:<String, String>

Return Type: String

HybridMCP_completePOU

After starting break-task, this method completes that POU-Task by clicking **Complete**.

Example:

```
HybridMCP_completePOU()
```

Input Parameters:<None>

Return Type: String

HybridMCP_startM1Activity

On the **Task - Home** page, this method finds the M1-task by the Task ID and then selects the particular task. When on the task action page, it clicks **Start**.

Example:

```
HybridMCP_startM1Activity()
```

Input Parameters:<String, String>

Return Type: String

HybridMCP_completeM1Activity

After starting break-task, this method completes that M1-task by clicking **Complete**.

Example:

```
HybridMCP_completeM1Activity()
```

Input Parameters:<None>

Return Type: String

HybridMCP_startM2ConnectSPActivity

On the **Task - Home** page, this method finds the M2-task by the Task ID and then selects the particular task. When on that task action page, it clicks **Enroute** and **Start**.

Example:

```
HybridMCP_startM2ConnectSPActivity()
```

Input Parameters:<String, String>

Return Type: String

HybridMCP_completeM2ConnectSPActivity

After starting break-task, this method completes that M2-task by clicking **Complete**.

Example:

```
HybridMCP_completeM2ConnectSPActivity()
```

Input Parameters:<None>

Return Type: String

HybridMCP_endShift

This method completes/ends the crew shift by performing the actions of a field worker to complete the current day's crew shift. This function also closes the Next Gen MCP application. It returns "Y" on success, else returns "N".

Example:

```
HybridMCP_endShift()
```

Input Parameters: <none> Return Type: String

HybridMCP_closeRemoteDriver

This method ends or closes the connection with the Remote IOS/ Android MDT device. It is usually used at the end of an automation test.

Example:

```
HybridMCP_closeRemoteDriver()
```

Input Parameters: <none> Return Type: String

Chapter 3

Sample Work Flows

This chapter describes the Oracle Utilities Mobile Workforce Management sample flows that illustrate common use cases for Oracle Utilities Testing Accelerator. It also explains the procedure to execute these sample flows. It includes the following sections:

- [Sample Flows](#)
- [Executing Sample Flows](#)

Sample Flows

The sample flows delivered as part of Oracle Utilities Testing Accelerator for Oracle Utilities Mobile Workforce Management demonstrate how flows can be created for Web services based testing and for a combination of Web services and UI based testing using the same framework.

These flows are designed to run using the demo data, there by giving the user, the ability to deploy Oracle Utilities Testing Accelerator for Oracle Utilities Mobile Workforce Management and execute the sanity flows immediately. The flows perform a part of the basic sanity testing required to certify that the Oracle Utilities Mobile Workforce Management environment has been setup appropriately.

This section includes the following sample work flows:

- [Non-MDT Flow](#)
- [MDT Flow](#)
- [M2 Non-MDT Flow](#)
- [MDT Flow Using NextGen MCP](#)
- [MDT Flow Using HybridMCP](#)
- [MWM_Cloud_Environment_Sanity](#)
- [M2 Android-MDT Sanity Flow](#)
- [M2 iOS-MDT Sanity Flow](#)

Non-MDT Flow

The Non-MDT flow comprises the creation and completion life cycle of an M1 activity - break, non-productive task (NPT), and real period of unavailability (POU), along with the crew shift.

The following table lists the tasks that are created and their respective components in Oracle Utilities Mobile Workforce Management:

Task	MWM Component
Activity	M1-Activity
Break	M1-Break
Non-productive Task	M1-NonProductiveTask
Real Period of Availability	M1-RealPOU

Activity, real POU, break, and non-productive tasks are scheduled to a planned crew shift that has the matching location, required capability, and scheduling time window. The dispatcher logs in the planned crew shift and starts the shift. Once the crew shift starts, activity, non-productive task, real POU, and break are dispatched to the crew shift. The dispatcher starts, completes the activity, real POU, break, and non-productive task.

The M1-Assignment component is used to assign a transition (to enroute > start > completing an activity). Similarly, M1-BreakTask, M1-NonProductiveTask, and M1-POUTask components are used to start, complete the break, non-productive tasks, and POU respectively. The dispatcher completes the crew shift after completing the dispatched tasks.

MDT Flow

The MDT flow includes the creation and completion of an activity, break, non-productive task (NPT), and real period of unavailability (POU) using connected MCP.

Task	MWM Component
Activity	M1-Activity
Break	M1-Break
Non-productive Task	M1-NonProductiveTask
Real Period of Availability	M1-RealPOU

Activity, real POU, break, and non-productive tasks are scheduled to a planned crew shift that has the matching location, required capability, and scheduling time window.

The MDT flow is as follows:

1. The dispatcher launches the connected MCP.
2. The crew logs in with the valid credentials and starts the crew shift.
3. The activity, break task, and non-productive tasks are dispatched to the crew shift.
4. Use the following components applicable for each task:

Task	MWM Component
Start the crew shift	ConnectedMCP_StartCrewShift
Verify if the tasks are dispatched	ConnectedMCP_DispatchedTask
Enroute status	ConnectedMCP_EnrouteActivity
Start activity	ConnectedMCP_StartActivity
Complete activity	ConnectedMCP_CompleteActivity

5. The crew shift is completed after completing all the dispatched tasks.

M2 Non-MDT Flow

The M2-Non-MDT flow includes the creation and completion of M-Activity (Install Meter Activity, for example).

