Oracle Real-Time Scheduler

Configuration Guide

2.1.0.1 for Windows

E26602-02

February 2012

ORACLE®

Oracle Real-Time Scheduler,Configuration Guide, Release 2.1.0.1 for Windows

E26602-02

Contents

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Chapter 11
Deploying the Application to Mobile Devices

Chapter 12
SMS Messaging

Appendix A
Scheduler Parameter Table

Appendix B
System Setup Quick Reference Table

Appendix C
Algorithm Entities

Appendix D

Appendix E

Appendix F

Index

Chapter 1

Overview

This guide describes how to configure Oracle Real-Time Scheduler (ORS). It is intended for implementers and system administrators responsible for configuration and initial setup of the application.

Oracle Real-Time Scheduler is based on the Oracle Utilities Application Framework (OUAF). For information about using and configuring basic Framework functions, see the Oracle Utilities Application Framework documentation. This guide only covers configuration of functions specific to Oracle Real-Time Scheduler.

The body of this guide presents conceptual information to help you understand how the system works as well as how the various configuration options affect system functionality. Once you have an understanding of the system's capabilities, you can plan your data setup and design any customizations you want to implement.

When you are ready to implement your design, use Appendix B to guide you through the setup process. Appendix B lists each object that can be configured, defines any prerequisites for configuration, and lists objects referenced by or associated to the object being configured.

> **Note:** The sequence in which you configure system objects is very important. Appendix B describes data dependencies and defines the order in which objects should be configured. By following this sequence carefully, you can streamline the configuration process and reduce the amount of time required for setup.

This section includes the following topics:

- *Architecture Overview*
- *Oracle Utilities Application Framework Configuration Tools*
- *Entity Naming Conventions*
- *Custom Algorithms*
- *Embedded Documentation*
- *Demonstration Examples*
- *Integrating with Other Oracle Applications*

Architecture Overview

Oracle Real-Time Scheduler (ORS) simplifies and optimizes the scheduling, dispatching, and tracking of mobile service crews and field activities.

Oracle Real-Time Scheduler has three functional components:

- **Resource Scheduling and Planning:** Supports resource planners and service managers in managing resources, planning shifts, and scheduling work. The system automatically

generates shifts and optimizes the schedule based on your business rules. This component comprises the following user functions:

- Resource Management setup and maintenance

- Service Management setup and maintenance

- Scheduler setup and maintenance

- **Common Dispatching Functionality:** Supports dispatchers as they handle exceptions throughout the day, and enables context-based decision making at the dispatcher level. The system can be configured to automatically dispatch all activities or limit auto-dispatching to certain activity types or shifts. The system maintains real-time communication with mobile resources, tracks the location of crews and vehicles, and allows dispatchers to monitor and manage activities, crews, alerts, and key performance indicators. Common dispatching functionality is provided through the Common Dispatching Interface (CDI) portal.

- **Mobile Communications Platform:** Supports mobile crews as they perform service work by facilitating communication with the dispatcher, providing GPS-based mapping services, and processing activity status updates and work completion details. Oracle Real-Time Scheduler mobile application provides the user interface for this functionality.

These three functional components are built atop the Oracle Utilities Application Framework.
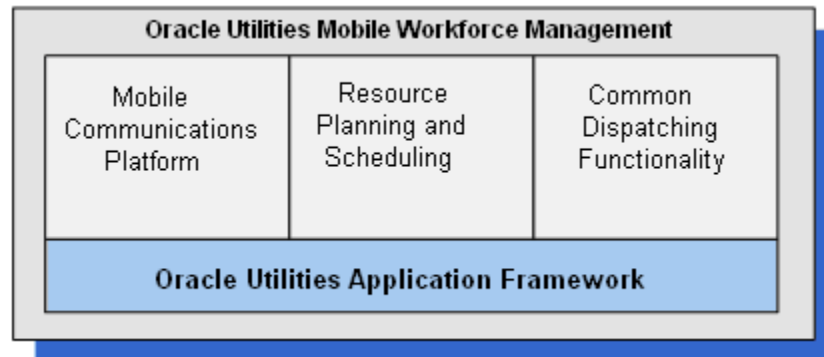
Oracle Utilities Application Framework Configuration Tools

The Oracle Utilities Application Framework (OUAF) configuration tools can be used to create and customize system entities, such as business objects, portals, zones, and UI maps. Refer to the Oracle Utilities Application Framework configuration tools documentation for instructions on using this tool.

This configuration guide will not duplicate the concepts and procedures presented in the Oracle Utilities Application Framework configuration tools documentation; rather, it will identify the ORS-specific objects that can be configured and customized using that tool, as well as application parameters and objects that can be managed within the ORS application components themselves.

This guide assumes that all individuals responsible for system configuration and implementation will be familiar with OUAF and will have completed training from Oracle on the OUAF Configuration Tool.

Entity Naming Conventions

Oracle Real-Time Scheduler uses naming conventions to identify and distinguish entities that belong to different Oracle applications. These conventions can help you locate entities and understand their context.

Each base product prefixes its entities with its 2-character owner code. For example:

- All Oracle Utilities Application Framework entities start with "F1"

- All Oracle Real-Time Scheduler entities start with "M1"

Oracle recommends that you follow these naming conventions and develop your own set of conventions for the entities you create. If you create new entities, DO NOT use these prefixes; use the prefix "CM" to identify entities that have been customized.

Custom Algorithms

Many functions in the system are performed using user-defined algorithms, also referred to as plug-ins. For example, user-defined algorithms can be used to create specific types of alerts and KPIs, generate shifts and POUs, and customize the content that appears in the dispatcher CDI Scheduling Gantt zone.

Custom algorithms allow implementers to modify how the system responds to certain system events. The system provides 'plug-in spots' where the custom algorithms can be invoked instead of the base algorithms provided with the system. See Appendix C for a list of all Oracle Real-Time Scheduler related system events supported in the base package. For instructions on creating custom algorithms, see the Framework documentation. To view information about specific algorithms provided with the base system, use the Application Viewer (also described in the Framework documentation). The Application Viewer provides information about the base logic, inputs, and outputs of each algorithm entity or plug-in spot.

Embedded Documentation

The Oracle Real-Time Scheduler server application, like all OUAF applications, provides extensive internal documentation. For example, detailed descriptions of system objects are included in the objects' maintenance portals. The lifecycle of each business object is described on the Lifecycle tab and depicted in flow diagrams on the Summary tab. This information is extremely useful for implementers and system administrators.

Embedded help is provided for all non-obvious fields at the UI map section level. The help content can be customized during implementation by adding custom help to the Override Help Text attribute. If a field on a UI map has associated help text, a **?** icon appears next to the field when the UI map is displayed. When the user clicks this icon, the contents of the Help Text attribute (or the Override Help Text attribute, if populated) is displayed in a pop-up window.

Demonstration Examples

The demonstration environment shipped with the base product provides setup examples that may be helpful as you implement your system.

Integrating with Other Oracle Applications

Refer to the appropriate Oracle Utilities Integration documentation for information about specific integrations.

Chapter 2

General Component Configuration

This section describes configuration of general components, including the following:

- *Installation Options*
- *Feature Configuration*
- *Master Configuration*

Installation Options

Installation options define the individual applications installed on your system and identify algorithms used to implement core system functions. These options also define global parameters such as the administrative menu style (alphabetical or functional), the country, language, currency code, as well as the base time zone to use for this implementation.

Installation options are stored in the installation record for your system. Use the Installation Options – Framework portal to configure these options. This portal is part of OUAF and is described in detail in the Framework documentation.

Base Time Zone

The date/time attributes for all time-sensitive application entities, including tasks and shifts, are stored in the server application time zone in standard time i.e. independent of any seasonal time shifting adjustments, if applicable, in that time zone. At the same time the system allows data to be entered and displayed in a different time zone in legal time, i.e. adjusted for seasonal time shifting, managing the conversion back and forth between the data entry and the storage time zones.

Entities associated with a geographic location such as activities, crew shifts and meetings, are entered and displayed in their corresponding time zone. The rest of the application uses the user's preferred time zone to display date and time information.

The server time zone, also referred to as the Base Time Zone, must be correctly specified on the installation options record. A user specific time zone can be specified in user preferences.

> **Note**: The installation record does not dictate the server time zone, but rather must match it.

Installation Algorithms

Installation algorithms implement global system functions and can be customized for each implementation.

The base package supports the following installation options ORS related system events:

- **Address Information**: Responsible for formatting address information for display throughout the system on the server.

- **Address Geocoding**: Responsible for geocoding an address (converting an address to a geocode latitude/longitude pair).

- **SMS Send**: Responsible for processing outgoing SMS messages.

- **SMS Receive**: Responsible for processing incoming SMS messages sent from a crew.

- **Remote Script Invocation**: Responsible for processing remote messages sent from the mobile device.

- **To Do Pre-Creation:** Responsible for making To Do Entries searchable by linking them to their related entity via a characteristic.

Refer to Appendix C for a complete list of ORS-related system events supported by the base package.

Feature Configuration

Oracle Real-Time Scheduler includes 'specialty' zones that invoke an external URL to render their contents. The Gantt and Calendar zones are both specialty zones. The functionality provided by these zones requires that you set up a feature configuration of type "Application Context Root," which defines the root location of the main application as well as other related applications.

The Common Dispatcher Interface (CDI) supports a map view of the dispatcher's monitored area. This CDI Map functionality requires that you set up a feature configuration of type "Map Server Configuration."

Refer to Appendix B to determine when in the configuration process you should perform feature configuration.

Follow the embedded help instructions associated with each feature configuration type to fully set up the associated option.

Master Configuration

Master Configuration is another source of global parameter records used by a system implementation.

Oracle Real-Time Scheduler uses a Global Configuration record that controls core system functions. This record must be set up for the system to properly operate. Refer to Appendix B for more information on when to set up this record.

Key concepts related to Master Configuration are discussed in this section. Refer to the embedded help for descriptions of the settings on the Master Configuration page.

Planning Horizon

The planning horizon, defined in the Global Configuration, is the number of days into the future to plan for crew shifts and periods of unavailability. This setting controls the number of day's worth of shifts to generate for perpetual crew shift templates. For example, if the planning horizon is set to 60, the system generates crew shifts and periods of unavailability (POU) sixty days into the future. Generation occurs in a sliding window; each day generates data for the last day incrementally. At any given time, the number of days of planned shifts and periods of unavailability is equal to the planning horizon value.

The planning horizon should be equal to or greater than the scheduling horizon, which is defined for a particular scheduler configuration. The scheduling horizon is the number of days into the future the scheduler will consider when scheduling crew shifts and activities. The scheduling horizon designed to limit the scope of data actually being scheduled by a scheduler. Oracle

recommends that the scheduling horizon be kept as short as is practical while still meeting customer business requirements. The optimum value for the scheduling horizon strikes a balance between long term planning and short term flexibility. See "Scheduler Configuration" on page 1 for more information about configuring schedulers.

Start Weekday Option

The Start Weekday Option in the Global Configuration allows you to specify which weekday marks the start of a week for your organization. The system uses this to properly display weekly-based information, such as shift weekly templates and calendars.

Chapter 3

Resource Management Configuration

This section provides configuration information for entities related to resource management.

- *Locations*
- *Skills*
- *Equipment*
- *Mobile Worker Types*
- *Vehicle Types*
- *Crew Types*
- *Dispatcher Types*
- *Mobile Workers*
- *Vehicles*
- *Crews*
- *Dispatchers*
- *Shift Cost Profiles*
- *Shift Weekly Templates*
- *Periods of Unavailability*
- *Breaks*
- *Non Productive Tasks*

Locations

You can define a set of common locations, such as service centers or business offices, that can be selected and referenced later. For example, predefined locations can be used as logon and logoff locations for crew shifts or as meeting locations for periods of unavailability. Locations save you time and simplify data entry.

A location record defines the location's name and description, the physical address of the location and its geocoordinates. The location address must be geocoded so the scheduler can direct crews to and from a common location.

The Location business object provided with the base package is associated with an algorithm that calls the address geocoding service (defined on the installation options record) to geocode the location's address (convert the address into a latitude/longitude pair), requesting the highest geocoding quality level. Refer to Appendix C for more information about the address geocoding algorithm service on installation options.

Skills

A skill indicates the ability of a mobile worker to perform certain types of work. Mobile workers can have skills; mobile worker types can have assumed skills; activity types can require skills.

A skill record defines the skill type and description, and indicates whether or not the skill has an effective period, such as a licensed-based skill. If a skill has an effective period, you will be required to enter the effective dates when you associate the skill with a mobile worker.

Equipment

Equipment is any device or apparatus used to perform certain types of work. Vehicles can carry pieces of equipment needed to perform certain types of work; vehicle types can have assumed equipment; activity types can require equipment.

An equipment record defines the equipment type and description. If equipment has an effective period, you can enter the effective dates when you associate the equipment with the vehicle.

Mobile Worker Types

Mobile worker type records are like job descriptions. They define the skills that mobile workers of a particular type are assumed to have. Assumed skills can be overridden for an individual mobile worker.

The mobile worker type definition also controls what business object is used when creating a new mobile worker of that type. Thus, it dictates the business rules governing content and behavior common to all mobile workers of this type.

Mobile Workers

A mobile worker is an individual workforce resource that performs work as part of a crew. Mobile workers can have skills, indicating their ability to perform certain types of work.

A mobile worker record defines the mobile worker type, the skills possessed by the worker (which default to the assumed skills of the mobile worker type), the service areas in which the mobile worker typically works, and other attributes. You can also specify the *Relative Efficiency*, which defines the mobile worker's efficiency relative to expected efficiency for mobile workers of this type. See *Entity Parameters* in *Appendix A*: *Scheduler Parameter Table* for more information about scheduling factors.

> **Note**: A mobile worker's home address may be used as a shift logon or logoff location, if the mobile worker is the only person on that crew. In this situation, the home address would need to be geocoded. For this reason, the mobile worker business object provided with the base package is configured with an algorithm to geocode the mobile worker's address and request an exact match on the address. See *Address Geocoding* for more information about this algorithm.

Vehicle Types

Vehicle type records define the equipment that vehicles of a particular type are assumed to have. Assumed equipment can be overridden for an individual vehicle.

The vehicle type definition also controls what business object is used when creating a new vehicle of that type. Thus, it dictates the business rules governing content and behavior common to all vehicles of this type.

Vehicles

A vehicle is a workforce resource used by a crew to perform work. Vehicles can carry pieces of equipment needed to perform certain types of work.

A vehicle record defines the vehicle type and the equipment possessed by the vehicle (which defaults to the assumed equipment of the vehicle type). You can also specify the *Relative Speed*, which defines the vehicle's speed relative to the expected speed for vehicles of this type. See *Entity Parameters* in *Appendix A*: *Scheduler Parameter Table* for more information about scheduling factors.

Crew Types

A crew type record defines a particular type of crew, such as a one-person crew or a multi-person crew. It also controls what business object is used when creating a new crew of that type, thus dictating the business rules that govern content and behavior common to all crews of this type.

Crews

A crew is a uniquely named group of mobile workers and vehicles that perform work in shifts. A crew shift is a planned period of time in which a crew (one or more mobile workers and vehicles) is scheduled to perform work. The specific mobile workers and vehicles allocated to a crew may vary from one shift to another, and a crew may perform different types of work and cover different service areas on different shifts. Therefore, crew allocation is defined for crew shifts, not for crews. Likewise, activities are scheduled to crew shifts, not to crews.

Dispatcher Types

A dispatcher type record defines a particular type of dispatcher. It mainly controls what business object is used when creating a new dispatcher of that type, thus dictating the business rules that govern content and behavior common to all dispatchers of this type.

Depending on your organization needs, you may need only one or very few dispatcher type definitions.

Dispatchers

A dispatcher is a user who manages day-to-day field operations.

A dispatcher record defines the dispatcher type, default date range for monitoring, dispatcher areas allowed to be monitored, and KPIs to be monitored. It also specifies whether or not the dispatcher can change certain settings when logging on to a shift.

Crew Shifts

This section describes key concepts and configuration information related to crew shifts.

Crew Shift vs. Crew Shift Type

A crew shift, sometimes referred to simply as a shift, is a planned period of time in which a crew (one or more mobile workers) is available to work. This should not to be confused with the common meaning of 'shift,' which typically refers to the time of day during which a crew performs work, such as first shift or second shift. A crew shift refers to a specific date and time planned for a specific crew to perform work.

Every crew shift is associated with a crew shift type (such as first shift, second shift, etc.) that controls what business object is used when creating a new crew shift of that type. Thus, it dictates the business rules governing content and behavior common to all shifts of this type.

Scheduling Details are Defined on the Shift

The specific mobile workers and vehicles allocated to a crew may vary from one shift to another. In the same way, a crew may perform different types of work and cover different service areas on different shifts. A crew may also start and end their route in different locations on different shifts. Therefore, crew resource allocation as well as all other scheduling information are defined for crew shifts, not for crews. Likewise, activities are scheduled to crew shifts, not to crews.

Crew Capabilities

The scheduler is encouraged to assign work to crew shifts where the crew possesses the capabilities (skills and equipment) necessary to perform the work. The overall skills and equipment associated with a crew on a specific shift are derived from the resources comprising that crew on that specific shift.

When a crew starts a shift, the shift's resource allocation may differ from its planned allocation. Resource allocation may even change throughout the shift. These changes are communicated to the scheduler, which may reconsider the planned work for the shift and rebuild the shift schedule.

Once the shift has started, any change made to its crew's resource allocation is recorded for reporting and other business needs.

> **Note**: A crew and any of its allocated resources can only be part of one shift at any given point in time. This means you cannot plan shifts for the same crew or mobile worker or vehicle to overlap in time.

Controlling Work Assignment

While a crew must be appropriately skilled and equipped to be assigned certain work, you can further plan and affect the work scheduled to a crew on a specific shift to meet your business needs as described below.

Primary Function

Types of work may be classified by service class. You can designate one of these service classes as the primary function of the crew on a specific shift. This encourages the scheduler to prefer work of that class, if any, over other classes of work. You can also list specific service classes of work that the crew should not be assigned on that shift, regardless of whether they are capable to perform that work.

Reserve Capacity

You can also reserve a certain percentage of the shift time for a specific service class of work, up until a specified lead time before the shift starts. Once started, the reserved time is released and made available to all other classes or work. For example, consider that your business is entering a seasonal busy cycle, and you want to reserve more time for emergency work during this period. Reserve capacity settings are defined on the Crew Shift and Crew Shift Template portals.

Crew Shift Logon / Logoff

A crew shift specifies a known A location record defines the location's name and description, the physical address of the location and its geocoordinates. The location address must be geocoded so the scheduler can direct crews to and from a common location. , typically a service center, where the crew is planned to start and end their work that day. A single-person crew can start and/or end their shift at the mobile worker's home address.

You may specify a maximum travel time the crew is expected to spend traveling from/to the logon/logoff location on their own time. You can also reserve time at the beginning and/or end of the shift for performing review and preparation work.

This information is used by the scheduler to properly plan the schedule or tasks assigned to the crew on that specific shift.

Shift Cost Profiles

Shift cost profiles are predefined shift-based cost factors that are used as multipliers against global costs. You can assign a shift cost profile to one or more shift templates or to individual shifts. Each shift is associated with a single shift cost profile. The following scheduling cost factors are defined in the shift cost profile and apply to all shifts associated with that profile:

- *Shift Cost*

- *Relative Travel Time Cost* (Time Cost)

- *Overtime Cost*

- *Shift Distance Cost*

- *Cost Idle*

See *Entity Parameters* in *Appendix A*: *Scheduler Parameter Table* for more information about scheduling factors.

The shift cost profile record defines the cost factor values and may include a list of associated service areas and/or service classes. These associations do not restrict the use of the shift cost profile in any way; you can associate any shift cost profile to any shift. Rather, the system uses these associations to recommend appropriate shift cost profiles when you are defining crew shifts.

Before you define shift cost profiles, study your organization's cost requirements and how they relate to specific service areas and/or service classes.

Drip Horizon

Once a crew starts their shift, you can configure the scheduler to automatically dispatch to the crew all or a subset of their work without requiring dispatcher intervention. The number of tasks to dispatch at any point in time, referred to as the drip horizon, may be controlled on the crew shift itself. Refer to the *Understanding Auto Dispatch* for more information.

Generate Shifts Using Templates

While crew shifts can always be manually created to address a specific situation, they are typically planned and generated ahead of time based on recurring templates. Refer to the next section for more information.

Shift Weekly Templates

This section describes key concepts and configuration information related to shift weekly templates.

What is Shift Weekly Template?

Typically, crew shifts are generated ahead of time from templates that define a cyclical weekly pattern. Resource planners can create shift weekly templates for specific crews, or they can create common templates and subscribe multiple crews to one template. Other attributes defined on both types of templates include effective period, and number of weeks in rotation.

General Shift Weekly Templates

The details for each shift within a cycle are captured on a shift template record. In other words, a shift weekly template is made of one or more shift templates that describe the recurring shift patterns. It is important to note that any detail you can define for an actual shift, you can also predefine on a shift template. Refer to *Actual vs. Template Shifts* for more information.

Once you have defined your templates, you must apply them so that actual shifts will be generated from them, based on the template's effective dates and your system's *Planning Horizon*. For example, you might define a weekly template for Crew A in which the crew works its regular 9-5 shift on M-F every week and also works an on-call shift on Saturday and Sunday every third week. Thus, the shift weekly template would rotate every 3 weeks. To configure this, you would do the following:

- Create a shift weekly template for Crew A.

- Indicate whether the template is perpetual (has an effective date but no end date) or has an effective period (start and end date). For example, if the template defines shift patterns to be used only during the summer months, enter a start and end date. If shift patterns for this crew remain the same throughout the year, do not enter an end date. For perpetual templates, the Planning Horizon parameter, defined in the master configuration, controls how many shifts to generate.

- Specify the number of weeks in rotation. In the example above, this value would be set to 3.

- Define crew shift templates for each day the crew will work during this rotation. (You can create one template and then duplicate it to save time.) In the example above, you would define shift templates for Monday through Friday of each week, and also for Saturday and Sunday of the third week only.

- Use the Preview feature to preview the shifts that would be generated if you applied the template.

- Apply the template. The system will automatically generate shifts based on the settings defined.

  Once applied, the system will continue generating shifts to meet the planning horizon. Refer to the *Shift Weekly Template Monitor* section for more information on this background process.

Common Weekly Templates

Resource planners can also set up common weekly templates which can be shared across multiple crews. This is helpful for the efficient generation of shifts in organizations where there are sometimes hundreds or possibly even thousands of crews that follow the same weekly shift pattern. The resource planner can subscribe crews to the common template, then shifts generated from the common template are exactly the same and follow the same timing in the rotation except for crew specific details such as allocated resources.

Common templates function the same way as general shift templates in almost every way except that the shift generation process uses the shift template merged with any shift related information defined on the crew level to generate the shifts. The process calls the Prepare Shift Data plug-in associated with the crew business object to perform the overlay of information on the crew before using it to generate shifts. Refer to Appendix C for more information on the Prepare Shift Data plug-in.

Actual vs. Template Shifts

Regardless of whether it is general or common, a shift template defines the details needed to generate actual shifts based on that template. This means the shift template's business objects has to share a common data structure with the corresponding business object used to create actual crew shifts based on that template.

A shift template is implemented as a crew shift entity that is simply marked as a template rather than an actual shift. Sharing the same maintenance object makes it easier for template and actual shifts to capture same information.

As mentioned earlier, Crew Shift Type defines the business object used to create actual shifts of this type. It also specifies the corresponding business object used to create shift templates of that type. This means a shift template can only create shifts of its same type.

Planning Breaks

When you define crew shift templates, you can plan one or more break types for the shifts. When the shifts are generated, the system automatically creates breaks and incorporates them into the crew shift schedule. See *Breaks* for more information.

Planning Non Productive Tasks

When you define crew shift templates, you can plan one or more non productive task types for the shifts. When the shifts are generated, the system automatically creates non productive tasks and incorporates them into the crew shift schedule. The location of the non productive task is taken from the crew record at shift generation time. See Non Productive Tasks for more information.

Shift Weekly Template Monitor

The Shift Weekly Template Monitor batch process controls how often shifts are generated going forward to meet the planning horizon, and thus how many days' worth of future shifts will exist at any time. For example, you can choose to generate shifts once a week, once a day, or multiple times in a day. If you choose to generate shifts once a week, then the system will generate seven more days' worth of shifts each week to meet the planning horizon; if you choose to generate shifts once a day, the system will generate one more day's worth of shifts each day.

Review the default settings for the monitor process, and schedule shift generation as often as appropriate to meet your organization's needs.

Shift Generation Algorithm

The logic used to generate shifts from templates resides in an algorithm associated with the shift template business object. This algorithm is invoked when the shift weekly template to which the shift template belongs is initially applied and also by the shift weekly template monitor process on an ongoing basis. The algorithm is also called if you make changes to a shift template, for which the associated shift weekly template has already been applied, in order to adjust future generated shifts to match recent changes.