Create an Install Meter Activity using the **M2-InstallMeterActivity** component.

The M2-Activity is scheduled to a planned crew shift that has the matching location, required capability, and the scheduling time window.

The M2 Non-MDT flow is as follows:

1. The dispatcher logs in the planned crew shift.
2. The dispatcher starts and completes the M2-Activity.
The M2-InstallMeterAssignment component is used for M2-Activity assignment transition.
3. The dispatcher completes the crew shift after completing the M2-Activity (Install Meter Activity).

MDT Flow Using NextGen MCP

The MDT flow includes the creation and completion of an activity, break, non-productive task (NPT), and real period of unavailability (POU) using NextGen MCP.

Task	MWM Component
Activity	M1-Activity
Break	M1-Break
Non-productive Task	M1-NonProductiveTask
Real Period of Availability	M1-RealPOU

Activity, real POU, break, and non-productive tasks are scheduled to a planned crew shift that has the matching location, required capability, and scheduling time window.

The MDT flow is as follows:

1. The dispatcher launches the NextGen MCP (Google Chrome Browser).
2. The crew logs in with valid credentials and starts the crew shift.
3. The activity, break task, and non-productive tasks are dispatched to the crew shift.
4. Use the following components applicable for each task:

Task	MWM Component
Start the crew shift	NextGenMCP_StartCrewShift
Verify if the tasks are dispatched	NextGenMCP_DispatchedTask
Enroute status	NextGenMCP_EnrouteActivity
Start activity	NextGenMCP_StartActivity

5. The crew shift is complete after completing all the dispatched tasks.

Note: The MDT Flow using NextGen MCP can be run/played back in OpenScript 12.5 version. OpenScript v12.5 currently does not support recording on Google Chrome at its current level. But, this is an expected feature in OpenScript v12.6 (which would soon be available).

MDT Flow Using HybridMCP

Note: While using secured (https) connection to access the HybridMCP Web application for the first time, “Your connection is not private” message is displayed. Accept and proceed further to access the application.

The HybridMCP MDT flow includes the creation and completion of an activity, break, non-productive task (NPT), and real period of unavailability (POU) using HybridMCP.

Task	MWM Component
Activity	M1-Activity
Break	M1-Break
Non-productive Task	M1-NonProductiveTask
Real Period of Availability	M1-RealPOU

Activity, real POU, break, and non-productive tasks are scheduled to a planned crew shift that has the matching location, required capability, and scheduling time window.

The MDT flow is as follows:

1. The dispatcher launches the HybridMCP (Google Chrome browser).
2. The crew logs in with valid credentials and starts the crew shift.
3. The activity, break task, and non-productive tasks are dispatched to the crew shift.
4. Use the following components applicable for each task:

Task	MWM Component
Start the crew shift	HybridMCP_StartShift
Verify if the tasks are dispatched	HybridMCP_TaskDispatched
Enroute status	HybridMCP_EnrouteActivity
Complete activity	HybridMCP_CompleteActivity
Complete the crew shift	HybridMCP_CompleteShift

5. The crew shift is complete after completing all the dispatched tasks.

Note: The MDT Flow using HybridMCP can be run/played back in OpenScript 12.5 version. OpenScript v12.5 currently does not support recording on Google Chrome at its current level. But, this is an expected feature in OpenScript v12.6 (which would soon be available).

MWM_Cloud_Environment_Sanity

Note: This Cloud Environment Sanity flow is not certified and is only for demo purposes. Also, to use this flow, the Oracle Utilities Application Framework patch 24941499 should be applied on Oracle Utilities Application Framework v4.3.0.1 (Oracle Utilities Mobile Workforce Management v2.3.0.3 uses this).

The MWM_Cloud_Environment_Sanity flow sets up the necessary gold data, its pre-requisites, and post gold data configuration, for any Oracle Utilities Mobile Workforce Management Cloud Initial Install environment. It also executes one sample Non-MDT Sanity test flow on that environment to ensure the environment is ready for use.

The flow includes the scenarios mentioned below:

Scenario	Task
Pre-requisites	Inserts the pre-required data, such as Time zone, Country, and other mandatory installation options into the Cloud application, before inserting the gold data.
Gold Data	Inserts all the required gold data setup into the application.
Post Gold Data Configuration	Inserts the post-gold data setup, such as Scheduler configuration and Scheduler setup.
Sanity	Performs the Non-MDT sanity to ensure the Cloud application is ready to use.

Note: This Oracle Utilities Mobile Workforce Management Cloud Environment Sanity Flow can be run/played back in OpenScript 12.5 version. OpenScript v12.5 currently does not support recording on Google Chrome at its current level. But, this is an expected feature in OpenScript v12.6 (which would soon be available).

M2 Android-MDT Sanity Flow

The M2 Android-MDT Sanity flow runs only on the Oracle Utilities Mobile Workforce Management Android apk file that is in debug build mode. Place your Oracle Utilities Mobile Workforce Management Android apk file in any of your local directory.