Review the logic defined in the base algorithm to determine if it is adequate for your needs. See *Generate Crew Shifts* in *Appendix C: Algorithm Entities* for more information.

Periods of Unavailability

This section describes key concepts and configuration information related to periods of unavailability (POUs).

What is a POU?

A POU is a period of time during which one or more resources are planned to perform a task other than regular work. The POU record defines the date, time, and duration of the period of unavailability as well as the list of resources to which the event applies.

POUs might be configure to handle the following types of non-work related situations:

- meetings

- training

- jury duty

- vacation time

- other leave

A POU is of a specific type. All resources associated with a specific POU must be of the same resource class as defined on the POU type.

Crew-related POUs may also define a list of associated service areas. This instructs the scheduler to incorporate the POU in the shift schedule of attending crews only if they are planned to work in the specified service areas.

POU type also controls what business object is used when creating a new POU of that type. Thus, it dictates the business rules governing content and behavior common to all POUs of this type.

What is a POU Task?

A POU task is a way of tracking a crew's attendance of a POU event. See *POU Task Types* for information about configuring task types.

Recurring POUs

If the type of event reoccurs (such as weekly meetings), POU details are captured on a POU template record, which defines how often the event occurs and other attributes. When a POU template is initially saved, the system generates POUs from the template automatically. If this is an ongoing event (i.e., the POU template does not specify a specific expiration time), the system only generates POUs to meet the *Planning Horizon*. The *Period of Unavailability Monitor* is responsible for generating more POUs going forward.

You can only create recurring POUs if the POU Type is defined to support POU templates. Refer to the next section for more information about POU templates.

Actual vs. Template POUs

A POU template defines the details needed to generate actual POUs based on that template. This means the POU template's business objects has to share a common data structure with the corresponding business object used to create actual POUs based on that template.

A POU template is implemented as a POU entity that is simply marked as a template rather than an actual POU. Sharing the same maintenance object makes it easier for template and actual POUs to capture same information.

As mentioned earlier, POU Type defines the business object used to create actual POUs of this type. It also specifies the corresponding business object used to create POU templates of that type. This means a POU template can only create POUs of its same type.

Period of Unavailability Monitor

The Period of Unavailability Monitor batch process controls how often POUs are generated going forward to meet the planning horizon, and thus how many days' worth of future POUs will exist at any time. For example, you can choose to generate POUs once a week or once a day. If you choose to generate POUs once a week, then the system will generate seven more days' worth of POUs each week to meet the planning horizon; if you choose to generate POUs once a day, the system will generate one more day's worth of POUs each day.

Review the default settings for the monitor process, and schedule POU generation as often as appropriate to meet your organization's needs.

POU Generation Algorithm

The logic used to generate POUs from templates resides in an algorithm associated with the POU template business object. This algorithm is invoked when the POU template is initially saved and also by the POU template monitor process on an ongoing basis. The algorithm is also called if you make changes to a POU, so the system can adjust future generated POUs to match recent changes.

Review the logic defined in the base algorithm to determine if it is adequate for your needs. See *Generate POUs* in *Appendix C: Algorithm Entities* for more information.

Breaks

A break is a planned time period within a crew shift during which the crew is eligible for taking a break. During a break, the crew may not carry out work or travel. Unlike a crew-related period of unavailability that takes place in a specific location, a break does not have a defined location.

A break type defines the duration of a break and the duration of the break window, which is the period of time during which the break can be taken. For example, a 15-minute morning break might have a 60-minute window during which the break can be taken.

While you can manually add a break to an existing crew shift, breaks are typically incorporated into planned shifts at shift template design time. Refer to *Shift Weekly Templates* for more information.

Non Productive Tasks

A non productive task is a planned time period within a crew shift where a crew can take time to perform a non work related task such as restocking the truck. Non productive tasks are always associated to a location, and the crew cannot carry out work or travel during this allocated time period.

A non productive task type defines the duration of a non productive task and the duration of the non productive task window, which is the period of time during which the non productive task can be taken. It also defines an "offset" which determines how much time into the shift that the non productive task can take place. For example, a 15-minute morning non productive task might have a 60-minute window during which the non productive task can be taken and it can only occur after the crew has been working for at least 2 hours.

While you can manually add a non productive task to an existing crew shift, non productive tasks are typically incorporated into planned shifts at shift template design time.

On the template, the user only defines the task type and the earliest time the NPT should be scheduled after shift start. The NPT location is defaulted from the crew record.

Chapter 4

Service Management Configuration

This section provides configuration information for entities related to service management

- *Remark Types*
- *Appointment Booking Groups*
- *Priority Profile*
- *Service Areas*
- *Activity Types*
- *POU Task Types*
- *Override Time Windows*
- *Service Classes*
- *Dispatch Areas*
- *Alerts*
- *KPIs*

Remark Types

Standard remarks define different types of comments that can be entered when an activity is completed. Remark types are associated to activity types, indicating the types of remarks that can be entered when completing this type of activity.

Remark types can define common indications used by the ORS system itself and/or common remark types known to the host system and sent as part of completion details.

Appointment Booking Groups

Appointment booking refers to how the system handles requests from the host system for appointments and how it responds with available appointment windows. Appointment booking groups (ABGs) define one or more valid time windows for appointment booking. ABGs are associated with activity types, indicating the ABGs to use when processing appointment requests for activities of that type.

The host system uses the Appointment Booking XAI inbound service to request a list of available appointments for an activity prior to interfacing it. Requests from the host specify the time periods for which the scheduler should return a list of available appointment windows.

The system processes an appointment booking request from the host as follows:

- Using the appointment booking group specified on the request, identify the actual time windows during which appointments can be scheduled. If no appointment booking group is supplied by the host, the system uses the activity type's default appointment booking group.

- Geocode the proposed activity's address.

- Calculate its arrival time window (ATW) based on the requested Effective Time Windows (ETWs). This is done by invoking the **Update Time Windows** algorithm associated with the activity business object defined on the proposed activity's type.

- The information gathered above is sent to scheduler(s) responsible for appointment booking.

- Appointment information is returned. If there are multiple schedulers for appointment booking, the first response is used.

Priority Profile

A priority profile is a predefined set of allocation factors used by the scheduler to prioritize the scheduling of an activity relative to other activities. Priority profiles are associated to activity types, indicating the priority settings for all activities of that type.

A priority profile indicates whether allocation is mandatory or optional for activities associated with this profile. If the allocation rule is set to Optional, allocation is driven by costs. If the allocation rule is set to Mandatory, the activity will be allocated regardless of the cost.

The priority profile record also defines the period within an activity's arrival time window that the activities should be scheduled. This period is referred to as the allocation priority window. An allocation priority window can be defined in any of the following ways:

- Allocation priority window aligns with the ATW window. This is typically the case for appointed work, where the priority remains constant throughout the lifespan of the ATW window.

- Allocation priority window is toward the beginning of the ATW. This is typically the case for short-cycle, SLA-oriented work, where the activity is critical for the first few days and then diminishes.

- Allocation priority window is toward the end of the ATW. This is typically the case for long-cycle work, such as inspections, where the priority is low until the end of the ATW when a backlog may occur.

Thus, you can define an allocation priority window as an offset from the ATW start date, as shown in the following figure:
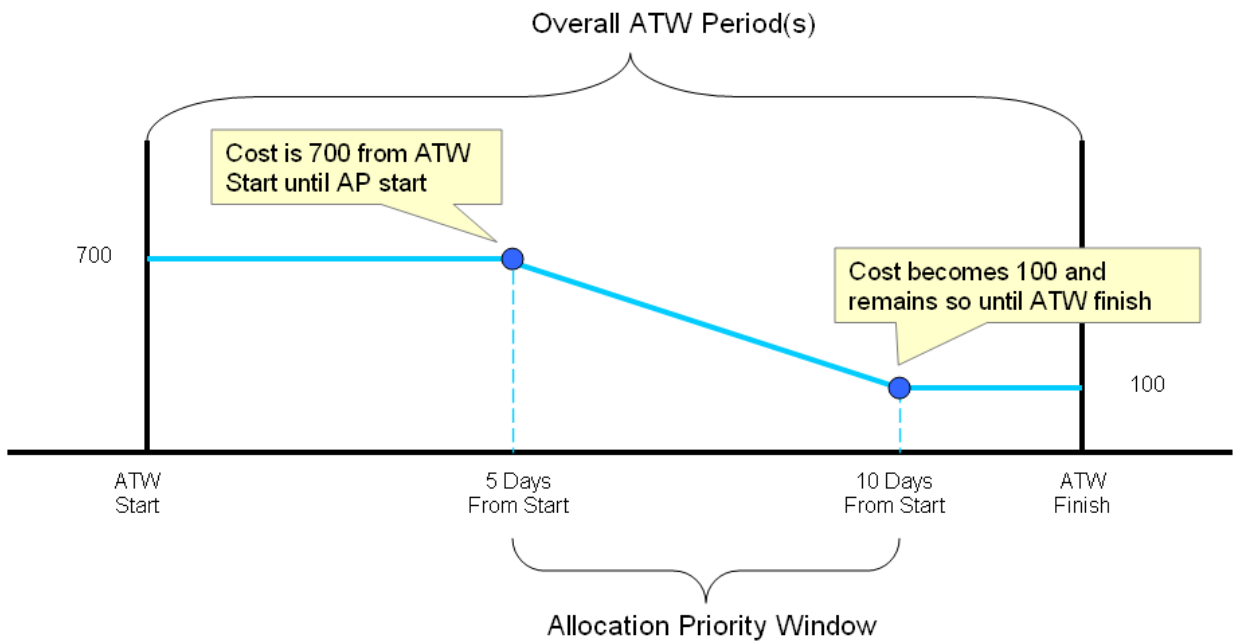
You can also define an allocation priority window as an offset from the ATW Finish date, as shown in the following figure:
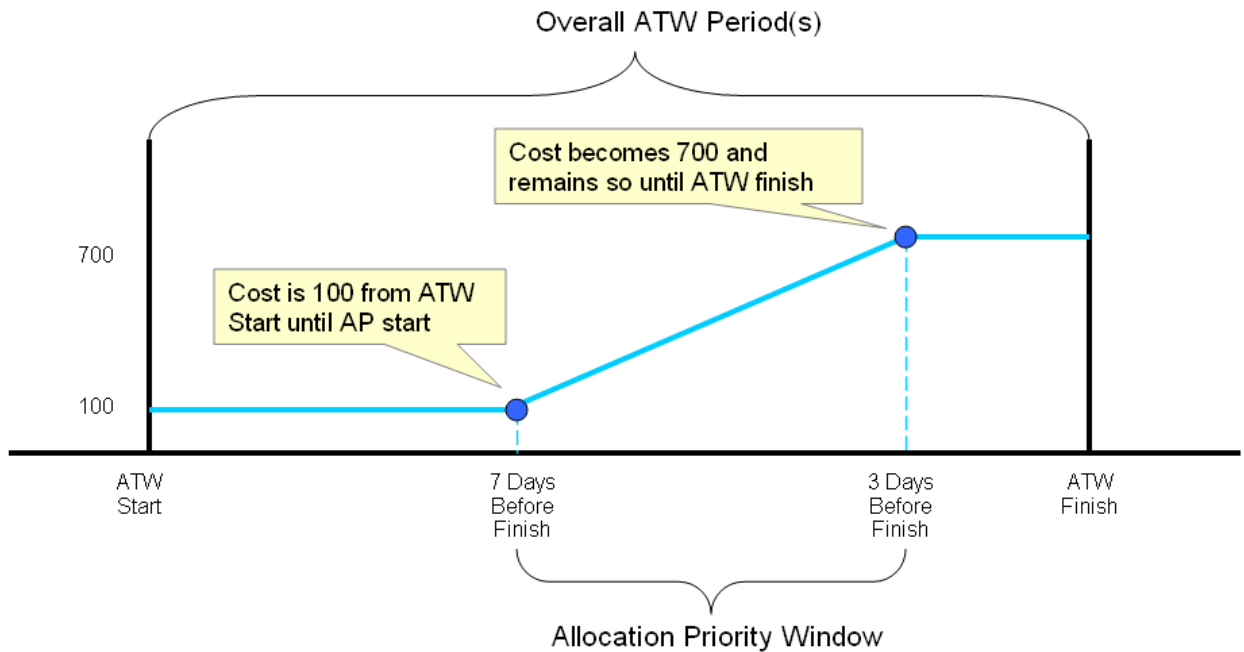
Note: If you define an allocation priority window and set the allocation rule to Mandatory, then allocation becomes mandatory when the allocation priority window end date is reached, rather than retaining the final calculated cost until the ATW finish (as it would if the allocation rule were set to Optional).

In addition to the allocation rule and offset options, the priority profile record defines the sort sequence, initial priority, and final priority.

Service Areas

A service area defines the logical boundary of an organization's territory. Typically, not all crews are allocated to do work in the whole territory and not all dispatchers monitor the whole territory. Service areas are used to divide a territory into broad areas of service.

Service areas can be based on geographic regions, services provided, business units, or any other breakdown that is meaningful for your organization. Mobile workers may typically work in specific service areas, which comprise their base service areas. Crew shifts may be limited to working in specific service areas, and dispatchers may be responsible for monitoring specific service areas.

Each activity is associated with a single service area. If it is not explicitly provided by the host system, the rules to determine an activity's service area reside in a plug-in on the activity business object so they can be customized as needed. Refer to the *Determine Service Area* entity in Appendix C for more information.

Activity Types

This section describes key concepts and configuration information related to activity types.

Basic Concepts

This section describes concepts that are important to understand before you configure activity types.

What is an Activity?

An activity is a task, typically requested by a host system, to be performed at a particular geographic location. An activity may have a site address and a service address. Only the site address is used by the scheduler to direct the crew to the activity location.

An **activity type** defines skills and equipment required to work activities of this type as well as details about how the activities should be dispatched, prioritized, and scheduled.

What is an Assignment?

An **assignment** is a copy of an activity that is created when an activity is dispatched and, for MDT crews, is also sent to the assigned crew for recording status updates and completion details.

If changes are made on the server to an activity, the system determines which changes are relevant to the crew and sends updated information to the crew accordingly. Thus, the assignment is kept in synch with its original activity. Likewise, when changes are made on the mobile, they are sent to the server immediately (or when a connection is available).

Activity and Assignment Lifecycles

An assignment is created when the activity lifecycle reaches a point in which work should be dispatched to the assigned crew. The lifecycle of an assignment affects the lifecycle of its activity. For example:

- When an assignment is completed, the original activity is automatically completed, provided that one and only one finalized assignment is received for the activity.

- If an activity's current assignment is returned and no other assignment is complete, the activity is rescheduled.

- If all assignments are final and only one is complete, then the activity is completed.

- If all assignments are final and more than one is complete, the activity's status is set to collision detected.

  > **Note**: The lifecycles of activities and assignments are fully described and depicted in the business object records. Refer to the Lifecycle tab of the

Business Object portal for complete state descriptions; refer to the Summary tab for a visual representation of the lifecycles.

The following diagrams illustrate the lifecycles of each object and how they affect each other. (Refer to the maintenance portals for assignment and activity business objects for complete descriptions of the lifecycle states.)

Activity states are depicted in the following diagram:
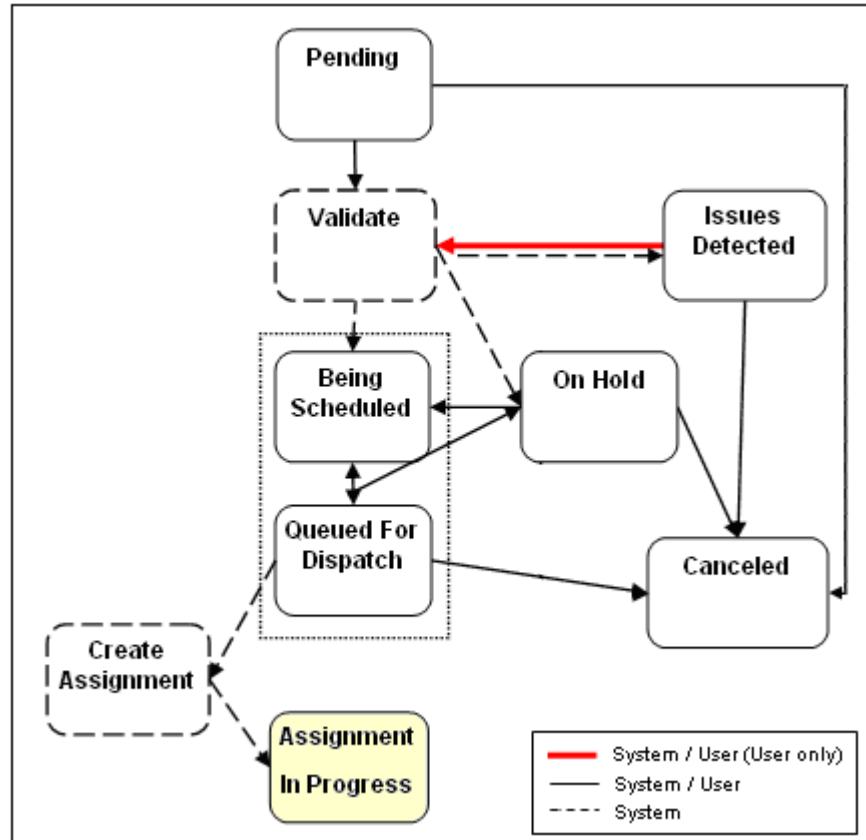
Assignment states are depicted in the following diagram:
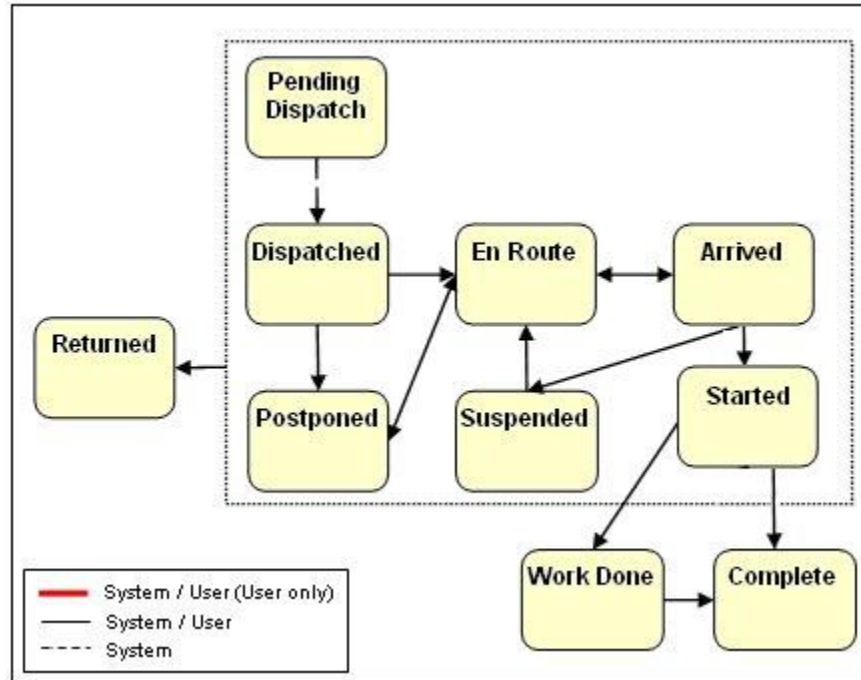
Incomplete State

All activity types have an Incomplete state as part of their lifecycle. This state provides a way for the crew to communicate to the scheduler that the work could not be completed for reasons beyond its control. For example, if the crew attempts to access a building but the building is not at the specified location, the crew can mark the assignment as incomplete. This bypasses any completion rules and allows the crew to move on to the rest of its work.

Work Done State

All activity types also have an Work Done state as part of their lifecycle which can be used by crews that are not connected to the server, such as SMS crews, to mark their work as done and move on without sending either a complete or incomeplet status to the server. This state queues the work so taht the crew can enter the required completion information later when they are logged on.

Activity and Assignment Business Objects Control Everything

The lifecycle, hierarchy, options and algorithms defined for the activity and assignment business objects control what activity information is received from and sent back to the host, what data is sent to and received back from the crew, what dispatcher actions are supported and what happens when each action is invoked, what crew actions are supported and what happens when each is invoked, and virtually every aspect of activity and assignment processing.

If you need to create a new activity business object, it is best to create it as a child, or sub class, of the activity business object provided with the base package. Thus, the new business object will contain all the core business rules and behavior of its parent object, saving you considerable time and effort.

The following business object algorithm entities are associated with the activity and assignment business objects.

- Activity Expiration

- Dispatcher functions, such as Allocate, Cancel, Defer, and Unassign (see *Common Dispatching Functionality*)

- **Set Alternate ID**
- **Update Time Windows**
- **Task Synchronization**

Refer to *Appendix C: Algorithm Entities* for more information about these algorithms.

Designing Your Activity Types

The base product can be used by many different types of businesses and can support a diverse set of activities ranging from field service orders, to installations and inspections, and even home deliveries. The underlying processes of scheduling, routing, and data transfer are quite similar; the primary differences are in the details, such as the implementation-specific data to be captured and how that data should be validated and processed.

Activity types are unique in that they typically require some amount of customization to adequately represent the data and business rules associated with a particular type of work. The base system provides a standard activity business object that can serve as the foundation for all custom activity types. This business object provides the basic logic to manage the activity lifecycle, handle scheduling and routing, and transfer data between the host and the mobile devices.

On top of this strong foundation, each implementation can build custom business objects by extending what is in the base system, rather than creating them from scratch. The customization process should begin with an analysis of what already exists in the base system, followed by a thoughtful design phase that identifies exactly what additional information and processes are needed. Only when the design phase is complete should an implementer begin using system configuration tools to create or customize system entities.

Below is a high-level summary of the steps required to create custom activity types for your organization.

1. Review the activity business object provided with the base package to familiarize yourself with its logic and related entities. Note that the base product's assignment business object is referenced as an option on the activity's business object. Review it as well.

2. Review and optionally customize activity-related algorithms. Refer to Appendix C for more information.

3. Identify all custom activity types required for your implementation.

   Note: You will typically need to create a different activity type for each type of work that requires different details to be sent from the host, completion details to be sent to the host, or validation logic. For example, one activity type might require that mobile workers be sent a payment history as part of the activity detail, while another might send a prior service record. Similarly, one activity type might require that mobile workers capture a particular kind of measurement or device reading, while another might capture customer satisfaction ratings; both types would require custom logic to validate the captured data.

4. The base package provides the To Do Type, M1-TSKTD, to capture activity data issues. Review its details. If you want the system to generate To Dos, associate it with To Do roles as per your business needs. Refer to the Framework documentation for more about To Do Types and Roles.

5. For each activity type you identified above, do the following:

   - Identify all data to be received from and sent back to the host.

   - Design the UI maps to be used to display and capture data. You will need to design screens for the server application and, if MDTs are used in the fields, the mobile application as well. Remember that you can clone the screens associated with the base business objects and then modify them as needed.

- Identify all behavior needed to extend the base business object for this activity type, including algorithms that will be needed to handle validation of any data to be captured, monitor processes, etc. Make a list of these items so you can be sure to address all of them in the development process.

- Identify the data that needs to be synchronized to the crew. Do not include any data the crew does not need to know in order to perform their work.

Defining Your Activity Types

The following describes the high-level steps needed to define a new activity type:

1. Define an activity business object that represents all data and behavior required for activities of that type.

   The activity business object should be defined as a child of the base activity business object; thus, all events associated with the parent business object are automatically supported for its child business object.

   The business object's schema should include the base activity business object schema as well as any incoming details sent from the host, but not completion details.

   Finally, associate the UI Maps, custom algorithms, and service scripts required to display or capture data, perform validation, or handle any other custom business processes.

2. Define a data area for the dispatch details that need to be synchronized to the crew, as identified during the design phase. The data area should include the base dispatch details data area associated with the base assignment business object, as well as specific details relevant to this type of activity. The **Task Synchronization** process uses this data area to dynamically send only this subset of details to the crew.

3. Define an assignment business object that represents all data and behavior required for an assignment of this activity type.

   The assignment business object should be defined as a child of the base assignment business object.

   Its schema should include the base assignment business object schema as well as completion details relevant to this type of activity to be sent to the host.

   As with the activity BO, associate the UI maps, custom algorithms, service scripts, etc., required to display or capture data, perform validation, or handle any other custom business processes.

   Reference the dispatch details data area described in the previous step as a business object option on your assignment business object.

4. Reference the assignment business object created in the previous step as a business object option on your activity business object.

5. Define an activity type record (using the Task Type portal, accessible from the Admin menu). When prompted, select the appropriate activity business object defined earlier.

   For each activity type record, you will need to provide the following information. Consider each item carefully before you create the record.

   - The host system for this activity type

   - How long an activity of this type typically takes to complete

   - Whether or not this type of activity should be automatically dispatched

   - Whether or not an acknowledgement from the crew should be required for this type of activity (emergency activities typically require acknowledgement)

- The priority queue and priority profile (priority profile defines attributes such as sequence, whether the activity type is mandatory or optional, and whether or not an offset is use. Note that emergency activities must also be set as mandatory.)

- The appointment booking group to use

- Any scheduling cost factors

- Whether or not activities of this type need to be allocated to the assigned crew at some predefined time prior to shift start

- How expired activities of this type be handled. The activity business object you created earlier for this activity type

- Skills and equipment required to complete activities of this type

- Equipment required to complete activities of this type

- Completion remarks associated with activities of this type

- To Do setup information

Activity-Related Scheduling Cost Factors

The following scheduling cost factors are defined at the activity-type level:

- *Relative Shift Promotion Cost*

- *SLA Priority*

The scheduler uses the following activity-based parameters when assigning tasks to crew shifts and optimizing crew shift schedules:

- *Relative Late Cost*

- *Relative Window Cost*

- *SLA Flexibility*

POU Task Types

If the resource type associated with a POU is set to crew, the system will automatically generate POU tasks for all crews in the specified service areas and incorporate them into the crew's shift schedule. A POU task type record defines the POU task business object to use when creating POU tasks. The system uses a single type of POU task, which is defined on the *Master Configuration*, to generate all POU tasks.

Override Time Windows

An override time window is a period of time during which the default arrival time window for activities of a particular type is overridden. Override time windows (OTWs) and public holidays can both be used to define such restrictions. If at least one of an activity's effective time windows overlaps with an override time window defined for the activity type, then the scheduling restrictions defined in the override time window are applied to the activity.

> **Note**: Override Time Windows are typically created on an as-needed basis, not during initial implementation.

Logic to apply overrides and recalculate time windows of affected activities resides in the **Update Time Windows** algorithm entity associated with the activity business object. Refer them to Appendix C for more information.

Use the Override Time window maintenance portal to define override time windows. An override time window record defines the activity type, effective time window, time usage (allowed or not allowed during this time), and service areas to which the override applies.

Service Classes

Service class is a broad categorization of activity types. A service class record defines the service class name and description, and a list of associated activity types.

Service classes may be used to control work assigned to crew shifts by the scheduler. Refer to *Controlling Work Assignment* for more information.

Service classes may also be used to restrict the crews and activities monitored by a dispatcher. Refer to *Dispatch Areas* for more information.

Dispatch Areas

A dispatch area is a predefined set of service areas and service classes that can be monitored by one or more dispatchers. A dispatcher record identifies one or more dispatcher areas the dispatcher is allowed to monitor. When a dispatcher logs on to start a shift, they can select which of the authorized dispatch areas they will monitor during that shift. The system determines which activities and crews the dispatcher is responsible for monitoring during their shift based on the selected dispatch areas:

- A crew's shift matches a dispatch area if at least one of the shift's service areas and at least one of its allowed service classes are included in the dispatch area definition.

- An activity matches a dispatch area if its service area and at least one of its service classes are included in the dispatch area definition.

- It is enough for an activity or shift to match one of the monitored dispatcher areas to be monitored by the dispatcher.

- A single dispatch area may be monitored by multiple dispatchers.

Before you set up your dispatch areas, study the distinct combinations of service areas and service classes to be monitored by dispatchers.

Alerts

This section describes key concepts and configuration information related to alerts.

What is an Alert?

An alert is a situation in which predefined conditions have been met that require user attention or intervention. An alert could be raised for a crew's shift or a task.

The system automatically assigns an alert to a single logged-on dispatcher based on configurable business rules. Once assigned, the alert appears on the user's Alert Queue dashboard zone.

You can configure alerts to be recycled if the assigned user doesn't acknowledge the alert within a configurable amount of time. Alerts assigned to a user are always recycled when the user logs off.

You can configure alerts to snooze for a configurable amount of time once the user acknowledges the alert. The alert is dropped from the user's alert queue while the user tries to address the issue, and will reappear after that amount of time if the conditions are still true for the alert.

To make sure that the underlying condition(s) of an alert are resolved properly, you can design business rules to constantly evaluate and finally close the alert automatically.

An alert can also be configured to expire, i.e., be closed, after a certain amount of time.

Alert Type Controls Everything

Alert type defines attributes common to all alerts of that type. Alert type attributes include the creation and evaluation algorithms, the threshold for creating alerts of this type, whether or not alerts of this type can be manually closed, and the length of time before the alert expires, is recycled, or is allowed to snooze.

Alert type also defines the business object used to create and control the structure and behavior of alerts of this type.

The base product provides a set of business objects and algorithms that can be used to create and evaluate different types of alerts. The base product does not provide the actual alert type records. At a bare minimum, an implementation needs to create alert type records that reference the creation and evaluation algorithms and the alert business objects provided in the base for each alert type. An implementation can also create additional entities to support other types of alerts.

> **Note**: Some alerts, like the missed appointment alert, do not have evaluation algorithms because the condition that causes the alert to be generated is already in the past. There is no condition to research or resolve; the alert is simply a notification.
>
> Likewise, some alerts do not have creation algorithms because they are generated when a system event, such as a synchronization timeout, occurs. The activity cancellation alert is another example; it is raised whenever the host requests to cancel an activity that is already dispatched.

Audible Alerts

You can configure alerts to make a sound to help get the attention of the dispatcher when they appear on the alert queue. Specify the sound files to use with master configuration sound play parameters.

Designing Your Alert Types

The following is a high-level summary of the steps involved in designing alert types for your organization.

1. **Understand the business conditions for which to raise alerts of this type.** If the alerts are to be raised upon a specific event, you need to design an algorithm that will be triggered by the event to create an alert. Many alerts are not event-driven, but rather are raised based on a situation that is determined by frequent analysis of the data. For such alerts, an Alert Creation algorithm needs to be developed based on the business rules. The Alert Creation algorithm is called by the monitor plug-in associated with the alert type business object. For more information, refer to the alert batch controls section and to Appendix C.

   Most Alert Creation algorithms require some kind of threshold. For example, you might want to raise a crew idle alert when a crew has been idle for more than a specific length of time. The idle time, in minutes, may be defined on the alert type.

2. **Determine whether alerts of this type should automatically expire after some time.** The expiration time, in minutes, may be defined on the alert type. If not specified, alerts of this type do not expire.

   An alert should only be allowed to expire if it is simply a notification and does not require a dispatcher to take any specific action.

3. **Determine whether the system should automatically recycle an alert of this type if it is not acknowledged in a timely manner.** The recycle time, in minutes, may be defined on the alert type. If not specified, alerts of this type are never recycled based on time; however, they will still be recycled when the assigned dispatcher logs off.

4. **Determine whether an alert of this type should snooze for a configurable amount of time after a dispatcher acknowledges it.** As a general rule, you should allow an alert to snooze if it is important that the alert be closed only when the underling issue has been resolved, and you want to allow the dispatcher for more time to resolve it. When the snooze time expires, the system evaluates the underlying condition and closes the alert if the condition no longer exists. The snooze time, in minutes, may be defined on the alert type. If not specified, an alert of this type automatically closes once a dispatcher acknowledges it.

5. **Understand the business rules that allow the system to automatically close an alert of this type.** When applicable, design a corresponding Alert Evaluation algorithm and specify it on the alert type. The Alert Evaluation algorithm is called by the monitor plug-in associated with the alert business object. For more information, refer to the alert batch controls section and to Appendix C.

6. **Determine whether to allow dispatchers to manually close alerts of this type even if there is an evaluation algorithm.** There may be situations where an alert cannot be automatically closed by the evaluation algorithm. For example, if a crew's mobile device loses its connection for a period of time and the crew communicates via phone to the dispatcher, you might want to allow the dispatcher to close the alert so it will be removed from the queue.

   Note: It is a best practice to let the system evaluate and close all alerts. If you have to allow manual closure, please be sure to build this logic into the Alert Creation algorithm to avoid raising repeated alerts for same issue.

7. **Determine the priority of alerts of this type relative to other alerts**. The priority dictates the order by which alerts are presented to dispatchers. The priority may be defined on the alert type.

8. **Review the alert business objects provided with the base package to familiarize yourself with their logic and related entities**. Choose the one that's most appropriate for your alert type. If you need to, you can create a new alert business object with a different lifecycle and business rules to meet your organization's needs.

9. Define the alert type and all its attributes, including the alert business object and calculation algorithm.

10. Determine if the alert requires a sound to be played.

Alert Batch Controls

Alert creation and evaluation algorithms are called by a batch process.

- The **Alert Type Monitor Batch Control** is responsible for creating alerts. It is critical that this process runs frequently enough to raise alerts in a timely manner, so the underlying condition can be addressed promptly.

- The **Alert Monitor Batch Control** is responsible for evaluating alerts. It is important that this process be called frequently so that alerts can be closed promptly when the underlying condition is resolved.

The system provides functionality to support scheduling of a batch process as often as needed. If your organization uses a third-party application to schedule batch processes, you should configure the batch control type to be Not Timed. The system will not automatically run the batch process according to any schedule, allowing your process to handle scheduling. If you want the system to run the batch process at regular intervals, you should configure the batch control type to be Timed and then set the Timer Interval to control how often the process runs. For example, if the Timer Interval is set to 60, the batch process will run every 60 seconds.

Alert Queue

The Alert Queue is a dashboard zone where users can view and manage alerts. The Alert Queue zone should be enabled in User Preferences for all users who are responsible for monitoring alerts, and the auto-refresh settings should be set to refresh frequently.

Dispatchers can also identify a backup dispatcher in the Backup Alert Queue zone. This zone lists alerts of other dispatchers that the shift dispatcher is backing up. If used, this zone must be enabled just like the alert queue zone.

KPIs

This section describes key concepts and configuration information related to key performance indicators (KPIs).

What is a KPI?

A key performance indicator (KPI) is a quantifiable measurement that supports dispatcher decision making and in-day exception handling. For example, a KPI might measure the number of out-of-service crews or the number of appointments in jeopardy of being missed.

KPI Record Controls Everything

A KPI record defines what is being measured and how to measure it. It specifies the calculation algorithm to use and the threshold values used by that algorithm.

Some KPIs measure the same underlying conditions as the alerts raised by your organization. For those KPIs, the base product provides a KPI calculation algorithm that counts the number of alerts whose related activity or crew shift is in the user's monitor data scope.

For KPIs not related to alerts, a new plug-in must be created. For example, if you want to create a KPI that counts the number of emergency activities received or the number of breaks canceled, you would need to first create an algorithm to perform the calculation and then create a KPI record that referenced the calculation algorithm.

The KPI calculation plug-in spot supports two modes, Summary or List. If called in the Summary mode, the algorithm returns the summary value of the KPI. If called in List mode, it returns the detail of the KPI.

The base system does not provide any KPI records. An implementation is responsible for creating all KPI records and then associating them to Dispatcher records, indicating that the dispatcher can monitor the KPI on their CDI portal.

Designing Your KPIs

The following is a high-level summary of the steps involved in designing KPIs for your organization:

1. **Determine what the KPI will measure.** Identify whether the KPI is related to activities or to crews.

2. **Determine whether there is an existing alert type that measures the underlying condition.** If so, you can use the KPI calculation algorithm provided with the base package. If not, you must create an algorithm to calculate the KPI. The calculation algorithm may be defined on the KPI record.

3. **Identify the beginning of the normal, warning, and severe ranges for the KPI.** These values may be defined on the KPI record.

4. **Review the KPI business object provided with the base package to familiarize yourself with its logic and related entities.** If you need to, you can create a new KPI business object to meet your organization's needs.

Chapter 5

Scheduler Configuration

This section describes how to configure the scheduling component and includes the following topics:

- *Scheduling Basics*
- *Managing the Scheduling Process*
- *Scheduler Areas*
- *Scheduler Parameters*
- *Scheduler Configurations*
- *Setting Up Your Schedulers*
- *Using the MapEditor*
- *Scheduler Registry*
- *Scheduling-Related Algorithms*
- *Scheduler Monitor Process*

Scheduling Basics

The scheduling process automatically assigns tasks to crew shifts, matching field work requirements to crew capabilities and optimizing the routes traveled by crews. The goal of the scheduling process is to produce the most cost-effective schedule. The scheduling process considers a large number of configurable factors to arrive at the optimum schedule.

Scheduling Process

The scheduling process has the following basic inputs:

- **Tasks:** Tasks can be field activities, breaks, non productive administrative tasks, and crew related periods of unavailability (POUs), all of which are incorporated into the schedule. If using depot functionality the scheduler inserts visits to the depot as tasks. Refer to the depot functionality.

- **Crew Shifts:** Shift data includes crew capabilities, covered service areas and service classes, logon and logoff locations, and shift-based cost factors.

- Depots: If you are using depot functionality, the scheduler can manage shifts which include the location where the crew picks up and drops off goods.

- **Map Data:** All locations within the service territory are represented in the map data.

- **Static Control Data:** This includes cost controls and appointment booking groups.

The scheduling process assigns each task a scheduled date/time on a particular shift of a crew. As part of the optimization process, a task may be assigned, unassigned, and reassigned any number of times until its current scheduled shift starts. This happens when a crew logs on to the shift and the task is dispatched.

Dispatchers can manually override virtually all scheduling decisions made by the system.

Appointment Booking Process

Appointment booking refers to how the system handles requests from the host system for appointments and how it responds with available appointment windows.

The appointment booking process receives appointment requests from the host that specify the time periods for which the scheduler should return a list of available appointment windows. Refer to *Appointment Booking Groups* for more information.

Scheduling Parameters

Parameters affecting scheduling may be defined at three levels:

- Task-based parameters are defined at the activity or activity-type level.

- Shift-based parameters are defined in a shift cost profile or individual crew shift record.

- Scheduler (global) parameters are defined in a scheduler configuration record.

See Appendix A for a complete list of scheduler parameters.

Managing the Scheduling Process

The term scheduler refers to the system component that manages a single scheduling process. An implementation typically runs at least two schedulers to provide for backup, but may run additional sets of schedulers if required for scalability purposes.

Each uniquely named scheduler is configured to manage a single scheduling process. Scheduling process responsibilities include:

- Holding - does not schedule shifts or tasks, only captures tasks and shifts that cannot be assigned to any online scheduler. Only create one scheduler of this type.

- Request Handler Only - only handles requests

- Optimization - uses the scheduler's full optimization functionality, but does not handle requests.

- Optimization and Request - handles optimization and requests.

A holding scheduler does not schedule shifts or tasks. It is designed to capture tasks and shifts that cannot be assigned to any online scheduler. One and only one holding scheduler should exist. Any task or shift attached to a holding scheduler is evidence of a scheduling error.

Request Types

A scheduler that handles requests can handle the following types of requests:

- chooser – returns a list of recommended shifts for an activity

- appointment booking - returns a list of valid possible appointments for an activity

- conditional booking - confirms the schedule of an activity •

- crew route and directions – returns the street level route as well as driving directions for a list of tasks in a crew shift's schedule

Please refer to Appendix A for the parameters that must to be set up for requests.

Scheduler Configurations

Each scheduler is associated with a scheduler configuration that defines the scheduling parameters to be used by that scheduler during the scheduling process. These parameters include cost controls, performance and optimization settings, and settings to control logging, auto-dispatching, and other scheduler functions.

Each scheduler is also configured to cover a specific geographic area called a scheduling area.

Using Multiple Schedulers to Scale the Scheduling Process

The scheduling process is very complex and has high memory requirements that increase dramatically as the number of tasks being scheduled increases. Memory requirements also increase as the scheduling horizon (the number of days into the future that must be scheduled) increases. To achieve optimum performance without sacrificing scheduling efficiency, the system can be configured to use multiple schedulers at once and limit the number of tasks and shifts for which each scheduler is responsible.

Oracle provides general guidelines for calculating the optimum number of schedulers required for a particular installation:

- Define a variable, **L**, to be the maximum task load that can be scheduled by a single scheduler. 20,000 is the typical maximum.

- Determine the total number of tasks, of unique location, to be scheduled in a given scheduler area within the scheduling horizon.

- If the task count is less than L, then a single scheduler can cover the entire area. One extra scheduler will be required for backup.

- If the task count is greater than L, then the scheduler area must be covered by multiple schedulers working in parallel, and the area must be divided into smaller segments called optimization areas. Each scheduler is typically configured with a single optimization area.

- To calculate the number of optimization areas (schedulers) required for a large task count, use the following guidelines:

  - Though the typical maximum load (L) for a scheduler is about 20,000, schedulers configured with an optimization area should be designed for a smaller average load, typically 62.5% of L, which we call *oL*.

  - The total number of schedulers required for the scheduler area can be estimated as follows, where *n* is equal to the expected maximum task count:

    $((n/oL$ rounded up to the next whole number$) * 3) - 1$

The result is multiplied by 3 because three sets of optimization areas are typically required to manage a scheduler area with a large task count. Subtract 1 because the third optimization set can typically have fewer optimization areas than the first two sets. Please note that these rules are guidelines only; greater or fewer schedulers may be required depending on local geographies and particular scheduling requirements.

The table below shows an approximate number of schedulers required for various task counts, assuming the following base values:

- Maximum task count for a single scheduler: $L = 20,000$

- Maximum task count for a single scheduler linked to an optimization area: $oL = 12,500$ (62.5% of L)

- n = Task Count

Task Count (n) Low	Task Count (n) High	Scheduler Count	Explanation
0	20,000	2	If the load is less than the maximum task count, then one scheduler can handle the entire load, with an additional scheduler for backup.
20,000	25,000	5	n = 25,000 n/oL (25,000 / 12,500) rounded up = 2 2 * 3 - 1 = 5
25,000	37,500	8	n = 37,500 n/oL rounded up = 3 3 * 3 − 1 = 8
37,500	50,000	11	n = 50,000 n/oL rounded up = 4 4 * 3 - 1 = 11

Thus, every physical location of an area must be covered by at least two schedulers for small task counts (typically under 20,000) or three schedulers for large task counts (typically over 20,000).

> **Note**: The actual number of schedulers necessary may vary due to geographic constraints or particular scheduling requirements. Some geographic areas may be too large for a single scheduler to process. Determining the appropriate number of schedulers and defining optimum scheduling areas are essential to achieve maximum scheduling efficiency.

Scheduler Areas

A scheduler area defines a geographic area to be scheduled by a number of cooperating schedulers. A scheduler area may comprise one or more service areas. If a scheduler area has too many tasks for a single scheduler, it may be broken down into smaller segments called optimization areas.

Refer to the online help for instructions on creating and maintaining scheduler area records. The following sections provide conceptual information to help you design your scheduler areas before entering them into the system.

Multiple Service Areas within a Scheduler Area

A scheduler is limited to scheduling tasks within its scheduler area. Thus, the set of service areas comprising a scheduler area must at least match the most complex set of service areas covered by any given shift.

Consider a territory with eight service areas: N, E, S, W, NE, SE, NW, SW. Shifts scheduled during the day typically operate in just two service areas, while shifts scheduled during the night can be routed to any of the eight service areas:

- Day Shift (N): N, NE

- Day Shift (E): E, SE

- Day Shift (S): S, SW

- Day Shift (W): W, NW

- Night Shift: N, E, S, W, NE, SE, NW, SW

Since the scheduler area must cover the most complex set of shift service areas, the Night shift is used to model the scheduler area:

Scheduler Area: N, E, S, W, NE, SE, NW, SW.

Using Optimization Areas

If a scheduler area has too many tasks for a single scheduler to manage (typically over 20,000), the system should be configured to run multiple schedulers in parallel and each scheduler should handle a segment of the total scheduling area. The segments, called optimization areas, should overlap each other. The overlap promotes task/shift migration between schedulers and thus achieves optimal shift routing.

Determining the Best Optimization Area Shape

Every route that crosses an optimization area boundary represents additional workload for the scheduler. The scheduler must acquire all tasks of every shift it schedules, so it is most efficient if every scheduler has most of its shifts wholly within its own optimization area. Thus, the best shape for an optimization area is the one least likely to have routes exiting and entering the area.

Optimization area shapes must be complementary. Optimization areas of different sets should not share common boundaries and should overlap as much as possible to promote optimum routing.

The diagram below illustrates how a pair of complementary optimization areas may be insufficient for optimal routing at specific points on the map. This simple diagram shows two optimization sets, each with two optimization areas. Each of the four optimization areas shown has its own scheduler - for simplicity, only partial sets are illustrated. The black and orange lines represent four different shifts.
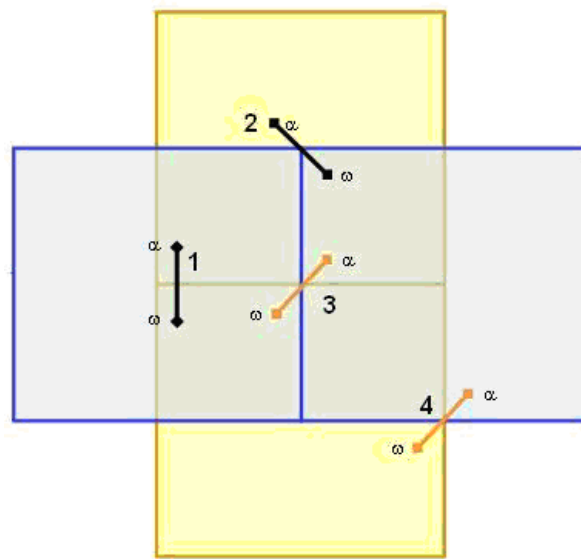
Shifts 1 and 2 can be adequately covered in this configuration, because each one lies entirely within at least one scheduler. Shifts 3 and 4, however, cannot, because no single scheduler contains all the tasks of the shift. This is referred to as the corner problem, which occurs wherever the boundaries of two different optimization areas intersect and the two areas belong to different optimization sets.

Adding a third optimization set, as shown in the following diagram, solves the corner problem. All shifts can be covered as shown, because each one lies entirely within at least one scheduler.
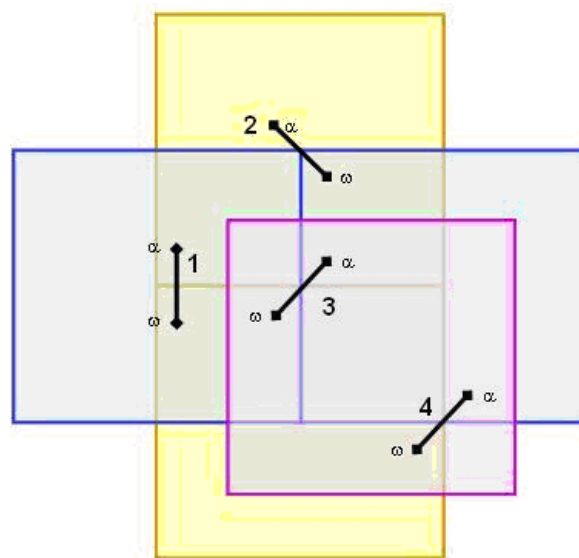
In general, corner problems caused by optimization area overlap must be covered by a third set of optimization areas. If no corner problems exist, only two optimization sets are required.

Follow these general steps when designing optimization sets:

1. Design a set of optimization areas to cover the scheduler area, where each optimization area is as large as possible (the number of tasks bounded by any area should not exceed the maximum scheduling load plus a growth factor) but still fits into a single scheduler. For best results, always try to use the least number of optimization areas possible and assign only one optimization area to a scheduler (though multiple optimization areas can be assigned to a single scheduler).

2. Design a second set of optimization areas where the optimization area boundaries are as far as possible from the first set's optimization area boundaries.

3. Design a third set of optimization areas to cover any 'corner problems' caused by the overlap of the first two sets.

Example:

The following image illustrates a scheduler area, named 'City', that contains more tasks than the physical limit for a single scheduler. Therefore, the scheduler area will require three sets of optimization areas.

City

Define the first optimization set with coordinates: (40.9, -75.1 NW) to (40.2, -74.0 SE). The optimization set will have two optimization areas:

- Optimization area 1: (40.9, -75.1 NW) to (40.6, -74.0 SE).