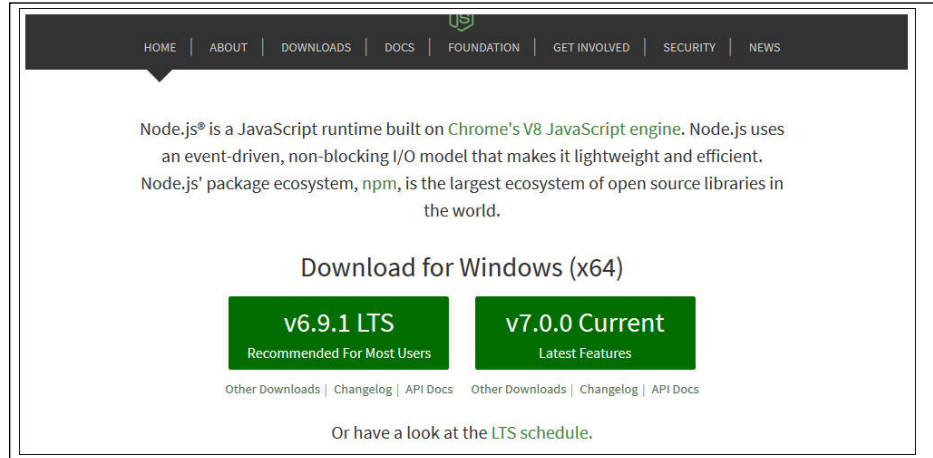
Note: To run this flow, download the required five JAR files. The table below lists the versions and links to download.

File Name with JAR Version	Download Link
selenium-server-standalone-2.53.0.jar	http://selenium-release.storage.googleapis.com/2.53/selenium-server-standalone-2.53.0.jar
selenium-java-2.53.0.jar	http://selenium-release.storage.googleapis.com/2.53/selenium-java-2.53.0.zip
commons-validator-1.4.0.jar	http://central.maven.org/maven2/commons-validator/commons-validator/1.4.0/commons-validator-1.4.0.jar
commons-lang3-3.4.jar	http://central.maven.org/maven2/org/apache/commons/commons-lang3/3.4/commons-lang3-3.4.jar
java-client-4.1.1.jar	http://central.maven.org/maven2/io/appium/java-client/4.1.1/java-client-4.1.1.jar

To execute the Android MDT Sanity flow:

1. Download the JAR files and place them in the local directory.
 <Open Script repository location>\outsp-function-libs\MWM\OUMWMHYBRIDLIB
2. Install Node.js server which is a freeware and does not require a license.
 - a. Navigate to the below URL.
<https://nodejs.org/en/>

- b. Download the latest version (v7.0.0 is the latest as on date).



- c. Launch the Setup Wizard and complete the installation.
- d. Verify that Node.js is installed properly. Run the following command at the Command prompt.

```
node -v
```

This command shows the Node.js version currently installed.

```
C:\Users\nilesing>node -v
v4.6.0
```

- e. Verify that Node Package Manager (NPM) is properly configured. Run the following command at the Command prompt.

```
npm -v
```

This command shows the NPM version currently configured.

```
C:\Users\nilesing>npm -v
2.15.9
```

3. Set the environment variable for proxy.

This step is required if you are connected behind the corporate proxy. In order to set the proxy variable for Node.js, “proxy-IP” and “PORT” values are needed. Please contact your respective IT-team for this information.

Run the following commands at the Command prompt:

```
set HTTP_PROXY=http://<<server>>:<<port>>
set HTTPS_PROXY=http:// <<server>>:<<port>>
```

```
C:\Users\nilesing>set HTTP_PROXY=http://[redacted].oracle.com:80
C:\Users\nilesing>set HTTPS_PROXY=http://[redacted].oracle.com:80
```

4. Install the latest Appium server. Run the command below at the Command prompt:

```
npm install -g appium --loglevel verbose
```

```

C:\Users\nilesing>npm install -g appium --loglevel verbose
npm info it worked if it ends with ok
npm verb cli [ 'C:\\Program Files\\nodejs\\node.exe',
npm verb cli 'C:\\Program Files\\nodejs\\node_modules\\npm\\bin\\npm-cli.js',
npm verb cli 'install',
npm verb cli '-g',
npm verb cli 'appium',
npm verb cli '--loglevel',
npm verb cli 'verbose' ]
npm info using npm@2.15.9
npm info using node@v4.6.0
npm verb install initial load of C:\Users\nilesing\AppData\Roaming\npm\package
  
```

Alternatively, you can install Appium GUI from the following URL. Download the latest version and extract the ZIP file in local directory.

<https://bitbucket.org/appium/appium.app/downloads/>

Name	Size	Up
Download repository	77.7 MB	
appium.dmg	106.2 MB	ast
appium-1.5.3.dmg	106.2 MB	ast
appium-1.5.2.dmg	104.1 MB	ast
AppiumForWindows.zip	47.3 MB	ast
AppiumForWindows_1_4_16_1.zip	47.3 MB	ast
appium-1.4.13.dmg	178.3 MB	dcu
AppiumForWindows_1_4_13_1.zip	46.2 MB	ast

5. Start the Appium server.

- **NPM command line:** Run the appium at the Command prompt.
- **Appium GUI:** Launch appium.exe file and click **Start**.

The appium server starts running at the default port.

```

> Launching Appium server with command: C:\Program Files (x86)\Appium\node.exe lib\server\
main.js --address 127.0.0.1 --port 4723 --platform-name Android --platform-version 22 --
automation-name Appium --log-no-color
> info: Welcome to Appium v1.4.16 (REV ae6877eff263066b26328d457bd285c0cc62430d)
> info: Appium REST http interface listener started on 127.0.0.1:4723
> info: [debug] Non-default server args:
{"address": "127.0.0.1", "logNoColors": true, "platformName": "Android", "platformVersion": "22", "automati
onName": "Appium"}
> info: Console LogLevel: debug
  
```

For more information on Appium server, see <https://www.npmjs.com/package/appium>.