- Optimization area 2: (40.6, -75.1 NW) to (40.2, -74.0 SE).
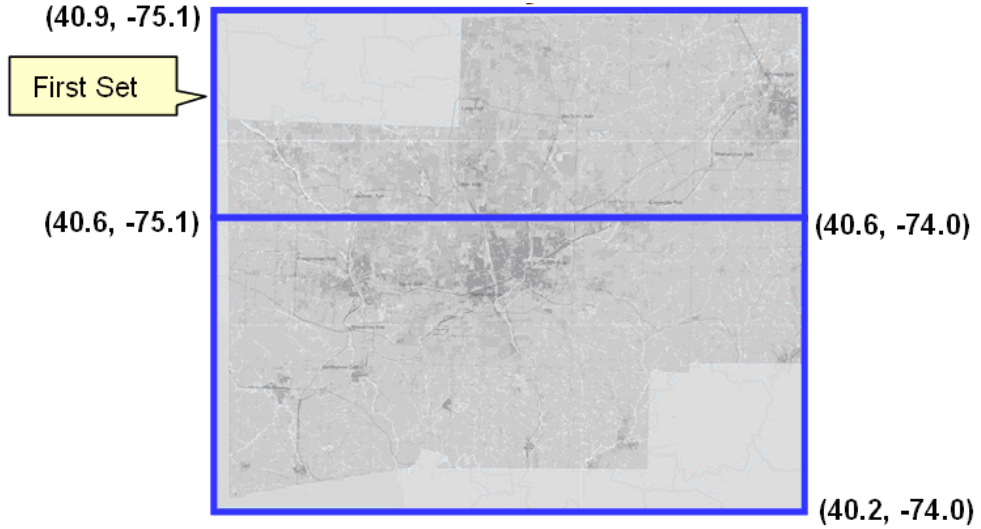
Define the second optimization set with coordinates: (40.9, -75.1 NW) to (40.2, -74.0 SE). The optimization set will have two optimization areas:

- Optimization area 1: (40.9, -75.1 NW) to (40.2, -74.5 SE).

- Optimization area 2: (40.9, -74.5 NW) to (40.2, -74.0 SE).
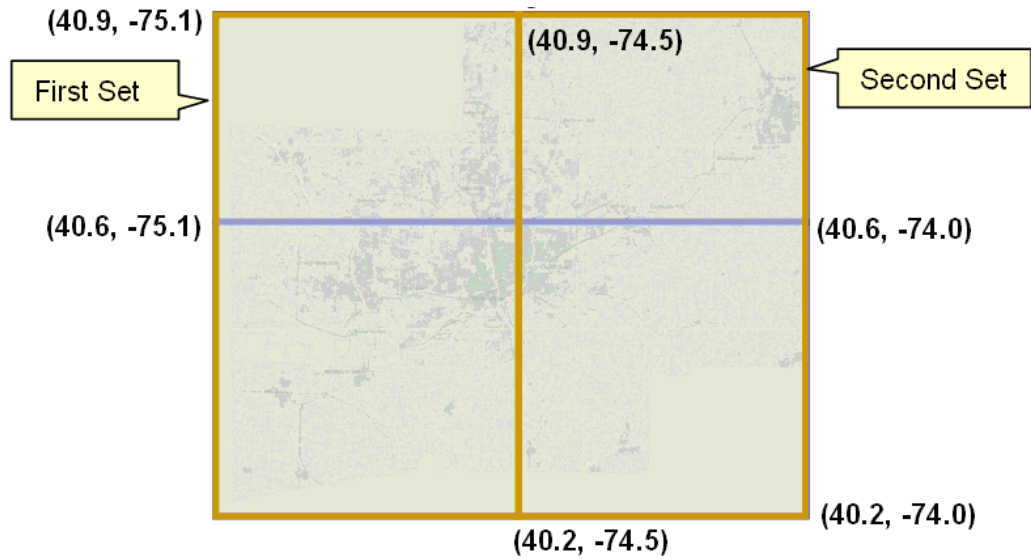
Define the third optimization set with coordinates: (40.7, -74.7 NW) to (40.5, -74.4 SE). The optimization set will have one optimization area:

• Optimization area 1: (40.7, -74.7 NW) to (40.5, -74.4 SE).
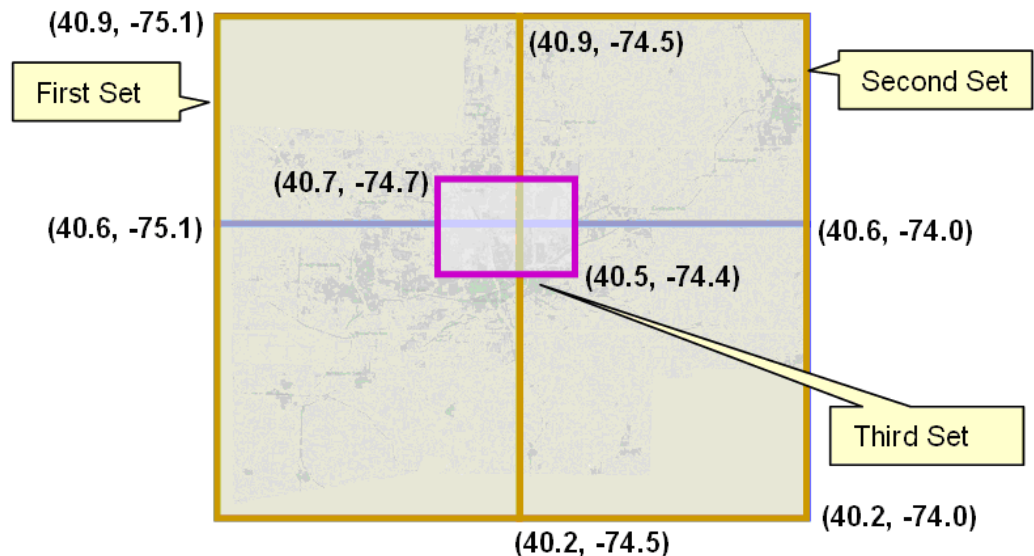
Scheduler Parameters

Scheduler parameters control how a particular scheduler will schedule tasks, optimize routes, and dispatch activities to crews.

Parameters are typically defined as part of the implementation process. The Parameter Definition portal allows admin users to set the minimum, maximum, and default values for each scheduler parameter.

Most scheduler parameters are associated with a scheduler configuration and their values are maintained on the Scheduler Configuration portal. Entity parameters are associated with other objects in the system, such as shifts or activity types, and are maintained on the maintenance portals for those objects.

Understanding Cost Parameters

The system uses cost control parameters to encourage or discourage particular actions within the system. Applying cost controls allows the system to produce the most cost-effective schedule while adhering to your business rules and processes.

Cost Parameters are of two types:

• **Relative:** Cost factors that work as multipliers against global costs. For example, consider that an organization has full-time crews and contractors that may be called on to handle excess work. In order to encourage the system to schedule work to contractors only when no full-time crew is available, you could apply a relative cost of ten to contractor shifts, indicating that these shifts would cost ten times as much to use in the schedule as a full-time crew shift. Examples of relative cost parameters are Relative Shift Cost, and Relative Late Cost. Relative costs factors default to a value of 1.0.

• **Global:** Costs that are applied globally at a base level within the schedule. Examples of global cost settings are Late Cost, Shift Cost, Travel Time Cost, and Travel Distance Cost.

Global costs can be of various types:

- **Flat:** Costs that are applied once, when the object is activated, and removed when the object is no longer being used. The cost for using this object does not go up over time but stays at a standard or flat cost. For example, the global Shift Cost is the flat cost of utilizing a shift. The Relative Shift Cost parameter can be used as a multiplier against the flat cost.

- **Variable:** Costs that are applied per use or per unit. These are also referred to as running costs. Travel Time Cost and Travel Distance Cost parameters fall into this category.

- **Complex**: Costs that increase exponentially and are nonlinear in nature. The Late Cost parameter, which defines the global cost of late arrival to an activity, is a function of the number of seconds late for arrival. For example, late cost is incurred beginning with the first second the crew is late for arrival. However, the exponential function can be defined so that the late cost is insignificant for the first ten minutes, but starts to ramp up quickly thereafter. This cost pattern models the idea that the customer will accept a crew being 5-10 minutes late, but will be increasingly displeased with any further delays.

The following figure illustrates the type of curve represented by a complex cost:
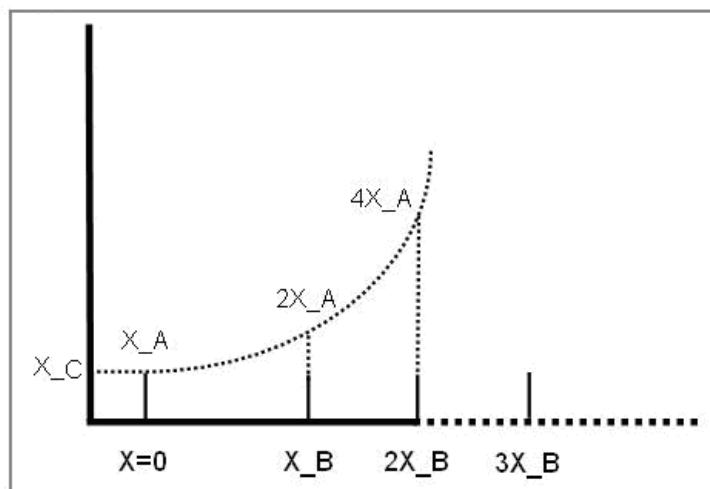
Complex costs are determined by mathematical functions controlled by a group of three cost control parameters:

- Cost Value A represents the initial slope for x>=0 (at x=0).

- Cost Value B represents the doubling rate (the slope of the function doubles every x_B).

- Cost Value C represents a flat additional cost, creating an initial step.

- **Time-Dependent:** Factors that adjust shift-based costs based on when a shift begins. For example, when the system is scheduling activities several days into the future, the goal is to combine activities to produce the best route for each day. As a result, some shifts may appear under-utilized or even empty. However, when scheduling activities for the current day, the main objective is to add as many activities as possible to fully utilize the crew's time. Time-dependent cost (TDC) factors allow the scheduler to produce the most cost-effective schedule considering these time factors.

To configure time-dependent costs, you must define three distinct curves.
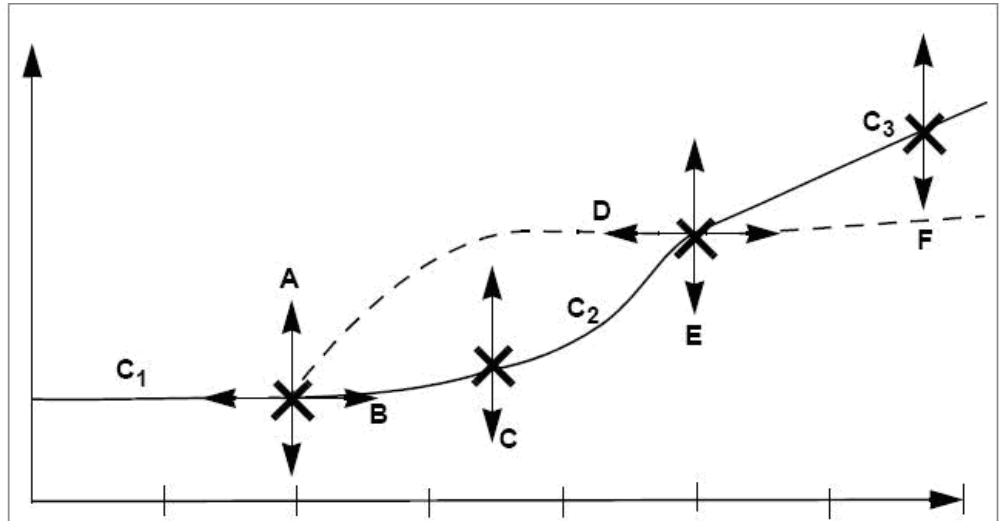
In the diagram above, the horizontal axis is time, in day increments (Day 0, Day 1, Day 2, etc.). The initial curve is always flat. The second curve is either convex or concave. The final curve is always a straight line into infinity.

Six independent variables (TDC Values A-F) are required to define this shape:

- **TDC Value A:** Initial value for the time-dependent cost factor, which is flat from day 0 to TDC Value B. This value must be greater than or equal to zero.

- **TDC Value B:** Controls how long the initial TDC Value A applies, in hours. For example, if the initial factor should apply for all of the first day, then this should be set to 24 (hours).

- **TDC Value C:** Controls the shape of the curve between the initial value and the final slope, referred to as the transition curve. The valid range is 0.1 to 10. A value of 4 produces a curve similar to the solid line in the previous diagram.

  A value of 0.2 produces a curve similar to the dashed line in the previous diagram.

  A value of 1 produces a straight, linear gradient between the initial and final slope, and is a good starting point.

    - **TDC Value D:** Number of hours after which the function changes from curved to linear. This value controls the width of the transition curve. This value must be greater than TDC Value B.

    - **TDC Value E:** Factor at the point at which the function changes from curved to linear. (This the final factor for the transition curve and the initial factor for the final curve.) This value must be greater than or equal to zero.

    - **TDC Value F:** Controls the gradient of the final slope (units/hour).

Thus, the three curves are defined as follows:

- **Initial curve:** C1 (x:0..B) = A

- **Second curve:** C2 (x:B..D) = $((x - B) / (D - B))C * (E - A) + A$

- **Final curve:** C3 (x:D..) = $(x - D) * F + E$

Time-Dependent Cost Examples

Example 1: TDC Applied to Late Cost

Time-dependent costs can be applied to the **Late Cost** parameter to address a situation where cost should remain the same for the first two days, then rise rapidly 1 cost unit per day over the next two days and then increase by 0.2 units for all the remaining days. In this example, TDC variables would be set as follows:

A = 1

B = 48 (hours)

C = 1

D = 48 (hours)

E = 3

F = 0.0083 (0.2 / 24)

Assuming that 5 minutes of late time is costed at 100 units to begin with, applying these time-dependent costs would result in the following:

For a shift on day 1, the cost is 100

For a shift on day 2, the cost is 100

For a shift on day 3, the cost is 200

For a shift on day 4, the cost is 300

For a shift on day 5, the cost is 320

For a shift on day 6, the cost is 340

For a shift on day 7, the cost is 360

Example 2: TDC Applied to Shift Promotion Cost

Time-dependent costs can be applied to the **Shift Promotion Cost** parameter to promote early completion of activities. The scheduler has no innate preference for activities to be completed as early as possible. Until other constraints are breached, the scheduler is mainly distance driven. Thus, it may schedule an activity for day 5, even though there is capacity on day 1, if it results in less travel. Time-dependent costing can be applied to the Shift Promotion Cost to bias against distance in favor of ensuring that no transport capacity is wasted for day 1. Ideally, costing is set so that day 1 fills up just before it becomes day 0. If costing is too weak, capacity is wasted; if too strong, days fill up well in advance, and the quality of routes suffers.

Example 1: TDC Applied to Reserve Capacity Cost

Time-dependent costs can also be applied to the **Reserve Capacity Cost** parameter to limit the area covered for individual routes when providing appointments. This is done by invoking an additional cost when a vehicle exceeds a certain percentage (such as 50%) of its available time. Thus, the first appointment determines the general direction of that day's run, with the distance travelled for subsequent appointments being restricted through the reserve capacity cost. The more distance is being covered, the stronger the restriction, causing activities that introduce excessive travel to be deferred to another day. Time-dependent costing is required to gradually relax the Reserve Capacity Cost, enabling the system to fill day 0, day l and so on, in case the reserve capacity cost turns out to be too restrictive.

Understanding Auto Dispatch

Dispatching is an action that locks a task's sequence in the shift schedule so no further optimization takes place and marks it as ready to be sent to the crew. Dispatching can be either manual (initiated by a dispatcher) or automatic (performed by the system without user initiation).

• Automatic dispatching applies only to a started shift. Until the crew starts their shift, the system does not automatically mark any task as ready to be dispatched.

• Manual dispatching applies to activities only. Breaks and POU tasks are always automatically dispatched.

• A dispatcher may manually request to dispatch an activity at any time. If a crew's shift has not started yet, then the activity remains queued for dispatch until the crew starts the shift.

You can enable or disable automatic dispatching of activities globally or for specific crew shifts or activity types.

Setting Auto Dispatch Globally

To enable or disable auto dispatching globally, set the *Enable Auto Dispatch* scheduler parameter to True or False in the scheduler configuration. (This is listed under Real Time parameters on the Scheduler Configuration portal.) If set to True, auto dispatching of activities is enabled globally, but can be disabled at the activity or shift level. If False, auto dispatching is disabled and cannot be enabled at the shift or activity level.

The scheduler configuration defines additional scheduler parameters affecting auto dispatching.

• *Break Dispatch Mode*

• *POU Dispatch Mode*

• *Non Productive Task Dispatch Mode*

• *Auto Dispatch Stability Period*

• *Auto Dispatch Interval*

• *Auto Dispatch On Completion*

• *Auto Dispatch Time Horizon*

• *Emergency Dispatch Mode*

Setting Auto Dispatch for Crew Shifts

To enable or disable auto dispatching at the crew shift level, set the following attributes for a particular crew shift or crew shift template:

• **Drip Mode:** Indicates how the system should handle dispatching of activities to the crew shift. Valid options are:

None: Automatic dispatching is disabled. All scheduled activities must be manually dispatched to the crew by the dispatcher or by another system-invoked process.

All: All scheduled activities are automatically dispatched to the crew and any additional activities scheduled throughout the shift are also auto-dispatched. This excludes any activities where the Auto Dispatch attribute is set to No on their Activity Type.

Standard: Only a fixed number of scheduled activities are automatically dispatched to the crew at any given point of time. The fixed number is specified in the Drip Horizon. Note that if an activity's type does not allow automatic dispatching, the activity will not be assigned to shifts that are set to Standard drip mode.

• **Drip Horizon:** Number of activities that can be dispatched to a crew at any time. Valid only if Drip Mode is set to Standard.

Setting Auto Dispatch for Activity Types

To enable or disable auto dispatching for a particular activity type, set the **Auto-Dispatch** attribute at the activity type level. If set to Yes, activities of this type will be automatically dispatched to the assigned crew. If set to No, activities of this type will not be automatically dispatched, overriding the global or shift-based auto-dispatch settings.

> **Note:** Preventing certain types of activities from being automatically dispatched is applicable only if auto-dispatch is enabled globally and the Drip Mode of the crew shift is set to All. Manual dispatching is not allowed if a shift is being drip fed tasks (Drip Mode of Standard).

Scheduler Configurations

A scheduler configuration is a predefined set of parameter values that can be associated with one or more schedulers.

Scheduling parameters include cost controls, optimization settings, and real-time processing parameters. A scheduling configuration called scheduling horizon also specifies the number of days in the future for which data should be considered for scheduling.

All schedulers whose scheduler area is associated with the same service area must use the same scheduler configuration. Only schedulers that do not cooperate (that is, they have different service areas) may have different scheduler configurations.

Use the Scheduler Configuration portal to add or change a scheduler configuration. Refer to the online help for instructions on creating a new configuration, completing it, and associating it to specific schedulers.

> **Note**: Use this portal to set the values of scheduler parameters for a particular scheduler configuration. The parameters and their default values are defined on the Parameter Definition portal.

Click the **Expand** icon to see the parameters of a particular group. Detailed descriptions are available in the embedded help. See *Scheduler Parameters* for more information.

Use the **Complete** action to indicate that the scheduler configuration is complete. Only complete configurations can be associated with online schedulers. When a scheduler configuration is in the Complete state, you cannot change any of its parameter values. Once complete, you can assign the scheduler configuration to one or more schedulers.

> **Note**: To change a completed scheduler configuration, you must set the status back to Pending first. If you need to change a scheduler configuration that is being used by one or more online schedulers, you will need to duplicate it first, then edit the copy and change its status to Complete. You can publish the configuration as described below.

Use the **Publish** action to replace an existing configuration with an updated one. This allows you to keep the previous configuration intact. You will need to take the schedulers offline before you can replace their configuration. Refer to the online help for complete instructions on publishing a scheduler configuration.

Use the **Boot** action to bring a scheduler online.

Use the **Shut Down** action to take a scheduler offline.

Setting Up Your Schedulers

This section provides a high-level summary of the steps required to set up schedulers and scheduling entities for your organization.

1. Review the scheduler-related plug-ins provided with the base system and determine if any customizations are needed. Refer to Appendix C for more information about plug-ins.

2. Use the MapEditor to build map files used by the scheduler to create and optimize routes, and calculate travel times and distances. Refer to the MapEditor User's Guide for details.

3. Review the scheduler parameter definitions (see Appendix A) and adjust the global default values if necessary.

4. Analyze the geographic areas requiring scheduling. Use the guidelines presented earlier in this section to determine number of schedulers required. If optimization areas will be used, determine optimal size and shape of each optimization area. Scheduler areas and optimization areas are defined on the Scheduler Area portal.

5. Determine how many scheduler configurations you must define. All schedulers whose scheduler area is associated with the same service area must use the same scheduler configuration. Only schedulers that do not cooperate (that is, they have different service areas) may have different scheduler configurations.

6. For each scheduler configuration, determine the scheduler parameter values to use and the number of days in the future for which data should be considered for scheduling.

7. Identify each scheduler you will be using. Associate a scheduler configuration and a scheduling area to each scheduler.

Using the MapEditor

The MapEditor allows you to create map files used by the scheduler to obtain travel information and to display places and travel paths in a particular geographical region. A scheduler map consists of geographic locations, referred to as nodes, and inter-node links for which the direction, distance, and travel time are known.

Documentation for this tool is provided in a separate guide. Refer to the Oracle MapEditor User Guide.

Scheduler Registry

The scheduler registry contains a record for each task, shift, appointment booking group, and scheduler configuration that is linked to a scheduler. All objects associated with the scheduling process register themselves to the scheduler registry.

- An activity is registered with all schedulers that host its service area or, if optimization areas are enabled, its optimization areas. If a scheduler's scheduler area is defined using optimization areas, then the activity registers itself to that scheduler only if the scheduler covers the activity location's geocode. If a scheduler is defined using service areas only, then the activity registers itself to the scheduler if it covers its service area.

- A break is registered with all schedulers that host its shift.

- A POU task is registered with all schedulers that have the common services areas with the POU.

- A shift is registered with all schedulers that host any of its service areas.

- Every appointment booking group is registered with all schedulers.

- A scheduler configuration is registered with all schedulers that use that configuration.

The online scheduler monitoring processes update the scheduler registry automatically to remove objects that no longer belong to any of the scheduler's optimization areas (if used) or service areas (if no optimization areas are used) and to add objects that belong to one of the scheduler's optimization areas but are not already linked to the scheduler. This can happen if a scheduler's optimization areas are altered while the scheduler is running.

Posting Actions

The system uses a posting action to communicate information to the scheduler about a change that was made to an entity. The scheduler then communicates this information to its scheduler process.

> **Note**: It is the responsibility of the audit plug-in of the entity's business object to update the registry when changes are made to that entity.

Posting actions in the registry include:

- **Send**: An entity has changed and the entire entity needs to be sent to the scheduler.

- **Send Status**: An entity's state has changed and state-related details for the entity, rather than the entire entity, need to be sent to the scheduler.

- **Send Schedule**: Another scheduler manager has made a schedule change to an entity and schedule-related details need to be sent to the scheduler.

- **Processed**: The registry record has been processed by the scheduler. For example, an object that needed to be sent to the scheduler process was delivered.

- **Remove**: The system has deleted or cancelled an entity (or changed it so that it is no longer relevant to the scheduler) and the entity needs to be removed from the scheduling process (that is, it should no longer be considered for scheduling). After removing the object, the scheduler deletes the registry entry.

If any errors or warnings resulted from the posting action, they appear in the registry record. Depending on your configuration, To Do entries may also be generated for scheduler registry errors.

Scheduling-Related Algorithms

Scheduler read algorithms are used to read registered objects and package them for transfer between the scheduler manager and the scheduling process. Read algorithms are enabled for tasks, shifts, appointment booking groups, and scheduler configurations.

The base package provides algorithms for the following scheduler-related system events:

- **Task Scheduler Read**: Responsible for setting up the data required to communicate a new or changed task to the scheduler.

- **Shift Scheduler Read**: Responsible for setting up the data required to communicate a new or changed crew shift to the scheduler.

- **ABG Scheduler Read**: Responsible for setting up the data required to communicate a new or changed appointment booking group to the scheduler.

- **Depot Scheduler Read:** Sets up data required to communicate depot information, such as locations and time windows to the scheduler.

- **Scheduler Configuration Read**: Responsible for setting up the data required to communicate a new or changed Scheduler Configuration to the scheduler.

- **Task Post Dispatch**: Responsible for executing the necessary business rules and processing associated with a schedule update of a dispatched task.

- **Update Time Windows**: Called when the system is processing an appointment booking request.

Refer to *Appendix C*: *Algorithm Entities* for more information.

Scheduler Monitor Process

The Scheduler Monitor background process invokes monitoring rules associated with the current state of schedulers. A holding scheduler needs this background process to run frequently, e.g., hourly. Review the settings for this batch process and adjust its scheduling as desired.

Chapter 6

CDI Configuration

This section describes topics related to configuration of the Common Dispatching Interface (CDI) portal and related functions.

- *What is the CDI?*

- *CDI is for Dispatchers Only*

- *The Scheduling Gantt*

- *Frequent Data Refresh*

- *Dispatching Functions*

What is the CDI?

The Common Dispatching Interface (CDI) portal is the primary point of user interaction for management of field resources and tasks. It provides a visual representation of scheduled shifts and their assigned activities, and provides access to all dispatcher functions. The CDI portal includes the following zones:

- KPI Summary

- Scheduling Gantt, from which users can launch the CDI Map

- Activity Search

CDI is for Dispatchers Only

Only dispatchers can access the CDI portal. When a dispatcher attempts to access the CDI portal, the system automatically displays the Logon screen if no active dispatcher shift already exists. Dispatcher shifts are not created until the dispatcher logs on to start a shift.

To ensure that the CDI is working properly and users can access its full functionality, please verify the following:

- The user has been set up as a dispatcher in the system.

- Dispatch areas have been defined, and the dispatcher record identifies the dispatch areas and KPIs the dispatcher is responsible for monitoring.

- The Alert Queue and the Backup Alert Queue zones are properly set up, as described in the Alert Configuration section.

  Note: Before a dispatcher can view the CDI portal, he or she must log on to start a dispatcher shift. When a dispatcher attempts to access the CDI portal, the system automatically displays the Logon screen if no active shift already exists.

The Scheduling Gantt

The system uses a Gantt zone to display a graphical representation of the crew shift schedules being monitored by the logged-on dispatcher.

The CDI also supports a geographical map view of the dispatcher's monitored area. The map is launched from a toolbar button on the CDI Scheduling Gantt zone.

Refer to *Chapter 7*: *Specialty User Interface Configuration* for more information on how these are configured.

Frequent Data Refresh

Key operational zones should be configured to automatically refresh their data to reflect latest updates.

Ensure the automatic refresh interval is properly set up for the appropriate zones. At a minimum, Oracle recommends setting the CDI-related KPI Summary and Gantt zones and the Alert Queue dashboard zone to automatically refresh frequently. The refresh interval for these zones is defined on their maintenance portals. Individual users can override these settings in the zone refresh parameter in user preferences.

The automatic refresh interval for the CDI Map is defined as a multiple of the refresh interval of the Gantt. For example, if the Gantt's automatic refresh interval is 60 seconds and the multiple is defined as 5, the map will be refreshed every 5 minutes. That multiple is defined as a parameter of the Gantt zone.

Dispatching Functions

Dispatchers resolve day-to-day exceptions by performing common dispatching functions. These functions can be performed from the Gantt or Activity Search zone on the CDI portal or from the activity maintenance portal.

These functions are implemented in dedicated algorithms associated with the activity business object. Review the algorithms that control dispatching functions to ensure that they will meet your organization's needs:

• *Allocate to Shift*

• *Cancel Activity*

• *Defer Activity*

• **Lock to Run** (if depot functionality is used)

• *Unassign*

See *Appendix C*: *Algorithm Entities* for more information about these algorithms.

Chapter 7

Specialty User Interface Configuration

This section describes special types of user interface configurations.

- *Gantt Zones*

- *Gantt Colors*

- *CDI Map*

- *Calendar Zones*

Gantt Zones

The system uses Gantt zones to display a graphical representation of crew shift schedules over time. This type of zone is used on the CDI portal, where it displays a graphical representation of the crew shift schedules being monitored by the logged-on dispatcher. This zone type is also used on the Crew Shift Schedule tab portal, where it displays a graphical representation of a particular crew shift's schedule.

When a user right-clicks on a shift or task displayed in the Gantt zone, the system displays detailed information about the object at the top of the context menu. The logic used to display detailed information about a shift or task resides in the following plug-ins:

- Task Detailed Information

- Crew Shift Detailed Information

The logic used to derive the contents for a Gantt zone resides in the *CDI Gantt Data* plug-in. Refer to *Appendix C: Algorithm Entities* for more information about these plug-ins.

The functionality provided by this zone requires that you set up a feature configuration of type "Application Context Root," which defines the main application's root location. See "Feature Configuration" on page 2 for more information.

Context Menus

You can add custom options to the context menu that appears when a user right-clicks an object in the Scheduling Gantt zone. The system supports several customizable context menus:

Context Menu	Displays when a user right-clicks...	From the...
M1-GANTT-ACTIVITY	A single activity that is not completed	Scheduling Gantt in the CDI

Context Menu	Displays when a user right-clicks...	From the...
M1-GANTT-ACTIVITY-MULTI	Multiple activities that are not completed	Scheduling Gantt in the CDI
M1-GANTT-BREAK	A break task	Scheduling Gantt in the CDI
M1-GANTT-DEPOT-TASK	A depot task	Scheduling Gantt in the CDI
M1-GANTT-DEPOT-TASK-MULTI	Multiple depot tasks	Scheduling Gantt in the CDI
M1-GANTT-POU	A POU	Scheduling Gantt in the CDI
M1-GANTT-SHIFT	A shift	Scheduling Gantt in the CDI
M1-GANTT-SHIFT-SHIFT_PORTAL	A shift	Scheduling Gantt in the Crew Shift Schedule tab.
M1-GANTT-TASK	Any other single task	Scheduling Gantt in the CDI

To add a new item, select Menu from the Admin menu and then find the appropriate content menu from the list above. Refer to the Framework documentation for instructions on adding menu items to the menu. Menu lines added to these context menus must invoke a BPA script to perform the work, and the BPA script must follow a specific schema. Each context menu provided with the base package contains at least one item that serves as an example of how to implement such a menu item.

The items you add will appear on the on the appropriate entity's context menu in addition to the standard menu times.

Gantt Colors

Customize the colors used to represent shift and task states by copying the base record from managed content to your own record.

1. Navigate to Admin -> Managed Content

2. Search for and select the M1-GanttStyles record

3. Copy to your own record.

   You must create a custom managed content record that is a copy of M1-GanttStyles then modify the copy.

4. Adjust the colors as needed using html hex codes.

5. Override the URL parameter on the Gantt zone to reference their custom managed context record instead of the base.

CDI Map

The Common Dispatcher Interface (CDI) supports a map view of the dispatcher's monitored area. The CDI map is launched from a toolbar button on the CDI Scheduling Gantt zone.

The CDI Map functionality requires that you set up a feature configuration of type "Map Server Configuration." See "Feature Configuration" on page 2 for more information.

Although no specific tools are provided to automatically configure the CDI Map, implementers can add or remove themes. (Themes are the types of objects displayed on the map, such as crews,

routes, and activities.) To add or remove themes from the CDI map, first duplicate the existing UI map for the CDI Map, and edit the java script to introduce your own theme using Amplifier's rules/scripts.

When a user hovers on a point on the map that is related to a shift or task, the system displays detailed information about the object in a context window. The logic used to display detailed information about a related shift or task resides in the following plug-ins:

- Task Detailed Information

- Crew Shift Detailed Information

Refer to Appendix C for more information about these plug-ins.

The CDI Map UI map is defined in the zone parameter of the Gantt zone.

> **Note**: Once you implement a custom UI map, any updates to standard UI maps will not be applied to the cloned maps.

Calendar Zones

The system uses Calendar zones to display events or items occupying time in a calendar format. For example, the Calendar zone on the Crew portal displays a crew's calendar of shifts and meetings.

Logic to derive the contents for a Calendar zone resides in a plug-in. Refer to Appendix C for more information about this plug-in.

The functionality provided by this zone requires that you set up a feature configuration of type "Application Context Root," which defines the main application's root location. See "Feature Configuration" on page 2 for more information.

Chapter 8

Mail Management

This section describes how to configure mail management functionality.

- *Mail Management Basics*
- *Mail Management User Interface*
- *Mailing Lists*
- *Mailing Groups*
- *Mail Templates*

Mail Management Basics

Mailing functionality is configured using a number of business objects. These dictate how mail is sent, how it is received, and who receives which messages.

When a mail message is created its business object determines the list of recipients it should be sent to and creates a recipient mail message for each recipient. The business object on the recipient mail record is responsible for delivering the message also to the recipient mobile device when applicable. Review the base package business objects to better understand how mail is processed and to customize these rules as needed.

At any time a user can view mail they sent or received via the My Mail portal.

Mail Management User Interface

Users are notified of new messages and have access to opening mail via the My Mail portal and the Mail Summary dashboard zone.

My Mail Portal

The My Mail portal provides access to view mail you have received or sent. Any user with a valid login can use this portal to see their messages. This is also useful for non-MDT or SMS crew members who do not have access to mail from a portable device. You can create new mail, reply, acknowledge mail you received and archive your mail using this portal.

Mail Summary Dashboard Zone

You can also manage your mail using the Mail Summary Dashboard zone. Make sure that users who should have access have this zone added in their preferences.

Mailing Lists

Mailing lists define a list of users for mail distribution purposes. When mail is sent to the list, each user receives a copy. Set the status of mailing lists to active or inactive to affect their availability to users. Mail cannot be sent to inactive lists.

Mailing Groups

Mailing Groups differ from Mailing Lists in that a group is dynamic. For example, a mailing group changes based on the logged on or logged off status of members of the group. The logic to interpret the business meaning of each group code resides in an algorithm associated with the mail business object. Use this business object to customize your group codes.

Mail Templates

Mail message templates establish standard settings for the fast creation of mail messages. Users can then select the template when creating new mail and the pre-defined subject, message and other settings are automatically copied to the new message from the template.

You should configure appropriate mail templates to handle standard situations which may arise in your business process.

Chapter 9

Transfer of Goods

This section describes how to configure transfer of goods functionality.

- *Transfer of Goods Basics*
- *Depots*
- *Configure your Transfer of Goods Custom Entities*

Transfer of Goods Basics

Transfer of goods involves the process of transporting goods from one location to another. Not all businesses use this business practice, however, if they do, the system provides a way to manage goods, restrictions and capacities related to the transfer of goods.

Transfer of goods functionality involves the following entities:

- Activities - Describe transported goods and any restrictions on what cannot be shipped with the goods on the activity type.
- Vehicles - Describe any capacity limits for goods on the vehicle type.
- Shifts - Provides the opportunity for dispatchers to override any restrictions or limits on a specific shift.

If goods are transferred to and from depots, an activity can specify a list of valid depots where the goods can be picked up or dropped off. Crews can also be associated with a specific list of depots that they are are allowed to visit. A dispatcher can also override this list for a specific shift. Refer to the depot section for more information.

Depots

If transferred goods are loaded or unloaded at a particular centralized location, your organization can set up depots to capture the information related to these locations. Depots specify details such as whether the depot is a pick up or drop off location, what types of goods the depot can handle, the physical address and the days and times that the deopt is open.

Two types of depot operations are supported:

- Distribution - a depot where crews pick up goods for delivery.
- Collection - a depot where crews drop off goods picked up from customers.

    Note: If a Depot location can function as both a distribution and a collection depot, one depot of each type should be created to refer to the same location.

Depot Tasks

Depot task is a tool for tracking the crew's travel, arrival and completion of the depot visit. The system generates depot tasks as needed when planning depot runs to and from depot centers to accommodate depot activities. A depot task is dispatched to a crew, just like an activity, and its status is updated and tracked in the same way.

The scheduler relies upon the master configuration for the type to use when creating depot tasks.

Depots and Depot Profiles Control Everything

Depot profiles can be set up to establish common settings for site delay, access time windows and service time windows across multiple depots. All other details are defined on each individual depot. If a depot refers to a profile, the settings established by the profile cannot be overwritten.

Depot Breaks

The system can also be configured to designate breaks which can only be taken at a depot. If a shift is not scheduled for depot work, the break is scheduled as a standard break.

Configure your Transfer of Goods Custom Entities

The following guidelines provide a basis for your transfer of goods configuration. For more detail and examples, please refer to the M1 demo environment delivered with this release. In general, to use transfer of goods functionality, you must complete the following:

Create a Custom Depot Activity Type BO

Create a custom business object to handle your depot activity types. The business object follow these parameters:

- Subclass the base Activity Type BO to create your custom Depot Activity BO. Make sure you include the base Activity Type BO in the CM BO schema

- Add mapping to the Depot Activity Class field.

- Create custom UI maps for the BO to expose this information.

- If all your activity types should be based on this BO change the base Activity Type BO to be not allowed to be instantiated.

Create a Custom Activity BO

Create a custom business object to handle your depot activities. The business object follow these parameters:

- Subclass the base Activity BO to create your custom Activity BO for handling transfer of goods. Make sure you include the base activity BO in the CM BO schema

- Include the "Common Activity Transfer of Goods Details" data area in your CM BO

- Create custom UI maps for the BO that expose goods related information

- Reference the custom activity BO on corresponding activity types

Create a Custom Vehicle Type BO

If vehicles are limited by maximum capacity you must also create a custom vehicle type business object. The business object should follow these parameters:

- Subclass the base vehicle type BO to create your custom vehicle type BO for handling transfer of goods. Make sure you include the base vehicle type BO in the CM vehicle type BO schema

- Include the "Common Maximum Capacities" data area in your CM BO

- Create custom UI maps for the BO that expose capacity information

- If all your vehicle types should be based on this BO change the base Vehicle Type BO to be not allowed to be instantiated.

Create a Crew Shift BO

- Subclass the base crew shift BO to create your custom crew shift BO for handling transfer of goods. Make sure you include the base crew shift BO in the CM vehicle type BO schema

- Include the "Common Shift Transfer of Goods Details" data area in your CM BO

- Create custom UI maps for the BO that expose this information

- If all your crew shifts should be based on this BO change the base Crew Shift BO to be not allowed to be instantiated. Reference your custom crew shift BO on your shift type records.

Create a Custom Crew BO

Create a custom business object to handle crews that will have depot related shifts. The business object should follow these parameters:

- Subclass the base single person crew BO to create your custom depot based crew BO. Make sure you include the base crew BO in the CM BO schema

- Include the "Common Crew Transfer of Goods" data area in your CM BO

- Create custom UI maps for the BO that expose this information

- If all your crews should be based on this BO change the base crew BOs to be not allowed to be instantiated. Reference your custom crew BO on your crew type records.

Create a Custom Break Type BO

If breaks are required to be taken at a depot, you must create a custom business object to handle these depot break types. The business object should follow these parameters:

- Subclass the base Break Type BO to create your custom depot based break type BO. Make sure you include the base BO in the CM BO schema

- Add mapping to the Location Option field, mark it required and provide a default value of "M1DP" which is "at depot".

- If all your break types should be based on this BO change the base break type BO to be not allowed to be instantiated.

Define Capacities

Capacities should follow these parameters:

- If you are using cumulative capacities to describe goods, such as weight, volume etc, override the descriptions of up to 5 capacity types defined in the lookup M1_CUMULATIVE_CAPACITY_FLG to enable them.

- If you are using non cumulative capacities to describe goods, such as size, override the descriptions of up to 5 capacity types defined in the lookup M1_NONCUM_CAPACITY_FLG to enable them.

- If you are using maximum cumulative capacities to describe maximum load, such as maximum weight, maximum volume etc, override the descriptions of up to 5 maximum capacity types defined in the lookup M1_MAX_CAPACITY_FLG to enable them.

- If you are using maximum non cumulative capacities to describe maximum load, such as maximum size, override the descriptions of up to 5 maximum capacity types defined in the lookup M1_MAX_NONCUM_CAPACITY_FLG to enable them.

Define Depot Task

Define the depot task, then reference it on the master configuration.

Scheduler Configuration for Depots

- Set up corresponding scheduler configuration costs for your applicable cumulative and non-cumulative capacity types. For example, if you defined "cumulative capacity 1" as "weight" then make sure you provide the cost for weight in the "cumulative capacity cost 1" parameter.

- Set up depot related parameters as needed.

Chapter 10

Mobile Communications Platform (MCP) Configuration

This section provides information relating to the Mobile Communications Platform (MCP) and its configuration.

- *Configuration Tools*
- *Navigation and Display*
- *Alert Types*
- *Remote Script Invocation*
- *Mobile Device Registration*
- *How Unprocessed Data is Handled*
- *Mobile Device Log Files*

Configuration Tools

Standard framework configuration tools, such as business objects, scripts, UI maps, algorithms, etc., are used to create, extend and customize the mobile application. etc.

It is important to note that the mobile framework is a slightly scaled-down version of the Oracle Utilities Application Framework, and there are a few differences worth noting.

Business Objects

Entity maintenance is only performed via interaction with business objects. The business object layer does not interact with an underlying maintenance object layer, but rather is based on a completely different data model architecture. This means that all business rules to be performed on the mobile application need to be defined as business object rules.

The business object framework is limited as follows:

- Automatic state transition is not supported.
- Transitory states are not supported.
- Only two plug-in spots are supported.
    - *MCP Enter*
    - *MCP Post Processing*

    Refer to *Appendix C: Algorithm Entities* for more information about these plug-ins.

- BO invocation for read, add, replace, and delete actions is supported.

The same business object definition that governs an entity on the server application also governs it on the mobile application; however, different business rules apply. On the mobile, entity-specific business logic must be triggered by these MCP-specific business object plug-in spots only.

Scripting is the only supported programming language for these plug-in spots and java scripting for user interface rules. Service scripts and plug-in scripts are limited to XPATH version 1.0.

As mentioned earlier, the data model architecture is completely different from the server framework. Instead of individual maintenance objects, the mobile framework is based on a single XML data storage for business object instances. This also means direct access to tables for query purposes is not possible. All queries need to be performed by calling the MCP Query mobile framework business service. Refer to that business service for more information.

Custom Business Services

Custom business services can be created to allow you to create more complex or performance sensitive logic. For example, these business services can be enabled to allow for field billing, scanning, printing, other digital captures and so on.

Details on environment setup, implementation requirements, and deployment can be found in the in the *Mobile Communications Platform (MCP) Configuration* Appendix.

User Interface

There is no notion of portals, zones or BPA scripts on the mobile device. Service scripting syntax in the mobile framework supports a "Terminate with Map" command that allows user interface interaction. This is the only means of presenting data to and obtaining data from a user.

Different devices may call for a different UI map to maintain and display an entity's details. For example, a handheld device has less real estate than a laptop. To accommodate differences in screen size, the MDT type record provides a display option attribute, which specifies the type of display (such as Windows Mobile VGA or Windows XVGA). Further, each business object that is maintained by the mobile application, like task and shift business objects, has an option where you can specify the mobile script to use for each supported device display option.

MCP Specific UI Map Attributes

This section describes additional HTML attributes supported by the MCP framework only.

Navigating Back to Previous Map

When navigating to a UI Map with this tag, this tag tells MCP to push the previous UI Map name and data onto the stack of backable maps in case the users executes a "back" at some point (MPC would pop the UI Map name and data from the stack and navigate to it). If this tag it not on the UI Map being navigated to, then the backable map stack is cleared.

For example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">

<html >

    <body oraMCPSupportsBack="true">

    …

    …
```

```
                    </body>

            </html>
```

Clickable Phone Numbers

You can use the oraPhone and oraPhoneContact attributes to identify phone numbers on UI Maps.

For example:

```
<td oraPhone="true" oraPhoneContact="boGroup/customerInformation/
contactName" oraField="boGroup/customerInformation/mainPhone"></td>
```

Device vs. Server Time Zones

The mobile framework supports the user working in the local time zone for their device while at the same time managing the conversion to and from the base time zone when communicating data to and from the server. Please refer to the Base Time Zones section for more information.

Geographic Map

If the device is configured to support mapping, a geographic map icon appears on the task list user interface as well as on task related maps. You can configure special theme layers to be displayed on the maps which users can access for tasks on their devices. For example, you might configure layers to show the location of company assets. Different types of crews can be associated with different applicable layers.

- Refer to customize device capabilities to enable mapping for a specific MDT Type.

- Refer to the Map Layers plug-in spot to configure custom layers to appear on the map.

Navigation and Display

This section describes configurable aspects of the mobile application work flow and display.

Everything Starts with the Initial Script

When a user logs on to the mobile application, the mobile framework invokes the Initial Script defined for the specific deployment on the device. (Refer to the deployment section for more information on deployments.) The base package Initial Script demonstrates how a crew obtains their shift for the day, starts the shift, and then is presented with their tasks for the day. The crew can work each task and use other actions to manage their shift.

As mentioned earlier, the maintenance user interface associated with each task and shift business object can easily be replaced with your custom user interface, as these are defined as business object options. If you need to customize common parts of the mobile application that are not business object specific, you can do that by cloning the Initial Script and adjusting it as needed.

Indicator Bar

The user interface on mobile devices includes an indicator bar which shows various indicators such as new email and connectivity. You can introduce your own indications by doing the following::

- Add the special icons to the "Mobile Device Indicator Bar" extendable lookup.

- Add custom logic to the mobile application to properly add or remove the indication from the bar as needed.

    To this you will need to call the MCP Indicator Bar Manager business service.

The Tool Bar

The mobile application displays a toolbar below the application indicator bar. The toolbar is available from every application screen, but the buttons that are displayed at any time vary depending on the current application screen you are viewing.

General Actions Menu

The Task List user interface supports an Actions icon that when clicked presents the user with a list of actions a crew can take during their shift, for example changing their primary function, changing crew allocation, going offline, completing their shift etc.

To add more custom actions

1. Copy the base actions script and UI map to create your custom script and map.

2. Enhance them to support your custom actions.

3. Copy the base initial script and set it to call your custom actions script.

4. Use your custom initial script in your deployments..

Task Actions Menu

When on a task related user interface a menu actions icon appears if configured on the task business object. The base activity business objects are configured with a BO option to manage activity related actions such as creating a new activity for the same location, capturing contents for this activity etc.

To add more task related custom actions

1. Copy the base actions script and UI map to create your custom script and map

2. Specify your actions script as a BO option with a higher sequence number than the base actions script.

Geographic Map

If the device is configured to support mapping, a geographic map icon appears on the task list user interface as well as on task related maps. You can configure special theme layers to be displayed on the maps which users can access for tasks on their devices. For example, you might configure layers to show the location of company assets. Different types of crews can be associated with different applicable layers.

- Refer to customize device capabilities to enable mapping for a specific MDT Type.

- Refer to the Map Layers plug-in spot to configure custom layers to appear on the map.

Download Attachments

The base product provides a service for downloading attachments to the MCP. The service call is initiated from the mobile application and requires the crew to be connected. Customization is required to use this functionality. When implemented, the crew can open the attachments in their native viewer. Examples of attachments include schematics, a list of materials, or specific repair instructions in a Microsoft Word or Adobe PDF document.

To use this functionality complete the following:

1. Define an attachment business object to match its file type. Refer to "CM-ActivityDocumentationPDF" business object in the demo environment for an example.

2. Load your attachments to the database into the Attachment maintenance object.   The attachment record must be based64 encoded and stored using your attachment business object.

Note: there is no user interface currently to maintain and display these attachments on the server.   Once you have your attachment encoded you can use a BPA or an XAI service to add the attachment business object. Refer to "CM-AddPDFBPA" BPA script in the demo environment for an example.

3.   Include the attachment ID in your a custom UI map deployed to the device. Implement a java script function so that when a link or button is clicked to down load the attachment it constructs the URL to download your attachment ID. Refer to "CM-MCPInsMtrActActionsMenu" UI Map in the demo environment for an example. In this example the link to the attachment is on the activity actions menu UI map but you can choose to place on any UI map as needed.

Alert Types

Specific alert types must be configured to support MCP functionality.

Panic Alert

Crews can select the panic button on their mobile device to alert the dispatcher that they are in distress. If the mobile device has the capability, the alert is sent with the GPS location of the crew at the time that the panic alert was issued to better locate the crew on the map.

This functionality requires:

- The panic alert type configured according to the following:
  - No creation algorithm
  - Allow manual close
  - Allow snooze and recycle
  - Use the "snooze forever" evaluation algorithm
  - Set to highest priority
- Reference the panic alert type on the Global Configuration and in the mobile application section specify the number of seconds for the countdown.

Timed Event Alert

From the general actions menu, crews can create events with an expiration date to indicate that the dispatcher should check on them if the event expires. This may be used if the crew feels they may be in a dangerous or hazardous situation and would like higher visibility from the dispatcher.

Configure the alert type for a timed event according to the following:

- No creation algorithm
- Allow manual close
- Allow snooze and recycle
- Use the "snooze forever" evaluation algorithm
- Set to highest priority

Remote Script Invocation

Data and updates sent to the crew, as well as updates and completion data received from the crew, rely on the Remote Script Invocation (RSI) mechanism supported by the mobile framework.

Data exchange is achieved by calling a framework business service to place an asynchronous request to invoke a service script and its corresponding data on a remote application. The system provides both server and mobile device versions of this service. The server version places the request to be performed on a specific MDT, whereas the mobile device version places the request on the server.

Messages Sent to the Crew

A call back notification functionality is supported for RSI messages issued from the server only. With this functionality, the process posting the RSI message to the crew may request to be notified if delivery of the message to the crew takes too long. For example, this may be used to raise an alert when an activity takes too long to dispatch to a crew.

Messages Received from the Crew

A synchronous form of an RSI call may be issued from the mobile device only, such as when the crew request and start their shift for the day.

On the server, messages sent from the crew are processed by the Remote Script Invocation installation plug-in. This allows a better handling when an application error occurs. The base algorithm persists the message as a crew message business object and if an error is encountered while processing the message a To Do Entry is raised for a user to correct the issue and reprocess the message. Refer to Appendix C for more information about this plug-in.

Mobile Device Registration

You must create an MDT record for each physical device that will communicate with the main server application. Each MDT record is associated with an MDT Type that defines the attributes and capabilities of all devices of that type, such as the display type and GPS settings. The first time a mobile user runs the mobile application from a new device, they will be required to register the device. The system will ask for the MDT Tag and the URL of the server application. (The MDT Tag is a unique, user-defined identifier that is assigned to the device when the MDT record is created on the server application) As part of the registration process, the system places the device's unique ID (derived from the MAC address of the device) in the MDT record on the server. Once the device is registered, the mobile user will be able to log on to the application from that device without further prompting.

During the logon process, a user provides their user name and password for authentication. If authenticated, the system then looks for the most recent version of all active, qualified deployments based on the user's user group, language, and MDT type. The mobile application cannot be deployed to a device unless it has been successfully registered. See the next section for more information on deployment.

When a user starts a shift, the system links both the MDT and the user ID to that shift and uses this association to communicate activity and shift data to the appropriate device.

How Unprocessed Data is Handled

If a user logs off a mobile device when unprocessed data exists, the system notifies the user that the data exists but allows the user to log off and does not remove the data from the device. When a user logs back in to the application from that device, the system does the following:

- If the user attempting to log in owns the existing data (that is, the user is allocated to the crew shift for which the data exists), the data is untouched and the user can resume work. The user should finalize the shift before another shift is started on the device.

- If the user attempting to log in does not own the existing data and a new deployment is available, the system warns the user that replacing the deployment will delete any unprocessed data.

- If the user attempting to log in does not own the existing data and no new deployment exists for the device, the system warns the user that non-finalized crew shift data exists and that continuing with logon will cause the previous shift data to be deleted. This allows the user to cancel the request to start a new shift.

    Note: Once you start a new shift, you cannot access the previous shift.

Mobile Device Log Files

Logs from the runtime mobile application are written to a file stored locally in the mobile application's logs folder. When the content in the file exceeds a configurable limit, the currently used log file is renamed, appending the current date and time to the file name, and a new file is created. When the number of log files exceeds a configurable limit, they are moved to the archived log files folder on the device. Archived log files are kept for a configurable number of days before being purged. Logging parameters (log file size, file count, archival days, etc.), are defined on the Mobile Data Terminal (MDT) record. When a device logs on to the mobile application, these settings are read from the server and saved to a local properties file.

The system does not send logs from the MDT automatically. These log files are only sent on demand, usually for debugging and tracing purposes. To send log files to the server, navigate to the device's MDT portal in the server application and then click the Get MDT Log Files button. This initiates a request to the device to send the active log files to the server. On the server, log files are stored in separate folders for each MDT and may be viewed using the Log Files tab on the MDT maintenance portal.

> **Note**: The MDT log files described in this section are not to be confused with the business audit log records associated with entities such as tasks and shifts. The latter are not files, but rather part of the business entity's completion data that are automatically sent back to the server when work on the entity is completed.

The logging level is specified in the MDT record.

Chapter 11

Deploying the Application to Mobile Devices

The deployment process refers to the packaging and delivery of application components to a mobile device terminal (MDT).

Mobile application development is done by creating application components, such as business objects, UI maps, and service scripts, using the same development tools used for the main application. Once the mobile application components have been developed and tested, they must be packaged together along with all necessary control table records to be delivered to a mobile device terminal (MDT).

A deployment part is a collection of such application components. Deployment parts provide a way to create reusable groups of application components. To create a deployment part, you must first gather the application components together into a bundle and then upload the bundle entities to the deployment part.

Refer to the demonstration database for examples of deployment definitions.

This section includes the following topics:

- *Version Control*
- *Defining Deployment Entities*
- *Creating the Deployment*
- *Downloading Deployments*

Version Control

The MCP communication architecture involves both client side and a server side pieces. These must be compatible or else errors will occur.

To enforce compatibility, the system validates the MCP version as follows:

- When the MDT attempts to connect to the server, the system versioning check validates whether or not the MCP Version on the MDT is compatible with the MCP version on the server. If the versions are not compatible, the device is not allowed to work online and no communication is sent to/from the server. In this situation, the MDT must be upgraded to the correct version before it can work online again. If there was data on the MDT, the user can continue offline and manually collect any data they need before the MDT is upgraded. If a user tries to log on with an incompatible MCP on their device, they receive a message.

- When a deployment is created on the server, it is marked with the version number of the server side MCP Components that created it. Then when a MDT logs on to download a deployment, only deployments with compatible MCP Version can be downloaded. This prevents getting into the situation above.

Warning: Before the server is upgraded, all MDT's should be logged off and all shift information on them should be successful completed for end of shift processing. . This will prevent the need for manual data copying. You should never do server side updates until all MDT Crew Shifts have been completed. Use the crew shift search query option to find all started shifts and coordinate their successful completion before server side upgrade.

Defining Deployment Entities

This section summarizes steps involved in the deployment process. Refer to the online help or user documentation for instructions on how to define each entity. Use the embedded help to display field descriptions.

> **Note**: Before begin defining deployment entities, make sure you have created all application entities to be deployed and have created an export bundle for those entities.

1. **MDT Type:** Create a record for each type of MDT your organization supports in the field. The MDT type record identifies the device type, display type, GPS port (if supported), and settings for logging and synchronization intervals.

2. **MDT:** Create a record for each device that is in use.The MDT record defines the unique identifier for a particular device, logging settings, and MCP runtime details.

3. **Deployment Part**: Create a record for each collection of deployment items, such as business objects, UI maps, scripts, control table records, etc. Items must have been previously bundled. When you create the deployment part, you import the bundle you created earlier. The base product provides preconfigured deployment parts that include the base mobile application components. You only need to configure parts to include your custom entities.

4. **Deployment Type**: Create a record for the application to be deployed. The deployment type record defines the deployment part, authorized user groups, supported MDT types, and message categories for the deployment.

When you have finished defining deployment entities, you must run a batch process to create the deployment.

Creating the Deployment

A deployment can only be created using the batch deployment process. The user creates a batch job submission using the batch code for deployment creation.

Once your deployment type setup is complete, use the batch control process to create a deployment for each specific cut or build of a deployment type for a specific language.

The status of the new deployment is Created. You must manually activate the deployment before it can be deployed to any MDTs. Use the Deployment maintenance portal to view existing deployments and change the status (activate or inactivate).

Once activated, the deployment will be available for download to MDTs.

Downloading Deployments

When the mobile user logs on to the mobile application, the system looks for the most recent version of all active, qualified deployments based on the user's user group, language, and MDT type.

• If no deployments are found, the system displays an authorization error message and prevents the user from continuing.

- If only one deployment is found and it is the same as the one currently deployed, then the deployment is not downloaded.

- If only one deployment is found and it is newer than the one currently deployed or there is none currently deployed, then the deployment is downloaded to the MDT.

- If more than one deployment is found, the user can select the one they want to download. Once the application components defined in the deployment are downloaded to the MDT, the user can begin using the system.

> **Note**: An MDT's current deployment ID is stored in the mdt.properties file in the Data subfolder of the mobile application's program directory. You can force the system to download a new deployment and keep your existing application data by setting the DEPLOYMENT_ID property in this file to REFRESH. This will cause the MCP to re-download a deployment the next time the MDT logs on.

> **Note**: If unprocessed data exists on the device when a user downloads a new deployment, the system warns that the unprocessed data will be deleted if the user continues with the deployment.

Chapter 12

SMS Messaging

Crews may use SMS messaging to carry out their scheduled work. SMS crews communicate via text messages sent from their cell phone.

Logic to process incoming SMS messages and send messages back to the crew resides in the following installation options plug-ins:

- *SMS Receive*

- *SMS Send*

Refer to *Appendix C*: *Algorithm Entities* for more information about these plug-ins.

Sample Scenario

The following scenario illustrates how SMS messaging might be used:

- A crew sends a text message of "Logon". The system logs the crew shift on and sends details of the first task.

- As the crew works the task, they send messages to communication their progress, such as "1" to indicate that they are en route to the task and "2" to indicate arrival at the task location.

- When work is done, the crew sends a text message of "3". In response, the system sends the next task.

  Note: The system sends POUs and breaks to the crew in the same way that it sends activities.

- At the end of the shift, the crew sends a "Logoff" message. Typically, an SMS crew enters completion details at the end of their shift when they return to their office.

Appendix A

Scheduler Parameter Table

This section provides descriptions of all scheduler parameters.

Parameters are grouped by type:

Parameter Type	Description
Appointment Generation	Controls how appointment booking requests from the host are processed.
Connection	Controls how the scheduler sends and receives data to/from client applications.
Cost Control	Defines how to calculate and apply costs used to build the most cost-effective schedule.
Entity	Defines shift-based and task-based parameters that are applied against global parameters.
General Administration	Defines system-wide scheduling parameters.
Logging	Controls how and where the scheduler creates log files.
Map Configuration	Defines mapping parameters used to build and optimize routes.
Optimizer Behavior	Defines parameters that affect optimization of the scheduler.
Optimizer Performance	Defines parameters that affect system performance.
Real Time	Controls the scheduler's real-time functionality.
Reference Time	Controls how and when reference time is updated. (Reference time is the time used by the scheduler as input to cost-calculations that are sensitive to the current time.)
Scheduler Manager	Defines parameters used by scheduler manager functions.
Site	Defines the default (UTC) Coordinated Universal Time and the error translation file used to translate error messages at initialization.

Appointment Generation Parameters

Parameter Name	Description	Detailed Description
slotGenChronologicalFilter	Chronological Filtering	If checked (True), non-chronological appointment windows are removed from (filtered out of) the returned list of appointment windows. If unchecked (False), non-chronological filtering is disabled.
slotGenChronoFilterTol	Chronological Filtering Tolerance	Maximum number of hours an appointment window can start before the previous one. Applicable only when the chronological filtering is enabled.
slotGenTermination	Termination Threshold	Number of candidates that are costed after the cost-limit has been reached during appointment booking generation.
slotGenWinCandMin	Window Candidate Minimum	Minimum number of candidates considered for a single time window by the appointment booking request process.
slotGenWinFilter	Window Filtering	Controls filtering of returned appointment windows based on time window border conditions. **Contained within**: An appointment window must be within the request time window. **Starts within**: An appointment window must start within the request time window. **Ends in**: An appointment window must end within the request time window. **Overlaps**: An appointment window must overlap with the request time window. Note that these conditions apply on top of the initial selection criteria indicating that the task can be scheduled to be worked at some time during the appointment window, based on the availability of suitable crew and task time window.

Connection Parameters

Parameter Name	Description	Detailed Description
clientTimeout	Client Timeout	Inactivity timeout interval. After this number of seconds of no activity, the scheduler will drop the connection to a client. Set to -1 to disable timeout.
io_bufsize	IO Buffer Size	Size (in kilobytes) of the data buffer used by IPC sockets for sending and receiving data. Used to set a sufficient buffer size without wasting memory resources. This is set internally and should not be changed.
maxPackets	Maximum Packets	Maximum number of packets that will be read in one socket poll. This should not be changed.
maxWaitBlock	Maximum Wait Block	Maximum time (in seconds) to wait on a blocked socket before closing the socket.

Parameter Name	Description	Detailed Description
pollTimeInterval	Poll Time Interval	Time interval at which sockets are polled for incoming data. This value should not be changed.
serverMode	Server Mode	If True, sets the mode of operation to Server and Client. In this mode, the scheduler immediately grants connection to all requesting clients. If False, the scheduler first obtains a connection from the server process and only then grants connection to all requesting clients.
socketBufferSize	Socket Buffer Size	Internal socket buffer size for sending data. This is the maximum amount of memory (in kilobytes) that can be allocated per socket. It should be set to at least twice the size of the schedule.

Cost Control Parameters

Parameter Name	Description	Detailed Description
ALLOCATION	Allocation Cost	Global cost applied to each optional task that is not assigned (i.e., on the free list). The initial and final allocation priority parameters, defined on a task's priority profile, act as multipliers against this cost.
BREAK_LATE	Late Break Cost	Exponential cost for taking a break late.
CCAP1 - 5	Cumulative Capacity 1- 5 Cost	The cost of cumulative on-board capacity of the shift assigned to the activity.
DEPOT_SLA_WINDOW	Depot SLA Window	Sets a value to indicate the cost of working a job outside the timeframe of the Depot SLA Window..
DEPOTLATE	Depot Late	Cost as a function of the number of seconds late at Stop. In order to maintain balance with STOP_LATE, the resulting cost is multiplied by the number of Jobs transferred here.
DEPWINCCAP_1 - 5	Depot Window Capacity 1 - 5	Sets a value to indicate the cost of exceeding the capacity limit for a depot time window.
HAUL	Shift Cost	Global cost of utilizing a shift. The Relative Shift Cost parameter, defined on the shift cost profile, can be used as a multiplier against this cost.
IDLETIME	Idle Time Cost	Global cost of a crew being idle.
JOB_ATTR	Job Attribute	Sets a value for the cost when a Job Preference and Job Attribute do not match. For example, when transporting prisoners, it is preferable not to have hardened criminals and juveniles in the same vehicle.
JOBS_LIMIT	Tasks Limit	Exponential cost for exceeding this number of tasks assigned to a shift. The input to this cost function is the number of tasks exceeding the tasks limit. For example: If a shift has a maximum tasks limit of 10 and the sum of all scheduled tasks is 12, then the cost input is 2. If a shift has a maximum tasks limit of 100 and the sum of all scheduled tasks is 102, then the cost input is also 2, so the resulting cost is the same.

Parameter Name	Description	Detailed Description
OVER_SKILL	Over Skill Cost	Flat cost applied for each crew skill level in excess of the required skill level for an assigned task. This is used to discourage the scheduler from assigning crews that are overskilled (either in quality or in quantity) for tasks.
LIFESPAN	Life Span	Sets a value to indicate the cost for late delivery. Cost is represented as a function of the number of seconds a product is on board longer than the maximum- Runlength defined for a Shift. For distribution runs, if a Shift specifies maxrunlength, and an Activity Lifespan, the shortest duration is applied.
OVERTIME	Overtime Cost	Exponential cost as a function of the number of seconds of over-time for a crew's shift. The Relative Overtime Cost parameter, defined on shift's cost profile, can be used as a multiplier against this cost.
PKUP_SERVICE	Arrival Costs	Exponential cost to advance the arrival of an activity within its time window. This is calculated as a ratio of the arrival into the applicable Arrival Time Window. A=Multiplier. Positive values bias towards early arrival, negative towards late arrival. B=The ratio of the time window at which the value that A is multiplied by is doubled. C=Initial step value applied only once at start.
RESERVE_CAPACITY	Reserve Capacity Cost	Exponential cost for exceeding shift's reserve capacity. Reserve Capacity reserves a percentage of shift-time for the on-site time of activities of specified service classes. The system adds together all on-site time for non-reserved activities and all travel-time (whether for reserved activities or not). If this total exceeds the un-reserved part of the shift time, the Reserve Capacity Cost function calculates a cost from the difference.
ROUND	Shift Area Cost	Cost for the geographical size of the area covered by a shift's schedule of tasks. Promotes geographical clustering of tasks in a shift. In order to determine the size of a shift's currently scheduled area coverage, the scheduler draws a rectangle encompassing all the tasks associated with the shift and then calculates the length of the diagonal. This Shift Area Cost is applied per meter of the length of this diagonal.
ROUND_LENGTH	Shift Area Time Cost	Cost of time spent from the first to last task of a shift. Promotes geographical clustering of tasks in a shift.
ROUND_ZONE	No Task In Service Area Cost	Flat cost for each shift that has no tasks within the shift's covered service areas. This cost is calculated in conjunction with the Service Area Cost parameter.
RSRC_ATTR	Resource Attribute Cost	Cost of having a task on a shift where the task's resource-related attributes do not match those on the shift. The cost is applied once for each task with a conflicting attribute.
RUN	Depot Run Cost	Cost of activating a run.

Parameter Name	Description	Detailed Description
RUN_SEPARATION	Depot Run Separation	Applies a cost to loading goods during a Distribution-run. cost = fABC(l/ c) * r; l = load added at Stop c = capacity available on Resource r = remaining Stops within the run In general this parameter minimises having load for multiple runs concurrently on board: Main example is to minimise (but not prevent) collecting return goods before the last delivery of the day has been made.
SERVICE_CLASS	Service Class Cost	Cost of having a task on a shift where the task does not belong to any of the service classes allowed to be performed by the shift. The cost is applied once for each task with a conflicting attribute.
SHIFT_PROMOTION	Shift Promotion Cost	Global cost of promoting a shift. This allows a preference to be applied for scheduling tasks to earlier shifts. The task-based Relative Shift Promotion Cost parameter, defined on activity type, can be used as a multiplier against this cost.
SITE	Site Cost	Cost for non-sequential visits made to the same site. Promotes working of all activities at the same site by a single crew in a single visit.
STOP_SLA_WINDOW	SLA Window Cost	Flat cost of working an activity outside its SLA time window.
STOPLATE	Late Cost	Global cost of late arrival to an activity. This exponential cost is a function of the number of seconds late for arrival. The Relative Late Cost parameter (defined at the task level) can be used as a multiplier against this cost.
TRAVELDIST	Travel Distance Cost	Global cost of time spent travelling, includes time contributions by site delay. The Relative Travel Time Cost parameter, defined on the shift cost profile, can be used as a multiplier against this cost.
TRAVELTIME	Travel Time Cost	Global cost of time spent travelling, includes time contributions by site delay. The Relative Travel Time Cost parameter, defined on the shift cost profile, can be used as a multiplier against this cost.