6. Download and install the latest Android Software Development Kit (SDK) from <https://developer.android.com/studio/index.html>.

Since the latest Android SDK does not come as a standalone installer, download and install Android Studio.



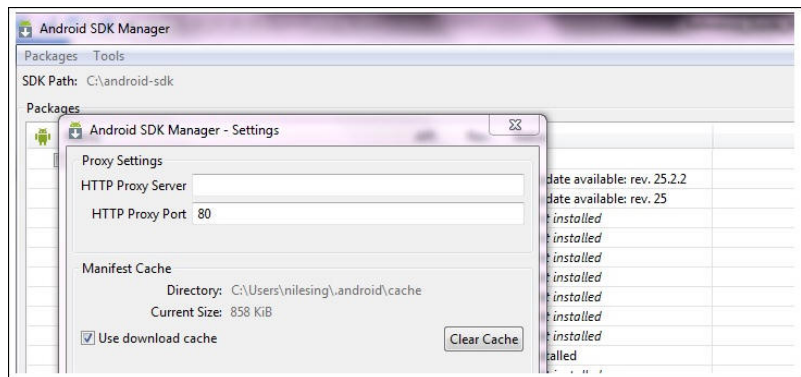
After download, do a quick installation. After successful installation, verify the ANDROID_HOME environment variable by running the command at the Command prompt. The output will be the location path where the android SDK resides.

```
echo %ANDROID_HOME%
```

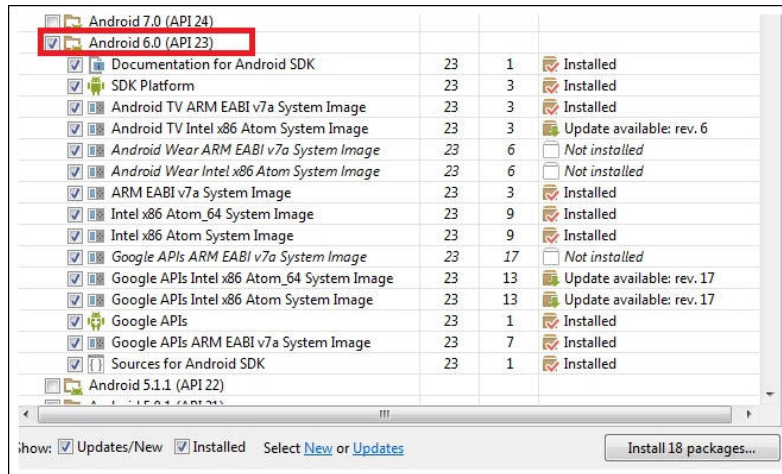


7. Ensure that the latest Android API (5.0 or above) is installed. Use ANDROID_HOME/SDK Manager.exe to verify the same.

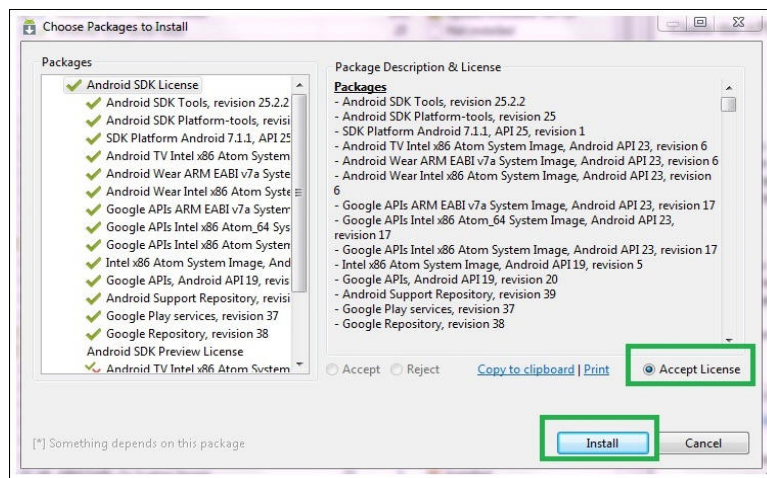
SDK Manager.exe automatic fetches the latest Android images and related API dependencies. If it does not, it may be due to the proxy firewall. Configure the proxy settings in **Tools > Options** and provide the proxy IP and port.



Select the API version to be installed. Finally, click **Install (N) Packages** to proceed with installation.



Accept the **License Agreement** and click **Install**.



At this point, all the necessary packages are successfully installed. Proceed with executing the flow.

8. To execute the Android-MDT Sanity Flow on an Android Emulator, add the virtual device in Android SDK as follows:
 - a. Run `ANDROID_HOME/AVD Manager.exe`. The **Android Virtual Device (AVD) Manager** opens.
 - b. On the **Android Virtual Devices** tab, click **Create** and enter appropriate details. The list below shows sample details.
 - AVD Name: Nexus_4
 - Device: Nexus 4 (4.7", 768 x 1280: xhdpi)
 - Target: Android (5.0 or above)
 - Keyboard: Enable
 - Skin: No Skin
 - Front Camera: None
 - Back Camera: None

- Ram: 1200, VM Heap: 64
 - Internal Storage: 200 MB
 - SD card Size: 1000 MB
 - Use Host GPU: Enable
- c. Click **OK**. A new AVD profile with name “Nexus_4” is created.