WAIT_TIME	Wait Time Cost	Quadratic cost for the shift being idle. The shift-based Cost Idle parameter, defined on the shift cost profile, can be used to control the cost on a shift-by-shift basis.
WAIT_TIME_PAST	Wait Time Past Cost	Quadratic cost for shift being idle in the past.
WINDOW	Window Cost	Global cost of using a task's arrival time window. The Relative Window Cost parameter (defined in the task time window) can be used as a multiplier against this cost.
ZONE	Service Area Cost	Flat cost for each task outside the shift's preferred service areas. Used to encourage crews to work in their preferred service areas.

Entity Parameters

Parameter Name	Description	Detailed Description
COST_WAIT_SHIFT	Cost Idle	The parameter is defined on the shift cost profile. If checked (True), the scheduler considers the cost of idle time for a specific shift's crew, weighted according to the "time from now" factor. No cost is applied if the idleness accrues further than the wait horizon time interval and, within this horizon, the applied cost is stronger if the idleness is closer to the current time. If unchecked (False), idle time cost is not considered.
DIST_COST	Shift Distance Cost	Shift cost profile factor used as a multiplier against the global Travel Time Cost. For example, a relative distance cost of 3 means that the distance cost for a particular shift is three times more than the distance cost defined globally.
LATE_COST	Relative Late Cost	Relative cost of arriving late to this activity. For example, a relative late cost of 3 means that the cost of late arrival for a particular activity is three times more than the late cost defined globally. This is used to discourage scheduling activities with planned late arrival time. This is set at the task level.
OVERTIME_COST	Relative Overtime Cost	Shift cost profile factor used as a multiplier against the global Overtime Cost parameter. For example, a relative overtime cost of 3 means that overtime for a particular shift is three times more than the overtime cost defined globally.
REL_EFFICIENCY	Relative Efficiency	Defines a mobile worker's efficiency relative to expected efficiency for mobile workers of this type. For example, if a mobile worker is twice as efficient as other mobile workers of this type, the Relative Efficiency is 2. A crew shift's relative efficiency is the average of all its allocated mobile workers' relative efficiencies.
RELSPEED	Relative Speed	Defines a vehicle's speed relative to the expected speed for vehicles of this type. For example, if a vehicle is twice as fast as the expected speed for vehicles of its type, the Relative Speed is 2. A crew shift's relative speed is the lowest of all its allocated vehicles' relative speeds.
SHIFT_COST	Relative Shift Cost	Shift cost profile factor used as a multiplier against the global Shift Cost parameter. For example, a Relative Shift Cost of 3 means that the cost of activating the shift is three times more than the shift cost defined globally.
SLA_FLEXIBILITY	SLA Flexibility	Period of time before and after the preferred time window (PTW) within which an activity must be scheduled to avoid incurring added cost. This is defined at the activity level. If arrival is outside this time period, the scheduler applies a fixed maximum cost (specified in the SLA Priority field for the activity type). For example, if the SLA Flexibility is set to 20 minutes, then scheduled arrival must be no more than 20 minutes before the start of this preferred time window or 20 minutes after the end of this window; otherwise, the SLA Priority cost is applied. Format: hhmmss.

Parameter Name	Description	Detailed Description
SLA_PRIORITY	SLA Priority	Fixed cost that will be applied for working an activity outside SLA Flexibility period. The cost is applied when the job is done. This is defined at the activity type level.
TASK_SHIFT_PROMOTION	Relative Shift Promotion Cost	Task-based cost factor used as a multiplier against the global Shift Promotion Cost. A higher value puts more pressure on the scheduler to find a cheaper (i.e., earlier) shift for tasks of this type. This value is defined at the activity type level.
TIME_COST	Relative Travel Time Cost	Shift cost profile factor associated with travel time for a crew shift. This is used as a multiplier against the global Travel Time Cost. For example, a Relative Travel Time Cost of 3 means that the cost of travel time for a particular shift is three times more than the travel time cost defined globally.
WINDOW_COST	Relative Window Cost	This task-based cost factor is used as a multiplier against the global Window Cost. It defines the preference of an activity's time window relative to other time windows. For example, a Relative Window Cost of 3 means that the cost of using an activity's particular time window is three times more than the Window Cost defined globally.

General Administration Parameters

Parameter Name	Description	Detailed Description
testLevel	Test Level	Refers to additional code for performing run-time consistency checks. 0: no optional tests 1: consistency tests that have a negligible impact on performance 3: consistency tests that have some impact on performance 5: consistency tests that may significantly alter performance Normal setting is 0 and should not be changed in a production environment.

Logging Parameters

Parameter Name	Description	Detailed Description
autosaveDirectory	Auto Save Directory	Directory path name in which auto save files are stored.
autosaveFiles	Auto Save Files	Number of auto save files to store.
autosaveInterval	Auto Save Interval	Interval (in minutes) between each autosave. When set to zero, auto save is disabled.
dualAutosaveDirectory	Backup Auto Save Directory	Directory path name in which backup auto save files are stored.
logCompress	Log Compression	File compression method for log files. Compression of log files is available only for UNIX installations. The default is gzip.
logFilePrefix_MSG	Log File	Path and prefix of log files. Must be a valid and accessible location.
logHours_MSG	Log Hours	Number of hours each log file spans. Log files are usually sent as email attachments. Setting this value to 1 hour will keep file size to a minimum.
logLevel	Log Level	Level of detail that will be logged for the process. Log levels: 0=No logging 1=Logs real-time/operator events/status changes, shift/ task status changes, etc. 2=Logs operation of automated processes impacting schedule execution, dead-reckoning, auto dispatch, etc. 3=Operation of optimizer 4=Anything else 5=Spare 6-9=Same as levels 1-4, but accompanied by performance penalties or excessive output
logToFile_MSG	Log to File	If checked (True), saves all log messages to a file. This should always be True.
noLogStdOut	Log to Standard Output	If checked (True), logging to standard output is enabled. If unchecked (False), logging is disabled.
numLogFiles_MSG	Number of Log Files	Number of log files stored.
rxHipLogging	Received HIP Logging	If checked (True), all the HIP packets received from the scheduler are logged. If unchecked (False), received packets are not logged.
txHipLogging	Transmitted HIP Logging Flag	If checked (True), all HIP packets sent from the scheduler are logged. If unchecked (False), sent packets are not logged.

Map Configuration Parameters

Parameter Name	Description	Detailed Description
logMap	Log Map	If checked (True), the system logs the number of map nodes, links, and matrix rows loaded. If unchecked (False), these items are not logged.
map	Map File	Directory path and filename of the connectivity map for use by the scheduler. A map consists of a list of nodes (geographical locations, suburbs, etc.) and a list of inter-node links (travel paths) whose direction, distance, and travel time are known. This data is used to create the schedule.
matrixDistFactor	Matrix Distance Factor	Distance factor (integer multiplier) used in the objective function to calculate the best route. If this is set to zero, the fastest routes are calculated. If Matrix Time Factor matrixTimeFactor is set to zero, the shortest routes are calculated. The contribution of distance and time are approximately equal when the ratio of Matrix Time Factor / Matrix Distance Factor matrixTimeFactor / matrixDistFactor is approximately 20. Do not change unless advised by Oracle.
matrixSegmentation	Matrix Segmentation	If checked (True), segmentation matrix is generated only for tasks that are compatible with the shifts. If unchecked (False), segmentation matrix will be generated for all tasks in the system.
matrixTimeFactor	Matrix Time Factor	Time factor (integer multiplier) used in the objective function to calculate the best route. If this is set to zero, the shortest routes are calculated. If Matrix Distance Factor is set to zero, the fastest routes are calculated. The contribution of distance and time are approximately equal when the ratio of Matrix Time Factor / Matrix Distance Factor matrixTimeFactor / matrixDistFactor is approximately 20. Do not change unless advised by Oracle.
matrixTollFactor	Matrix Toll Factor	Toll factor (integer multiplier) used in the objective function to calculate the best route. Used to discourage specific roads or road-segments.
matrixUpdateInterval	Matrix Update Interval	Interval (in seconds) to check the matrix for unused cells.
matrixVersion	Matrix Version	Matrix implementation version. Options are: Full matrix, no segmentation High connectivity Low connectivity Sparse matrix, much segmentation Sparse matrix, symmetrical segmentation The default is set by Oracle for best map performance and should not be changed.
offMapSpeed	Off Map Speed	Used if the specified coordinates do not match the road-network. This is the travel speed (meters/per second) between a task location and the nearest road or other task location.
routeCacheMemoryLimit	Route Cache Memory Limit	Maximum amount of memory the route cache will use.

Parameter Name	Description	Detailed Description
warnDist	Warning Distance	Sets the maximum distance that task locations are allowed to be from the nearest map road before a warning is issued.

Optimizer Behavior Parameters

Parameter Name	Description	Detailed Description
applyMinTravelTimeToNonStop	Apply Minimum Travel Time to All Tasks	If checked (True), minimum travel time between tasks is applied for all tasks. If checked (False), minimum travel time is applied between activity tasks only.
arrivalMargin	Arrival Margin	Desired number of minutes prior to the start of the time window for arriving at a task.
bUEC	Unexpected Event Handling	If checked (True), the system automatically identifies and resolves issues caused by unexpected events. For instance, a long duration task may be unexpectedly allocated to a shift, introducing 3 hours of overtime. The system will then identify activities that may be transferred to other shifts, in order to relieve overtime.
costByRound	Cost by Round	Determines how round-based costs are bounded. Set to TRUE to have round costs bounded by DepotStops. Set to FALSE to have round costs bounded by LogStops. "Round" based costs are usually calculated on a depot-based run, but may optionally be calculated on an entire Shift.
DepotCapacityTWRule	Depot Capacity Time Window Rule	Determines which DepotTW an Activity's load is assigned to. Legacy: earliest DepotTW that overlaps the Shift in AB; Arrival DepoTW in Scheduling; EarliestOverlappingShift: earliest DepotTW that overlaps the Shift; EarliestOfDay: earliest DepotTW that starts within the Day;
depotLateByArrival	Calculate Depot Late Cost by Arrival Time	Set to true to calculate the cost of being late to a depot starting from the arrival time.
firstJobDistanceFactor	First Task Distance Factor	Multiplier of the distance cost applied to the first task on a shift. Promotes the initial assignment of activities close to a crew's starting location.
idbOptLevel	Optimization Level	Various operations (such as allocating an activity to a shift, initial assignment of a new activity to a shift, etc.) require the sequencing of tasks on a shift to be optimized. This parameter determines how much effort is expended on this kind of optimization. Options are None, Standard, Thorough.
immediateAssignment	Immediate Assignment	If checked (True), all mandatory tasks will be assigned to a shift immediately on receipt by the scheduler. If unchecked (False), tasks will be placed on the free list for later, gradual insertion.
loadFactor	Load Factor	Sets a value to define the factor by which over-loading is allowed. This is used in the Stop-Shift compatibility check.

Parameter Name	Description	Detailed Description
maxUpLoadInterval	Maximum Upload Interval	Maximum period (in seconds) between improvement to a solution and its uploading to the database. This is used to ensure that the scheduler periodically saves the improvements to the current solution to the database so they are available to other processes. If set to zero, every solution improvement is saved to the database immediately.
minTravelTime	Minimum Travel Time	Minimum travel time (in seconds) between tasks. The scheduler will adjust the travel time if it falls below this value. The system issues a warning if you try to set this value to greater than 600 seconds (10 minutes), as it can have a severely adverse effect on the schedule.
nJobsInsert	Number of Tasks Inserted	When there are multiple jobs on the free list, this is the maximum number that will be included in the plan at any one time when not in an Immediate Assignment mode.
planMaxMergeDistance	Plan Max Merge Distance	Sets a value to define a distance so that the plan-optimizer creates a new run when the crew travels above that duration. Entered in seconds. This is used to control Scheduler behavior in terms of breaking runs in two.
planNewRunProbability	Plan New Run Probability	Sets a value to define the minimum probability that the plan-optimizer will create a new run. This is used to control scheduler behavior in terms of breaking runs in two.
relOverCap	Relative Overcapacity	Depot-window capacity calculated based on relative overcapacity.
sameSiteRestrict	Same Site Factor	Multiplier to control allowed ratio between number of sites and number of tasks in a cluster. Do not change without consulting Oracle.
ShiftWinExtension	Shift Window Extension	Number of minutes the scheduler can artificially extend the shift time window for the purpose of optimization when considering compatibility with the tasks. This defines the mismatch allowed between the shift window and task time windows. If this is set to zero, the shift and task time windows must overlap for allocation to be possible. If this is greater than zero, then a time window mismatch within this number of minutes will be acceptable to the scheduler when making recommendations.
stopLateByArrival	Task Late By Arrival	If checked (True), the Late Cost is applied to tasks based on comparison with arrival time. If unchecked (False), the Late Cost is applied to tasks based on the predicted departure time.
timeWindowsSegmentation	Time Windows Segmentation	Defines a boundary (in seconds) to segment time between time windows for calculating the time of execution, late/idle time and cost. A value of -1 splits time in the middle between time windows.
UEC_costThreshold	Unexpected Event Handling Threshold	Cost threshold that must be reached in order to activate the Unexpected Event Handling functionality.

Parameter Name	Description	Detailed Description
waitHorizon	Wait Horizon	The time interval from the current time when the Wait Time Cost is applied. The default is two hours.

Optimizer Performance Parameters

Parameter Name	Description	Detailed Description
allocMove_initRelFreq	Allocation Initial Frequency	Frequency at which the allocation shuffler is activated when the scheduler starts. The default is set to 0.1 internally and should not be changed without consulting Oracle.
allocMove_lastRelFreq	Allocation Final Frequency	Frequency at which the allocation shuffler is activated when the scheduler finishes. The default is set to 0.1 internally and should not be changed without consulting Oracle.
clusterMove_initRelFreq	Cluster Initial Frequency	Frequency at which the cluster shuffler is activated when the scheduler starts. The default is set to 1.0 internally and should not be changed without consulting Oracle.
clusterMove_lastRelFreq	Cluster Final Frequency	Frequency at which the cluster shuffler is activated when the scheduler finishes. The default is set to 1.0 internally and should not be changed without consulting Oracle.
costLimit	Cost Limit	The scheduler is suspended when the solution cost goes below this value.
deAllocMove_initRelFreq	Deallocation Initial Frequency	Frequency at which the deallocation shuffler is activated when the scheduler starts. The default is set to 0.1 internally and should not be changed without consulting Oracle.
deAllocMove_lastRelFreq	Deallocation Final Frequency	Frequency at which the deallocation shuffler is activated when the scheduler finishes. The default is set to 0.1 internally and should not be changed without consulting Oracle.
fillIdle_initRelFreq	Fill Idle Time Initial Frequency	Frequency at which the fill idle shift shuffler is activated when the scheduler starts. The default is set to 0.1 internally and should not be changed without consulting Oracle.
fillIdle_lastRelFreq	Fill Idle Time Final Frequency	Frequency at which the fill idle shift shuffler is activated when the scheduler finishes. The default is set to 0.1 internally and should not be changed without consulting Oracle.
fillShift_initRelFreq	Fill Shift Initial Frequency	Frequency at which the fill shift shuffler is activated when the scheduler starts. The default is set to 0 internally and should not be changed without consulting Oracle.
fillShift_lastRelFreq	Fill Shift Final Frequency	Frequency at which the fill shift shuffler is activated when the scheduler finishes. The default is set to 0 internally and should not be changed without consulting Oracle.
fillTheGap_initRelFreq	Fill Gap Initial Frequency	Frequency at which the fill gap shuffler is activated when the scheduler starts. The default is set to 0 internally and should not be changed without consulting Oracle.
fillTheGap_lastRelFreq	Fill Gap Final Frequency	Frequency at which the fill gap shuffler is activated when the scheduler finishes. The default is set to 0 internally and should not be changed without consulting Oracle.

Parameter Name	Description	Detailed Description
maxTerm	Maximum Exponential Rate	Places a cap on the rise of exponential costs. When the B value in exponential cost has been used this many times to double the A value, the latter will no longer be doubled. The default value of 0 denotes true exponential costs.
nOptRuns	Number of Optimization Runs	Number of distinct optimization runs that the scheduler should perform to get itself out of local minima. This applies only when Scheduler Mode is set to either Fixed Real Time or Fixed CPU Time. The default is set to 1 internally and should not be changed without consulting Oracle.
pcCascadeChance	Cycle Cascade Chance	Average number of additional scheduler subcycles. Setting this to a larger number allows the scheduler to use a broader strategy when searching for cost improvements. If a particular solution appears stagnant, then some additional scheduler sub-cycles may free the restriction. This is set to 0.2 internally and should not be changed without consulting Oracle.
relClusterSize	Relative Cluster Size	Multiplier for the number of tasks in cluster. The scheduler generally tries to improve the schedule by selecting a set of geographically close Activities. This set is referred to as "cluster".
runMove_initRelFreq	Run Move Initial Relative Frequency	Sets a value to define the initial relative frequency for the runMove shuffler.
runMove_lastRelFreq	Run Move Last Relative Frequency	Sets a value to define the final relative frequency for the runMove shuffler.
schedDuration	Scheduler Duration	Number of seconds that scheduler should continue to optimize. If the Scheduler Mode parameter is set to Variable or Forever, this value controls the temperature. If Scheduler Mode is set to Fixed Real Time or Fixed CPU Time, this value limits the optimization time to the required amount of CPU-time or wall-time. Optimization time is also limited by recalculating the temperature based on time left.
schedHeatRate	Scheduler Heat Rate	Temperature increase/unit drop in cost. The resulting value can be seen in the scheduler log as "t=nnnn." If the resulting value does not rise on making large cost improvements, then scheduler heat rate may need to be increased.
schedMaxTemp	Scheduler Maximum Temperature	Highest scheduling temperature. Temperature used when the scheduler is started and highest temperature during normal scheduling. See Scheduler Heat Rate. Higher temperatures generate better final schedules, but take longer to generate good/acceptable schedules.
schedMinTemp	Scheduler Minimum Temperature	Lowest scheduling temperature. Temperature used when scheduling nears completion.

Parameter Name	Description	Detailed Description
schedMode	Scheduler Mode	The length of time that the scheduler will continue to improve the solution before terminating.
		Forever: Schedules into infinity, so progress is always 0.
		Variable: Finds the remaining amount of time. It is linearly related to the log of the relative temperature.
		Fixed Real Time: Scheduling will terminate at the specified wall-time.
		Fixed CPU Time: Scheduling will terminate at the specified CPU-time.
segmentMove_initRelFreq	Segment Initial Frequency	Frequency at which the segment shuffler is activated when the scheduler starts. The default is set to 1.0 internally and should not be changed without consulting Oracle.
segmentMove_lastRelFreq	Segment Final Frequency	Frequency at which the segment shuffler is activated when the scheduler finishes. The default is set to 1.0 internally and should not be changed without consulting Oracle.
shiftMove_initRelFreq	Shift Initial Frequency	Frequency at which the shift shuffler is activated when the scheduler starts. The default is set to 0.1 internally and should not be changed without consulting Oracle.
shiftMove_lastRelFreq	Shift Final Frequency	Frequency at which the shift shuffler is activated when the scheduler finishes. The default is set to 0.1 internally and should not be changed without consulting Oracle.
stopsInSearchArea	Tasks In Search Area	Number of tasks that will determine the search area. The search area limits the maximum number of compatible tasks that can be moved by a cluster at one time. The default is set to 50 internally and should not be changed without consulting Oracle.
tspClassTime	Task Grouping Travel Time	Maximum travel radius (in seconds) for a group of tasks. Used for grouping tasks, this determines how far apart tasks can be and still belong to the same group. If the tasks have overlapping time windows and lie within the specified travel radius, then the scheduler will group them for allocation to the same crew.

Real Time Parameters

Parameter Name	Description	Detailed Description
AD_MaxLoad	Auto Dispatch Maximum Load	Maximum load as a ratio of vehicle capacity for Auto Dispatch
AD_MinLoad	Auto Dispatch Minimum Load	Minimum load as a ratio of vehicle capacity for Auto Dispatch
AD_Rule	Auto Dispatch Rule	In case goods are scheduled to be picked up at a Depot before the previous run has been completed, this flag controls whether the Scheduler dispatched both runs at the same time, or one by one. ONE_AT_A_TIME: single run only; COMBINED: logical run;

Parameter Name	Description	Detailed Description
AD_StabilityPeriod	Auto Dispatch Stability Period	Minimum period of stability (in seconds) at a task location before auto dispatching. A stability period is applied to each task that has been nominated for auto dispatching. A task is considered stable if its shift or work sequence has not been changed for at least the specified stability period.
autoDirectInterval	Auto Dispatch Interval	Determines how often (in seconds) the automatic dispatching module checks for tasks to dispatch.
autoDirectOnComplete	Auto Dispatch On Completion	If checked (True), the scheduler will auto dispatch the next task when the previous task is completed. If unchecked (False), auto dispatch will occur when the previous task is started.
autoDirectTimeHorizon	Auto Dispatch Time Horizon	Maximum allowable idle time (in minutes) before automatically dispatching a task. This applies only when Enable Auto Dispatch is True. If the task to be dispatched next will be arrived at more than this number of minutes before it can be started, then it is auto dispatched only after the time period specified here.
breakDespatchMode	Break Dispatch Mode	Defines the timing in which breaks are automatically dispatched. **Dispatch all at Shift start**: All breaks on the started shift are auto dispatched regardless of the shift's drip horizon. **Dispatch all within horizo**n: Only breaks within the shift's drip horizon are auto dispatched.
closeDistributionRun	Auto Dispatch Close Distribution Run	Close run when AutoDirect DIST stop.
emergency_despatch_mode	Emergency Dispatch Mode	Defines emergency task dispatch considerations. **Allow Onsite Crews:** An emergency task can be dispatched to a crew even if the crew is working on another task. **Disallow Onsite Crews**: An emergency task will not be dispatched to a crew that is currently onsite on another task.
emergency_despatch_time	Emergency Dispatch Time Left	The scheduler will not consider a crew for an emergency task if the estimated time to complete its current activity is greater than this value. If set to 0, there is no limit.
enableAutoDirect	Enable Auto Dispatch	If checked (True), auto dispatching of activities is enabled. If unchecked (False), auto dispatch is disabled.
pouDespatchMode	Non Productive Task Dispatch Mode	Defines the timing in which non productive tasks (NPT) are automatically dispatched. Options are: Dispatch all on shift start: All NPTs on the started shift must be auto dispatched regardless of the shift's drip horizon. Dispatch all within horizon: NPTs are auto dispatched only if they are within the shift's drip horizon. This is the default.

Parameter Name	Description	Detailed Description
meetingDespatchMode	POU Dispatch Mode	Defines the timing in which periods of unavailability (POU) are automatically dispatched. Valid modes: Advance: All POUs on the started shift must be auto dispatched regardless of the shift's drip horizon. Horizon: POUs are auto dispatched only if they are within the shift's drip horizon. This is the default.
processAllAtSite	Process All at Site	If checked (True), all consecutive tasks at a site will be processed all at once. If unchecked (False), each arrival for the same site task will be processed one at a time. Applies to dead-reckoning only.
realTime	Real Time Mode	Controls real time and dead-reckoning behavior. Schedule crews towards their destination: No work can be planned before now and crews are dead-reckoned towards dispatched tasks. This is the default. Complete activities as scheduled: Same as above, but additionally simulates crews working tasks. Tasks are automatically started and completed at the scheduled times. Real time off: No simulated crew activity. Crew times and positions remain unchanged.
timeInterval	Crew Location Time Interval	Time interval, in minutes, for relocating the crew by the dead-reckoning subsystem. The location of the crew is determined (in a real-time operation) by dead-reckoning and the occurrence of task and shift milestone events. This parameter also controls the frequency at which a task's schedule is updated for estimated time changes.
vehicleTimeToActive	Crew Back In Service	Controls when an out-of-service crew is assumed to be back in service. If set to zero, the crew is automatically assumed to be back in service at the estimated time provided by the crew before going out of service. If set to a non-zero value, the crew is assumed to be out of service until it actively reports back that it is in service.
waitForEnroute	Wait for Enroute	If checked (True), the system only dead-reckons towards tasks once they become Enroute. If unchecked (False), the system also dead-reckons towards tasks that are in Closed or Dispatched state.

Reference Time Parameters

Parameter Name	Description	Detailed Description
intervalRT	Interval Reference Time	Interval (in hours) with which the reference time is updated. This is set internally to 2 hours and should not be changed without consulting Oracle.
startTimeRT	Start Time Reference Time	The time of day at which reference time is synchronized with wall-time. Reference time is used as input to various wall-time dependent functions, like TDC. Reference time is advanced at the interval specified in the Interval Reference Time parameter.

Scheduler Manager Parameters

Parameter Name	Description	Detailed Description
appointmentBookingCost	Appointment Booking Maximum Cost	Sets a maximum so that the scheduler does not return an Appointment Booking when the cost exceeds this value.
chooserMaximumCost	Crew Shift Chooser Maximum Cost	Sets a maximum so that the scheduler does not return a shift when the cost of allocating the activity to the shift exceeds this value.
cleanseInterval	Schedule Manager Cleanse Interval	Sets a value, in seconds, to define the interval period at which the Schedule Manager checks to remove old data from the scheduler.
conditionalBookingCost	Conditional Booking Maximum Cost	Sets a maximum so that the scheduler does not return a Conditional Booking when the cost exceeds this value.
shiftDependOffset	Shift Dependency Offset	Sets a value to specify the duration between shifts that causes them to be considered as linked. When this parameter is set to -1 shifts are never considered linked, even if their time windows overlap.

Site Parameters

Parameter Name	Description	Detailed Description
errorTranslationFile	Error Translation File	Directory path and filename of the error translation file used to translate error message text at initialization.
hipDefaultUTCoffset	Default UTC Offset	Default UTC offset for packet field, in the format: (-)HHMM or (-hhmmss. The default is the application time zone. The scheduler only deals with absolute, not abstract, times. Use this to pass an optional UTC offset when exporting time data.

Appendix B

System Setup Quick Reference Table

This document lists and describes all objects that must be defined as part of the ORS setup process. It identifies the order in which objects should be defined and any prerequisites for setup. Finally, it identifies all objects associated with or referenced by each setup object, thus providing a useful map of the relationship between objects in the system.

This document is divided into the following sections:

- *Administration Setup and Maintenance*: These setup tasks are typically performed by an admin user and are accessed from the Admin Menu. The objects are defined at implementation and then updated only as needed when system configuration or foundation data changes.

- *Resource Setup and Maintenance*: These setup tasks are typically performed by a resource planner and are accessed from the Main Menu under the Resource Management submenu. The objects, which include crews, mobile workers, vehicles, and shifts, are defined during initial system setup and updated regularly as new resources are added or changes are made to existing resources.

    Note: All basic Framework setup, including system and database setup and any modifications or extensions to base business objects, must have been completed before beginning setup tasks for Oracle Real-Time Scheduler. See the Framework documentation for more information.

- *Transfer of Goods Setup and Maintenance*: These setup tasks are typically performed by an admin user and are accessed from the Admin Menu. The objects, which include depot, depot profile, capabilities, etc are defined during initial system setup and implementation and are updated as new depots are added.

Setup Sequence