- d. Select **Nexus_4** and click **Start**. The emulator runs in the GUI mode.
- e. At the Command prompt, run the `adb devices` command. You can see the emulator type and UID.

```
C:\Users\nilesing>adb devices
List of devices attached
emulator-5554    device
```

- f. Enter the emulator name and UID in the `configuration.properties` file. The file is in the `OATS_REPOSITORY/etc` folder.

```
29 gStrAndroidDeviceName=emulator
30 gStrAndroidDeviceUID=emulator-5554
```

See the [Setting Up Sample Flows](#) section to enter details in the `configuration.properties` file.

For more information on managing your Android virtual devices, go to <https://developer.android.com/studio/run/managing-avds.html>.

9. Follow the steps listed out in the [Running the Android-MDT Flow](#) section to complete executing the flow.

To execute the Android-MDT flow on a real Android device that is connected, follow these steps:

1. Follow the instructions below to allow the debugging mode:
 - a. On the Android device, navigate to **Settings > About Device**.
 - b. Tap the **Build Number** button about 7 times. Developer Mode should now be unlocked.
 - c. Navigate to **Settings > Developer Options > USB Debugging**.

For more information on setting the debug mode, go to <http://www.howtogeek.com/129728/how-to-access-the-developer-options-menu-and-enable-usb-debugging-on-android-4.2/>.

- Run the `adb devices` command at the Command prompt. You can see the real device UID.

```
C:\Users\nilesing>adb devices
List of devices attached
emulator-5554    device
02157df2bcdf3528  device
```

- Enter the device name and UID in the **configuration.properties** file. The file is in the OATS_REPOSITORY/etc folder.
See the [Setting Up Sample Flows](#) section to enter details in the configuration.properties file.
- Enter the Appium node URL connection string and Appium build path in the **configuration.properties** file.

```
gStrAndroidMCPRemoteNodeURL=http\://127.0.0.1\4723/wd/hub
```

- Set the IP and Port details based on the running Appium service. This can be found from the Appium console logs.
- Follow the steps listed out in the [Running the Android-MDT Flow](#) section to complete executing the flow.

Running the Android-MDT Flow

The Android-MDT flow includes the creation and completion of an activity, break, non-productive task (NPT), and real period of unavailability (POU) using Web Driver.

Task	MWM Component
Activity	M1-Activity
Break	M1-Break
Non-productive Task	M1-NonProductiveTask
Real Period of Availability	M1-RealPOU

Activity, real POU, break, and non-productive tasks are scheduled to a planned crew shift that has the matching location, required capability, and scheduling time window.

The Android-MDT flow is as follows:

- The dispatcher launches the Android Emulator or a real Android device.
- The crew logs in with valid credentials and starts the crew shift.
- The activity, break task, and non-productive tasks are dispatched to the crew shift.
- Use the following components applicable for each task.

Task	MWM Component
Login into Crew Shift on Android device	Webdriver_AndroidLogin
Start the crew shift	Webdriver_startShift

Task	MWM Component
Verify if the tasks are dispatched	Webdriver_waitForTaskDispatch
Enroute status	Webdriver_startM1Activity
Complete activity	Webdriver_completeM1Activity
Complete the crew shift	Webdriver_endShift

5. The crew shift is complete after completing all the dispatched tasks.

M2 iOS-MDT Sanity Flow

The M2 iOS-MDT Sanity flow can only be run on an iOS system. As of now, this flow can be run only on an iOS emulator and cannot be run on iOS real devices. It runs only on the MWM iOS ipa file which is in debug build mode. At the time of execution, place the iOS ipa file in a local directory.

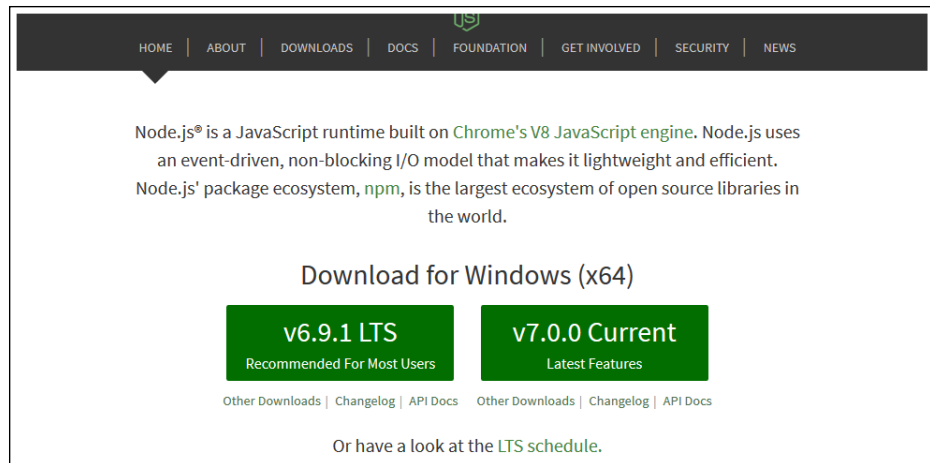
Note: To run this flow, download the required five JAR files. The table below lists the versions and links to download.

File Name with JAR Version	Download Link
selenium-server-standalone-2.53.0.jar	http://selenium-release.storage.googleapis.com/2.53/selenium-server-standalone-2.53.0.jar
selenium-java-2.53.0.jar	http://selenium-release.storage.googleapis.com/2.53/selenium-java-2.53.0.zip
commons-validator-1.4.0.jar	http://central.maven.org/maven2/commons-validator/commons-validator/1.4.0/commons-validator-1.4.0.jar
commons-lang3-3.4.jar	http://central.maven.org/maven2/org/apache/commons/commons-lang3/3.4/commons-lang3-3.4.jar
java-client-4.1.1.jar	http://central.maven.org/maven2/io/appium/java-client/4.1.1/java-client-4.1.1.jar

To execute the iOS-MDT Sanity flow, follow these steps:

1. Download the JAR files and place them in the local directory.
 - <Open Script repository location>\outsp-function-libs\MWM\OUMWMHYBRIDLIB
2. Install Node.js server which is a freeware and does not require a license.
 - a. Navigate to the below URL.
 - <https://nodejs.org/en/>

- b. Download the latest version (v7.0.0 is the latest as on date).



- c. Launch the Setup Wizard and complete the installation.
- d. Verify that Node.js is installed properly. Run the following command at the Command prompt.

```
node -v
```

This command shows the Node.js version currently installed.

```
C:\Users\nilesing>node -v
v4.6.0
```

- e. Verify that Node Package Manager (NPM) is properly configured. Run the following command at the Command prompt.

```
npm -v
```

This command shows the NPM version currently configured.

```
C:\Users\nilesing>npm -v
2.15.9
```

3. Set the environment variable for proxy.

This step is required if you are connected behind the corporate proxy. In order to set the proxy variable for Node.js, “proxy-IP” and “PORT” values are needed. Please contact your respective IT-team for this information.

Run the following commands at the Command prompt:

```
set HTTP_PROXY=http://<<server>>:<<port>>
set HTTPS_PROXY=http:// <<server>>:<<port>>
```

```
C:\Users\nilesing>set HTTP_PROXY=http://[redacted].oracle.com:80
C:\Users\nilesing>set HTTPS_PROXY=http://[redacted].oracle.com:80
```

- Install the latest Appium server. Run the command below at the Command prompt:

```
npm install -g appium --loglevel verbose
```

```

C:\Users\nilesing>npm install -g appium --loglevel verbose
npm info it worked if it ends with ok
npm verb cli [ 'C:\\Program Files\\nodejs\\node.exe',
npm verb cli 'C:\\Program Files\\nodejs\\node_modules\\npm\\bin\\npm-cli.js',
npm verb cli 'install',
npm verb cli '-g',
npm verb cli 'appium',
npm verb cli '--loglevel',
npm verb cli 'verbose' ]
npm info using npm@2.15.9
npm info using node@v4.6.0
npm verb install initial load of C:\Users\nilesing\AppData\Roaming\npm\package

```

Alternatively, you can install Appium GUI from the following URL. Download the latest version and extract the ZIP file in local directory.

<https://bitbucket.org/appium/appium.app/downloads/>

Name	Size	Up
Download repository	77.7 MB	
appium.dmg	106.2 MB	ast
appium-1.5.3.dmg	106.2 MB	ast
appium-1.5.2.dmg	104.1 MB	ast
AppiumForWindows.zip	47.3 MB	ast
AppiumForWindows_1_4_16_1.zip	47.3 MB	ast
appium-1.4.13.dmg	178.3 MB	dcu
AppiumForWindows_1_4_13_1.zip	46.2 MB	ast

- Start the Appium server.
 - NPM command line: Run the appium at the Command Prompt.
 - Appium GUI: Launch appium.exe file and click **Start**.

The appium server starts running at the default port.W

```

> Launching Appium server with command: C:\Program Files (x86)\Appium\node.exe lib\server\main.js --address 127.0.0.1 --port 4723 --platform-name Android --platform-version 22 --automation-name Appium --log-no-color
> info: Welcome to Appium v1.4.16 (REV ae6877eff263066b26328d457bd285c0cc62430d)
> info: Appium REST http interface listener started on 127.0.0.1:4723
> info: [debug] Non-default server args:
{"address": "127.0.0.1", "logNoColors": true, "platformName": "Android", "platformVersion": "22", "automationName": "Appium"}
> info: Console LogLevel: debug

```

For more information on Appium server, see <https://www.npmjs.com/package/appium>.

- Install X-code on your iOS system if your machine does not have it. X-Code is available with your iOS installation CD.
- Get the UID, name, and version of the simulator. Enter the following command using Terminal.

```
instruments -s devices
```

- Enter the UID, name, and version in the **configuration.properties** file.

```
37 gStrIOSVersion=9.1
38 gStrIOSDeviceName=iPhone 4s
```

- Enter the Appium node URL connection string and Appium build path in the **configuration.properties** file.

```
gStrIosMCPRemoteNodeURL=http\://10.178.151.137\ :4723/wd/hub
```

- Set the IP and Port details based on the running Appium service. This can be found from the Appium console logs.

The iOS-MDT flow includes the creation and completion of an activity, break, non-productive task (NPT), and real period of unavailability (POU) using Web Driver.

Task	MWM Component
Activity	M1-Activity
Break	M1-Break
Non-productive Task	M1-NonProductiveTask
Real Period of Availability	M1-RealPOU

Activity, real POU, break, and non-productive tasks are scheduled to a planned crew shift that has the matching location, required capability, and scheduling time window.

The iOS-MDT flow is as follows:

- The dispatcher launches the iOS Emulator.
- The crew logs in with valid credentials and starts the crew shift.
- The activity, break task, and non-productive tasks are dispatched to the crew shift.
- Use the following components applicable for each task.

Task	MWM Component
Login into Crew Shift on Android device	Webdriver_AndroidLogin
Start the crew shift	Webdriver_startShift
Verify if the tasks are dispatched	Webdriver_waitForTaskDispatch
Enroute status	Webdriver_startM1Activity
Complete activity	Webdriver_completeM1Activity
Complete the crew shift	Webdriver_endShift

- The crew shift is complete after completing all the dispatched tasks.

Executing Sample Flows

This section describes the procedure to setup sample flows and execute them.

- [Pre-requisites](#)
- [Setting Up Sample Flows](#)

Pre-requisites

To execute the sample flow, ensure the following pre-requisites are met:

- Oracle Utilities Mobile Workforce Management v2.3.0.3/Oracle Real-Time Scheduler v2.3.0.3 is up and running with the demo data pack.
- OpenScript is installed in the local machine. Refer to the *Oracle Utilities Testing Accelerator Installation and Administration Guide* for the version details.
- Oracle Utilities Testing Accelerator is installed and repository/directory is setup in the local machine appropriately. See the *Oracle Utilities Testing Accelerator Installation and Administration Guide* for more details.
- **Important:** Browser Playback settings of OpenScript must be set explicitly to either Mozilla Firefox or Chrome Browser for MDT flow and only “Chrome Browser” for NextGen MCP and HybridMCP flows.

Setting Up Sample Flows

To setup a sample flow:

1. Login to Oracle Utilities Mobile Workforce Management/Oracle Real-Time Scheduler.
2. Import the Inbound Web services into the Oracle Utilities Mobile Workforce Management/Oracle Real-Time Scheduler application where the scenarios need to be executed.

See the **Inbound Web Services** section in *Oracle Utilities Testing Accelerator User's Guide* for steps to import the Inbound Web services.

3. Navigate to **Admin > B > Bundle Import > Add**.
4. Enter the **External Reference**, **Detailed Description**, and **Bundle Details** from the IWS Bundle Export Dump.
5. Click **Save**, and then click **Apply bundle**.
6. Launch OpenScript in the local machine and perform the following steps:
 - a. Navigate to **View > OpenScript Preferences**.
 - b. In the left tree, select **OpenScript**. In the sub tree, select **Playback**, and then select **Error Recovery**.
 - c. Click **SetAll** and select **Report Error and Continue**.
 - d. Click **Apply**, and then click **Close**.
7. Configure the **configuration.properties** file:
 - a. Provide the application and NextGen MCP URLs for the parameter:


```
gStrNewMCPURL=http\://<%serverName%>\:<%portNumber%>/ouaf/mobility/www/index.html
```
 - b. Provide the additional path required for inbound web service URL:


```
gStrApplicationXAIServerPath=/<%webservices/gStrApplicationURL%>/<%AppendThisToAbove gStrApplicationURL%>/
```

- c. Provide an environment name for display in the results email:

```
gStrEnvironmentName= <%testEnvironmentName%>
```

- d. Provide the application login user ID:

```
gStrApplicationUserName= <%UserName%>
```

- e. Provide the application login password:

```
gStrApplicationUserPassword= <%password%>
```

- f. Provide the SMTP email server and e-mail ID:

```
gStrSMTP_HOST_NAME=<%SMTP ServerName%>
```

```
gStrSMTP_PORT=<%PortNumber%>
```

```
gStrTO_EMAIL_RECIPIENTS=<%e-mail Id%>
```

- g. Provide the application database details as below:

```
gStrApplicationDBConnectionString =<%jdbc Connectionstring%>
```

```
eg: jdbc\:oracle\:thin\:@<DBserverName%>\:<%port%>\:<%DBSID%>
```

```
gStrApplicationDBUsername=<DBUserID%>
```

```
gStrApplicationDBPassword=<DBPassword%>
```

- h. Provide the full directory path of Oracle Application Testing Suite repository directories in the local machine.

```
gStrOutputFilePath=<%LogFilePath%>
```

```
Example: C:\\OATSOUWWM_DEMO\\OUTSP\\Logs\\
```

```
gStrXSDFiles=<%XSD Folder path%>
```

```
Example: C:\\OATSOUWWM_DEMO\\OUTSP\\Logs\\
```

- i. Provide the Mobile Android device details.

Note: Below is an example for mentioning the version, name, and UID. Run the 'adb devices' command for identifying the respective Android device details.

```
gStrAndroidDeviceName=emulator
```

```
gStrAndroidDeviceUID=emulator-5554
```

```
gStrAndroidDeviceName=Nexus_4
```

- j. Provide the Mobile iOS device details.

Note: Below is an example for mentioning the version, name, and UID. Run the 'instruments -s devices' command for identifying the respective iOS device details.

```
gStrIOSVersion=9.1
```

```
gStrIOSDeviceName=iPhone 4s
```

```
gStrIOSDeviceUID=33ad17f071d36d191d0a6e6b102b8ed3e7783c58
```

- k. Provide the Appium node URL details.

```
gStrAndroidMCPRemoteNodeURL=http\://<<Appium
```

```
Server>>\:<<port>>/wd/hub
```

```
gStrIosMCPRemoteNodeURL= http\://<<Appium Server>>\:<<port>>/
```

```
wd/hub
```

- l. Provide the Android/iOS build paths.

```
gStrAndroidBuildPath=<<your local directory
```

```
>>\\Oracle_MWM_Debug.apk
```

```
gStrIosBuildPath=<<your local directory
```

```
>>\\Oracle_MWM_Debug.ipa
```


8. Create two folders (MWM and Core) in the outsp-function-libs folder in the Oracle Application Testing Suite repository directory. Copy the function libraries to the respective folders.
 - MWM
 - OUMWMCONNECTEDMCPLIB
 - OUMWMLIB
 - OUMWMMCP LIB
 - Core
 - OUTSPCORELIB
 - WSCOMMONLIB
 - WSVVALIDATELIB
9. Copy all the .jar files provided in the installer into the **genericJars** folder in the Oracle Application Testing Suite repository directory.

Appendix A

Inbound Web Services

The Oracle Utilities Mobile Workforce Management components are developed using Web services method, and these components require Inbound Web Services to be defined in the application.

For instructions to create, import, or search an Inbound Web Service, see the **Setting Up Inbound Web Services** appendix in *Oracle Utilities Testing Accelerator User's Guide*.

The list of Inbound Web Services provided to use with the delivered components and flows is as follows:

- ATM1Activity
- ATM1Alert
- ATM1AllocateActivityToAShift
- ATM1AppointmentBookingRequest
- ATM1Assignment
- ATM1AdditionalUserInfo
- ATM1BreakTask
- ATM1CommonShiftWeeklyTemplate
- ATM1Company
- ATM1ComplexActivity
- ATM1Contractor
- ATM1ContractorEligibility
- ATM1Crew
- ATM1CrewShift
- ATM1CrewShiftCloseOpenAllocate
- ATM1CrewShiftTemplate
- ATM1DailySpeedTemplate
- ATM1DeferActivity
- ATM1DepotRelatedActivity
- ATM1DepotRelatedShift
- ATM1DepotSinglePersonCrew
- ATM1DepotTask

- ATM1Dispatcher
- ATM1DispatcherShift
- ATM1FixToCrew
- ATM1ForceLogOffCrewShift
- ATM1GetShiftCapabilities
- ATM1HazardArea
- ATM1MailingList
- ATM1MainMail
- ATM1MobileWorker
- ATM1NonPrdTask
- ATM1OverrideTimeWindow
- ATM1POUtask
- ATM1RealPOU
- ATM1ResTimesheet
- ATM1ResourceLeave
- ATM1SchedulerRegistryRead
- ATM1ShiftWeeklyTemplate
- ATM1SimpleProcedure
- ATM1SinglePersonCrew
- ATM1SpeedProfileTemplate
- ATM1SubscriptionWeeklyTemplate
- ATM1TaskScheduleDetails
- ATM1TemplateMail
- ATM1TemplatePOU
- ATM1UnassignActivity
- ATM1Vehicle
- ATM1VehicleLeaveArea
- ATM2InstallMeterActivity
- ATM2InstallMeterAssignment
- ATM2ConnectSPActivity
- ATM2ConnectSPAssignment
- ATM2CutNonPayItemActivity
- ATM2CutNonPayItemAssignment
- ATM2DisconnectMeterActivity
- ATM2DisconnectMeterAssignment
- ATM2ExchangeItemActivity
- ATM2ExchangeItemAssignment
- ATM2InstallMeterActivity

- ATM2InstallMeterAssignment
- ATM2MeterReadActivity
- ATM2MeterReadAssignment
- ATM2RemoveMeterActivity
- ATM2RemoveMeterAssignment
- ATM2BasicItemActivity
- ATM2BasicItemAssignment
- ATM2RemoveItemActivity
- ATM2RemoveItemAssignment
- ATM1ActivityType
- ATM1AlertType
- ATM1BreakTaskType
- ATM1Capacity
- ATM1CapacityTemplate
- ATM1CapacityType
- ATM1CapacityWeeklyTemplate
- ATM1ComplexActivityType
- ATM1ContractorType
- ATM1CrewHierarchy
- ATM1CrewShiftType
- ATM1CrewType
- ATM1Deployment
- ATM1DeploymentPart
- ATM1DeploymentType
- AT_M1DeploymentTypeRestfulSvc
- ATM1Depot
- ATM1DepotProfile
- ATM1DepotTaskType
- ATM1DispatchArea
- ATM1DispatcherType
- ATM1DispatcherShiftLite
- ATM1Equipment
- ATM1GeographicArea
- ATM1GlobalConfigurations
- ATM1KPI
- ATM1Location
- ATM1MobileDeviceTerminal
- ATM1MobileDeviceTerminalType

- ATM1MobileWorkerType
- ATM1NonPrdTaskType
- ATM1POUTaskType
- ATM1POUType
- ATM1PriorityProfile
- ATM1ProcedureType
- ATM1RemarkType
- ATM1ResTimesheetType
- AT_M1ResourceLite
- ATM1Scheduler
- ATM1SchedulerArea
- ATM1SchedulerConfiguration
- ATM1ServiceArea
- ATM1ServiceAreaHierarchy
- ATM1ServiceClass
- ATM1ShiftCostProfile
- ATM1Skill
- ATM1StatusReason
- ATM1VehicleType
- ATM1WorkCalendar
- ATM1WorkProfile
- ATM2MaintainUtilityActByHost
- ATM1RetrievePOUTaskIdsOfPOU
- ATM1RetrieveMWIdFromUserId
- ATM1RetrieveDeploymntId
- ATM1RetrieveAssgnmtsOfTask
- ATM1FetchActiveResourceIDByType
- ATM1CheckIfActScheduled