In the setup tables that follow, the **Sequence** column displays the following codes:

L1 = Object has no ORS setup prerequisites and should be defined before L2-L6 objects.

L2 = Object has some L1 prerequisites and should be defined after all L1 objects have been defined and before L3 objects.

L3 = Object should be defined after all L1 and L2 objects have been defined.

L4 = Object should be defined after all L1, L2, and L3 objects have been defined.

L5 = Object should be defined after all L1, L2, L3, and L4 objects have been defined.

L6 = Object should be defined after all other objects have been defined.

Administration Setup and Maintenance

To access the maintenance portals for the objects in this section, do one of the following:

- If you are using functional menus, select **Admin Menu>[**_Functional Menu_**]>[**_object name_**]**

- If you are using alphabetical menus, select Admin Menu>[_object name_]

The [_Functional Menu_] and [_object name_] are provided in the appropriate columns in the following table.

Seq	Object	Functional Menu	Description	Prerequisites	Associated with / Referenced by
L1	Alert Type	Service Management	A type of alert (situation requiring user attention or intervention)	None	Alerts, KPIs Global Configuration
L1	Appointment Booking Group	Service Management	Defines valid time windows for appointment booking	None	Activity Types ABG Scheduler Read algorithm
L1	Country	General	Your organization's country.	None	
L1	Crew Type	Resource Management	Type of crew	None	Crews
L1	Currency	Financial	Your organization's native currency.	None	
L1	Dispatcher Type	Resource Management	Type of dispatcher	None	Dispatchers
L1	Display Profile	General	Controls how dates, times, and numbers displayed.	None	
L1	Equipment	Resource Management	Non-human mobile resource capability used by a crew	None	Vehicles, Vehicle Types (assumed equipment), Activity Types (equipment required)
L1	Feature Configuration	System	Used to configure special zones, including the Gantt and Calendar zones. Refer to the Feature Configuration section earlier in this document.	None	
L1	Language	General	The language to use for this implementation.	None	
L1	Parameter Definitions	Scheduler	Parameters used by the scheduler. A complete set of definitions and their default values are shipped with the product. This setup step is only required if you need to override the user minimum/maximum values.	None	Scheduler Configuration

Seq	Object	Functional Menu	Description	Prerequisites	Associated with / Referenced by
L1	Period of Unavailability (POU) Type	Resource Management	A period of time in which a crew is planned not to perform work, such as while attending a meeting	None	Periods of Unavailability, Period of Unavailability Templates
L1	Priority Profile	Service Management	Allocation factors used to prioritize the scheduling of activities	None	Activity Types
L1	Remark Type	Service Management	Type of activity completion remark	None	Activity Types
L1	Service Area	Service Management	A logical boundary of an organization's territory that may or may not be based on geography	None	Mobile Workers, Crew Shifts and Crew Shift Templates, Activities, Periods of Unavailability (POUs), Dispatch Areas, Shift Cost Profiles, Scheduler Areas Override Time Windows
L1	Shift Type	Resource Management	Type of crew shift	None	Crew Shifts and Crew Shift Templates
L1	Skill	Resource Management	Human skill required to complete certain types of work	None	Mobile Workers, Mobile Worker types (assumed skills), Activity Types (skills required)
L1	Status Reason	System	The reason why an entity transitioned to its current state. This setup step is only required if the base set of status reasons is not adequate for your needs.		Business object's lifecycle
L1	Task Type: Break Type	Service Management	A type of break	None	Crew Shift Templates, Break tasks
L1	Task Type: Non-Productive Task Type	Resource Management	A type of a non productive time task a crew is planned not to perform shift related work, such as cleaning or loading vehicles.	None	Non Productive Tasks
L1	Task Type: POU Task Type	Service Management	A type of period of time unavailability (POU) task	None	POU Tasks, Global Configuration

Seq	Object	Functional Menu	Description	Prerequisites	Associated with / Referenced by
L1	Time Zone	General	Your organization's base time zone.	None	
L1	To Do Role	General	Used to associate users with To Do entries.	None	
L1	Work Calendar	General	The work calendar for your organization, which identifies your public holidays and is used for scheduling	None	
L1	Extendable Lookup		Various aspects of the application require extendable lookup custom values to be set up for certain functionality to work.		
L2	Installation Options	System	Control various aspects of the system. Refer to the Installation Options section earlier in this document.	Time Zone, Language, Currency	
L2	Key Performance Indicator	Service Management	Quantifiable measurement that supports dispatcher decision making and in-day exception handling.	Alert Type (if alert-based)	Dispatcher
L2	Location	Resource Management	Predefined physical locations. Use the Location maintenance portal, accessible from the Admin menu, to configure common locations.	Time Zone Country	Crew Shifts, Crew Shift Templates (as logon/logoff locations), POUs (as event locations)
L2	Master Configuration	System	Enables an implementation to capture various types of information in the system. For ORS, edit the Global Configuration record to suit your needs.	Alert Type, POU Task Type	
L2	MDT Type	Device Management	Defines a type of mobile data terminal, including display options, and synchronization details	None	MDTs, Deployment Types
L2	Mobile Worker Type	Resource Management	Type of mobile worker	Skills	Mobile Workers
L2	Scheduler Area	Scheduler	Predefined set of service areas and optimization information.	Service Area	Scheduler
L2	Scheduler Configuration	Scheduler	Predefined set of parameters and settings used by one or more schedulers.	Parameters (cost control)	Scheduler
L2	Task Type: Activity Type	Service Management	A type of task performed at a particular geographic location.	Priority Profile, Appointment Booking Profile, Skill, Equipment, Remark Type, To Do Type and To Do Roles	Activities (Activity tasks), Service Classes

Seq	Object	Functional Menu	Description	Prerequisites	Associated with / Referenced by
L2	User	Security	Defines a user's user groups, data access roles, portal preferences, default values, and To Do roles	Language, Display Profile, To Do Roles	
L2	Vehicle Type	Resource Management	Type of vehicle	Equipment	Vehicles
L3	Mailing Lists	Mail Management	A list of users for mail distribution.	User	
L3	MDT	Device Management	Defines attributes of this mobile device terminal, including registration/tag/ serial number, system logging data, etc.	MDT Type	Crew Shift
L3	Scheduler	Scheduler	An individual instance of the ORS scheduler.	Scheduler Configuration< Scheduler Area	
L3	Service Class	Service Management	Broad categorization of activity types, can be used to restrict a crew shift to certain activity types.	Activity Type	Shift cost profile, Crew Shifts and Crew Shift Templates, Dispatch Area, Activity Type
L3	User Group	Security	A group of users who have the same degree of security access	User	
L4	Deployment Part	Device Management	A collection of deployment items, such as business objects, UI maps, scripts, administrative data used on the device, etc.	Any referenced entities	
L4	Dispatch Area	Service Management	Set of service areas and service classes to be monitored	Service Areas, Service Classes	Dispatchers, Dispatch Shift
L4	Mail Templates	Mail Management	Predefined mail messages.	Mailing Lists if you are using them on templates	
L4	Shift Cost Profile	Resource Management	Set of shift-based cost factors used as multipliers against global costs	Service Areas, Service Classes	Crew Shifts and Crew Shift Templates
L5	Deployment Type	Device Management	Defines attributes of an application to be deployed, including authorized user groups, supported MDT types, etc.	Deployment Part, Message Category, MDT Type, User Group	Deployment
L6	Deployment	Device Management	A specific cut or build of a deployment type for a specific language.	Deployment Type	

Resource Setup and Maintenance

To access the maintenance portals for the following objects, select **Main Menu**>**Resource Management>**[*object name*].

Seq	Object	Description	Prerequisite	Associated with / Referenced By
L3	Mobile Worker	An individual (human resource) performing work as part of a crew	Mobile Worker Type, User (System User ID), Country, Skills, Service Areas	Crew Shift
L3	Vehicle	A non-human resource used by a crew	Vehicle Type, Equipment	Crew Shift
L4	Crew	Uniquely named group of resources (mobile workers and vehicles) scheduled to perform work	Crew Type	Shift Weekly Templates, Crew Shifts, POUs, POU templates
L5	Dispatcher	An individual responsible for monitoring and managing day-to-day field operations	User, Dispatcher Type, Dispatch Area, Key Performance Indicators	Dispatcher Shift
L5	Period of Unavailability	A period of time in which a crew is planned not to perform work, such as while attending a meeting	Country, Time zone, Work Calendar, Location, POU Type. Resources (Crews, Mobile Workers, or Vehicles attending POU), Service Areas	POU Task
L5	Shift Weekly Templates	A cyclical weekly pattern made up of Crew Shift Templates	Crew, Work Calendar, Time zone (system setup), Service Classes, Break Types, Mobile Workers, Vehicles, Service Areas, Location (for logon/logoff), Shift Cost Profile, Shift Type	Crew Shift

Transfer of Goods Setup and Maintenance

To access the maintenance portals for the objects in this section, do one of the following:

- If you are using functional menus, select **Admin Menu>**[*Functional Menu*]**>**[*object name*]

- If you are using alphabetical menus, select Admin Menu>[*object name*]

The [*Functional Menu*] and [*object name*] are provided in the appropriate columns in the following table.

Seq	Object	Functional Menu	Description	Prerequisites	Associated with / Referenced by
L1	Capacity Type Look Up fields		Enable relevant capacity types by populating their override description		

Seq	Object	Functional Menu	Description	Prerequisites	Associated with / Referenced by
L1	Task Type: Depot Task Type	Service Management			Depot
L2	Depot Profile	Resource Management	Establish common settings for site delay, access time windows and service time windows across multiple depots.		Depot
L3	Depot	Resource Management			

Appendix C

Algorithm Entities

This section lists and describes Oracle Real-Time Scheduler specific algorithm entities, also referred to as plug-in spots.

> **Note**: For instructions on creating custom algorithms, see the Framework documentation. To view detailed information about specific algorithms provided with the base system, use the Application Viewer (also described in the Framework documentation). The Application Viewer provides information about the base logic, inputs, and outputs of each algorithm entity.

Algorithm entities are grouped by type:

* *Installation Options*

* *Activity Management*

* *Common Dispatching Functionality*

* *Sending Data to a Crew*

* *Scheduler*

* *Alerts*

* *KPI*

* *Shift and POU Generation*

* *Zone Contents*

* *Mobile Application*

The **Optional/Required** column in the following tables indicates whether or not the plug-in is required for the system to operate. If Required, the system assumes that a plug-in exists and will generate an error if it does not.

Installation Options

Algorithm Entity (System Event)	Optional / Required	Description
Address Information	Required	Formats address information based on standard address constituents and requested format type. The base product supports a single line format as well as a postal format.
Address Geocoding	Required	Determines the geocoordinates of an address. An algorithm of this type is used to convert a standard address into a geocode latitude/longitude pair. The algorithm is exposed as a business service and called whenever an entity associated with an address needs to be geocoded. The minimum quality level for address geocoding is configurable. By default, the base system requires an exact match, which is the maximum quality level. You can control the quality level of geocoding for any entity that has an address by configuring a custom algorithm on the entity that calls this service with a different minimum quality level, based on your business needs.
Remote Script Invocation	Optional	Processes a remote message sent from a mobile application to the main application. Only required if your organization uses SMS messaging and mobile devices
SMS Send	Optional	Used to send an SMS message. Only required if your organization uses SMS messaging and mobile devices.
SMS Receive	Optional	Used to process an incoming SMS message. Only required if your organization uses SMS messaging and mobile devices.

Activity Management

Algorithm Entity (System Event)	Optional / Required	Description
Activity Expiration	Optional	Handles the situation of an expired activity. The algorithm is called periodically by the activity batch monitor process once the activity has been expired letting it either extend the activity, cancel it, raise a to do entry for a user to take action or any other business rule needed by your organization. This type of algorithm is defined on the activity business object.
Determine Service Area	Optional	Determines an activity's service area if not provided by the host system. It may use any detail associated with the activity to do so. For example, the entity may use the activity's postal code. The algorithm is called when a new activity is created as well as when a proposed activity is sent in on an appointment booking request. This type of algorithm is defined on the activity business object. If you are using the base algorithm you need to also set up the "Postal Code to Service Area Mapping" extendable lookup to associate your postal codes with a corresponding service area.
Update Time Windows	Required	Responsible for calculating an activity's arrival and service time windows. This algorithm is called when an update is made to an activity or an appointment booking request is received. This type of algorithm is defined on the activity business object.

Common Dispatching Functionality

Algorithm Entity (System Event)	Optional / Required	Description
Allocate to Shift	Required	Performs business rules necessary to allocate an activity to a particular crew's shift. This type of algorithm is defined on the activity business object
Cancel Activity	Required	Performs business rules necessary to cancel an activity. This type of algorithm is defined on the activity business object.
Defer Activity	Required	Performs business rules necessary to defer scheduling of an activity to a specific date. This type of algorithm is defined on the activity business object.
Lock Depot Run	Required (Depot only)	Locks a depot related activity to its currently assigned depot task. The latter represents the scheduled run to the depot to pickup or drop off goods. This type of algorithm is defined on the depot task business object.
Unassign	Required	Unassigns an activity from its currently assigned crew if any and set to be rescheduled. If it is currently allocated to a specific crew, this also de-allocates the activity from that crew, making it eligible for all qualifying crews. This type of algorithm is defined on the activity business object.

Sending Data to a Crew

Algorithm Entity (System Event)	Optional / Required	Description
Set Alternate ID	Required	Responsible for assigning a unique alternate activity ID that is relatively easier to handle and shorter in length than the system-generated prime key. This is mainly used by dispatchers to radio-dispatch an activity to a crew when automatic synchronization with their mobile device is not possible. This type of algorithm is defined on the activity business object.
Shift Synchronization	Required	Responsible for sending current shift details to the crew. This is called once when the crew requests the shift at logon time and on certain changes after that. This type of algorithm is defined on the crew shift business object.
Task Synchronization	Required	Used to send current task details to the assigned crew. This is called when the task is first dispatched and whenever dispatched information is changed. The algorithm is also called to prepare the preview information for a task when it is previewed before the shift starts. It may also be called to resend a task to the crew if the original mobile device breaks. This type of algorithm is defined on the task business object.

Scheduler

Algorithm Entity (System Event)	Optional / Required	Description
ABG Scheduler Read	Required	Responsible for obtaining appointment booking group (ABG) details relevant to the scheduler. The schedule manager invokes this algorithm whenever a new or changed appointment booking group needs to be communicated to the scheduler. This type of algorithm is defined on the ABG business object.

Algorithm Entity (System Event)	Optional / Required	Description
Depot Scheduler Read	Required (Depot only)	Responsible for obtaining scheduling details of a depot associated with any of the scheduler's registered tasks and shifts. The algorithm is called when the depot is first time interfaced to the scheduler and upon subsequent changes made to its information. A depot is sent along with its access and service time windows for a requested time range. Time window calculation takes into account holidays as well as explicit overrides canceling or extending normal hours of operation. This type of algorithm is defined on the depot business object.
Scheduler Configuration Read	Required	The schedule manager invokes this type of algorithm to communicate new or changed scheduler configuration details to the scheduler. This type of algorithm is defined on the scheduler configuration business object.
Shift Scheduler Read	Required	Responsible for obtaining shift details relevant to the scheduler. The schedule manager invokes this algorithm whenever a new or changed shift needs to be communicated to the scheduler. This type of algorithm is defined on the crew shift business object.
Task Post Dispatch	Required	Responsible for executing the necessary business rules and processing associated with a schedule update of a dispatched task. The schedule manager calls it when a task reaches the dispatched scheduler state and on every schedule update to it after that. This type of algorithm is defined on all task business objects except for assignment business object.
Task Scheduler Read	Required	Responsible for obtaining task details relevant to the scheduler. The schedule manager invokes this algorithm whenever a new or changed task needs to be communicated to the scheduler. This type of algorithm is defined on all task business objects except for assignment business object.

Alerts

Algorithm Entity (System Event)	Optional / Required	Description
Create Alert	Optional	Responsible for raising new alerts of a given type. This algorithm is defined on the alert type record and is called by a monitor algorithm on the alert type business object.
Evaluate Alert	Optional	Responsible for evaluating an existing alert to determine whether it can be closed. This algorithm is defined on the alert type record and is called by an alert business object monitor algorithm executed when the alert monitor background process runs or when updates are made to the activity/crew shift associated to an alert.

KPI

Algorithm Entity (System Event)	Optional / Required	Description
KPI Calculation	Required	Calculates details related to a KPI for a given dispatcher based on a requested mode of operation. By default, this algorithm calculates the KPI measurement value and formats its summary description for display. It may also be called to provide detailed, actionable lists of the actual activities and shifts contributing to the measurement. This algorithm is defined on the KPI record.

Shift and POU Generation

Algorithm Entity (System Event)	Optional / Required	Description
Generate Crew Shifts	Required	Generates shifts for a crew based on a template. This type of algorithm is defined on the shift template business object.
Generate POUs	Required	Generates periods of unavailability for based on a template. This type of algorithm is defined on the POU template business object.
Prepare Shift Data	Optional	Prepares and adjusts shift details to be used for creating shifts based on details configured on the crew record. The algorithm is defined on the crew business object and is called from the shift generation process to further apply crew details on top of a shift template before it is used to create shifts. It is also called when an ad-hoc shift is added online.

Zone Contents

Algorithm Entity (System Event)	Optional / Required	Description
CDI Gantt Data	Required	Prepares the data to be displayed on the common dispatcher interface Gantt zone. This algorithm is referenced on the Scheduling Gantt zone as a zone parameter.
Calendar	Required	Supports a special kind of zone that displays a calendar populated with objects for display. An algorithm of this type prepares the data to be displayed on an entity's calendar zone. This algorithm is referenced on the Calendar zone as a zone parameter.
Crew Shift Detailed Information	Required	Prepares various formats of detailed shift information, based on where the data is to be displayed. For example, it formats a shift's detailed information as displayed at the top of the context menu when a user right-clicks on a shift in the Scheduling Gantt zone. It also formats the data to be displayed when a users hovers over a shift-related point on the CDI Map. This algorithm is referenced on the shift's business object.
Task Detailed Information	Required	Prepares various formats of detailed task information, based on where the data is to be displayed. For example, it formats a task's detailed information as displayed at the top of the context menu when a user right-clicks on a task in the Scheduling Gantt zone. It also formats the data to be displayed when a users hovers over a task-related point on the CDI Map. This algorithm is referenced on the task's business object.

Mobile Application

The following algorithms are only applicable to the mobile application.

Note: These MCP Business Object algorithms are modeled after the server concept of the corresponding algorithms, but apply only to the mobile device and have a different hard parameter interface. Unlike the server framework, when these MCP algorithms are called the entity in the local database still reflects the unchanged version, while the changed entity is provided in memory as an input hard parameter. Further changes to the entity should be made to the input copy, not directly to the database. Once all algorithms are processed, the mobile framework persists the changed version to the database.

Algorithm Entity (System Event)	Optional / Required	Description
Capture Content	Optional	Implements a specific type of digital content capture on a mobile device for a given entity, for example capture sound or picture related to an activity. The algorithm posts the captured file on a designated file location on the device. When synchronized to the server the file is stored as an attachment to the related entity. A business service is provided to invoke such algorithms as needed by the mobile application. The MDT type has to be configured with such an algorithm as a capability.

Algorithm Entity (System Event)	Optional / Required	Description
General Event	Optional	An algorithm of this type could be used to implement a generic mobile device capability. It accepts a data area in raw format as well as the capability type in context. A business service is provided to invoke such algorithms as needed by the mobile application. The MDT type has to be configured with such an algorithm as a capability.
Map Layers	Optional	Determines the applicable list of map viewer layers to be displayed when the geographic map is launched on the mobile application for a specific shift. This type of algorithm is configured on the crew shift business object. . If you are using the base algorithm you need to set up the "Mobile Device Map Viewer Themes" extendable lookup to list your map viewer theme layers.
Mapping	Optional	Launches a geographic map on a mobile device. The algorithm is called when the user clicks on the map icon. The icon is available for devices that were configured to support mapping capability. The MDT type has to be configured with such an algorithm as a capability.
MCP Post Processing	Optional	Used to perform additional business logic after a business object instance has been processed on the mobile application.
MCP Enter	Optional	Applies business rules when a business object instance enters a given state on the mobile application.

Appendix D

Batch Controls

This chapter describes the base package batch processes. Use the Batch Control portal and Application Viewer for more details about these batch processes.

Monitors

The main batch controls that might require configuration are the monitors. Monitors determine when an event has taken place within the system and they trigger another event in response. These may or may not require modification to the interval, or frequency of doing a status check on events and launching triggers.

As a general rule, all monitors are executed the first time an entity enters a given state. After that event, the system relies upon the batch process configuration to determine when the process is called again.

Non-Timed Monitors

Monitor type batch controls can be configured with no defined timer interval so that they can be run whenever required. Typically, organizations that use a third party software to manage monitoring and batch triggers would configure monitors in this way.

Timed Monitors

Monitor type batch controls that do not have third party software to trigger the processes can use the simple processing in base package batch controls to launch the batches at whatever time interval is set as the timer interval.

Other Batch Controls

The remaining batch controls include several that control extract and load functions for the integration with Business Intelligence. Two additional batch controls manage additional functions. These are described below.

Base Package Monitor Batch Controls

This section lists the monitor batch controls that are provided in the base package. You are free to configure these batch controls in any way that suits your business practices, however the table provides a suggested timer interval for each type of batch control.

Batch	Description	Recommended Timer Interval
M1-ALMTR	Alert Monitor	60 sec - This monitor should run very frequently to evaluate existing alerts and recycle them in a timely manner.

Batch	Description	Recommended Timer Interval
M1-ATMTR	Alert Type Monitor	120 sec - This monitor should run very frequently so it can raise alerts as soon as they occur and bring them to a user's attention.
M1-CRSHF	Crew Shift Monitor	5 - 10 minutes. Base shift monitor rules are typically used to send estimated arrival time updates to crews for their schedule, so they should be frequent, but it is not necessary for them to be by the minute.
M1-DEPOT	Depot Monitor	Once a day.
M1-DPLOY	Create Deployment	Should only be run as needed, when a deployment items has changed and a new deployment version needs to be created.
M1-DPTTW	Depot Time Window Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
M1-DSMTR	Dispatcher Shift Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
M1-MAIL	Mail Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
M1-POU	Period of Unavailability Monitor	Once a day to generate POUs based on template.
M1-RESRC	Resource Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
M1-RMMSG	Remote Message Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
M1-RSIBP	Remote Script Invocation Manager	Must run constantly.
M1-SCHCF	Scheduler Configuration Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
M1-SCHED	Scheduler Monitor	Every few hours. This can be set at a low frequency since base logic mainly only handles the cleanup and creation of to do errors for persistent scheduling issues.
M1-SHWKT	Shift Weekly Template Monitor	Once a day to generate shifts.
M1-TSKTR	Task Monitor	Every few hours or once a day. This can be set at a low frequency since base mainly handles activity expiration and preallocation. The necessity for running this process depends upon how often an implementation needs these business rules to be performed.
M1-TTWTR	Override Time Window Monitor	There are no base monitor rules for this entity so unless you added custom rules it should not be run.
F1-SYNRQ	Sync Request Monitor Process	Every 5 - 10 minutes. This is a FW owned batch process but MWM uses this entity to send messages to the host system so it needs to run as often as these messages need to be sent.

Other Batch Controls

These additional batch controls may require configuration depending upon your business practices.

Batch	Description	Recommended Timer Interval
M1-DPLOY	Create Deployment	This batch process creates a new deployment for a deployment type and language. It should be run as needed to reflect changes in the mobile application configuration.
M1-RSIBP	Remote Script Invocation Manager	An outbound Remote Script Invocation (RSI) request may be associated with a script to call back if it takes too long for a message to be delivered at the device. This batch process constantly monitors the outbound RSI queue on the server for callback requests. An incoming RSI request sent from a device is typically processed right away. However, there may be times where this is not possible and the request remains pending in the queue. This process also monitors the RSI inbound message queue and executes pending requests if any. The process is designed to run for the duration specified on the processing time parameter. If your implementation does not use a 3rd party batch scheduler and relies on the base product batch scheduling functionality to resubmit the process on a timely basis set the batch control type to "Timed" along with the appropriate timing details. Callback processing is multi-threaded whereas processing pending incoming request is single threaded and performed by the first thread only. Commit frequency may not be overridden.

Appendix E

Mobile Communication Platform Custom Business Services Development Setup Guide

This document describes how customers can create their own custom business services using the MCP Business Service API. This includes information on development environment setup, API usage, implementation guidelines, deployment, and unit testing.

This section includes the following:

- *Set Up the Eclipse Environment*
- *Write the Java Business Services*
- *Compile and Building the Custom Java Business Services*
- *Test Inside MDT Runtime*
- *JUnit Test the Java Business Services*
- *Java Business Service API Reference*
- *Frequently Asked Questions*

Set Up the Eclipse Environment

To set up the eclipse environment you must create a java project in a new or existing eclipse workspace.

Pre-requisites

- Eclipse 3.5+ IDE for Java Developers
- JDK 1.6.20
- spl-mcp-BS-2.1.0.jar (Delivered separately along with MDT Runtimes)

Creating a Java Project (in a new or existing Eclipse workspace)

1. Create a New Java Project using the File -> New Project Wizard.

2. Type in a project name, set the execution environment JRE to JavaSE-1.6 and click Next.
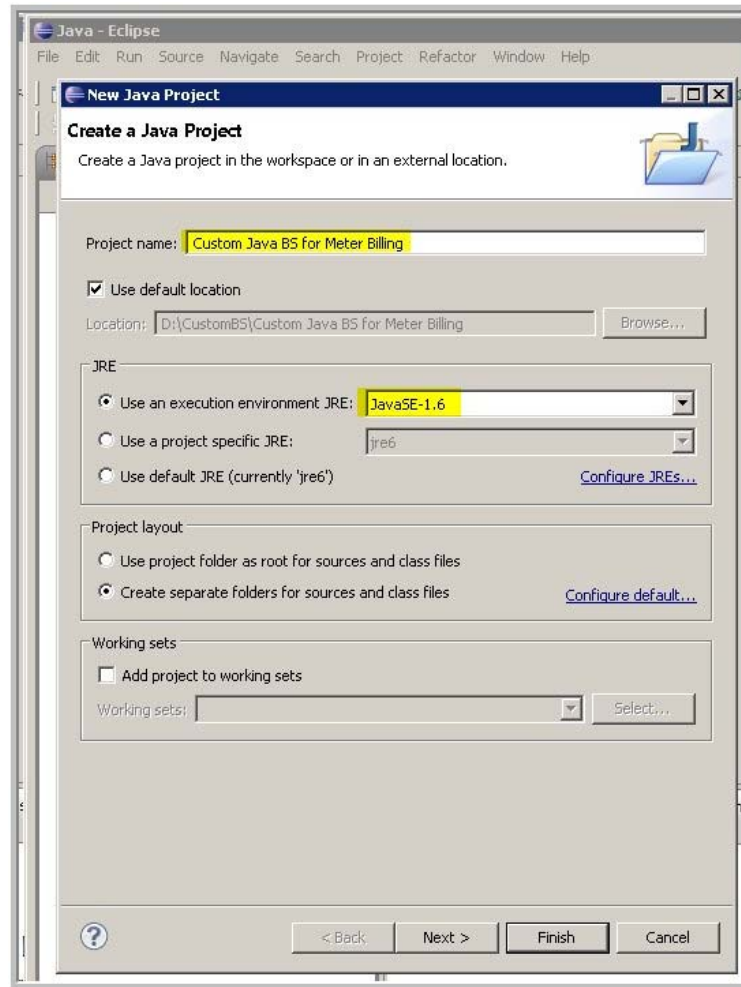
3. Click the **Add External Jars** button and select the Libraries tab.

4. Add the **spl-mcp-BS-2.1.0.jar** as an external JAR.
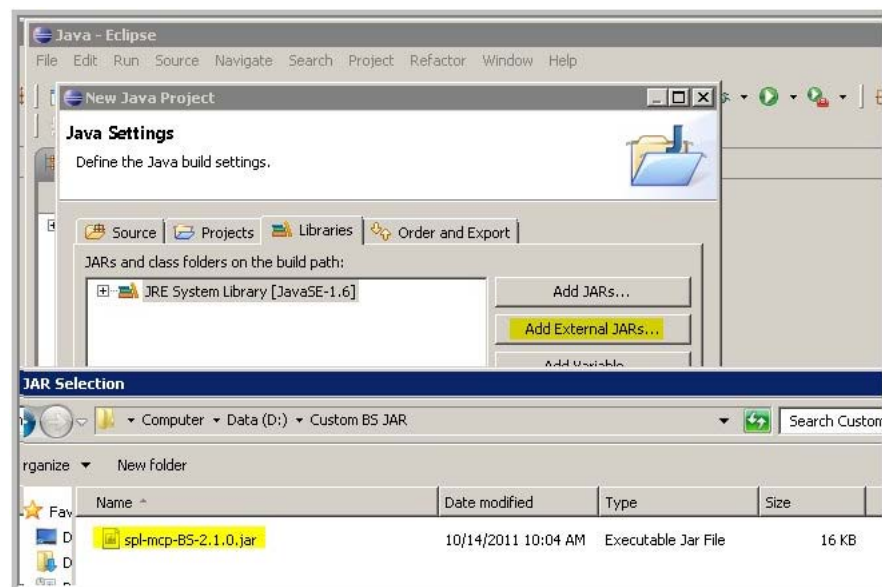
5. After the Import, Click "Finish" to create the Java Project.

The Java project is now successfully created with the imported Java BS API JAR file.
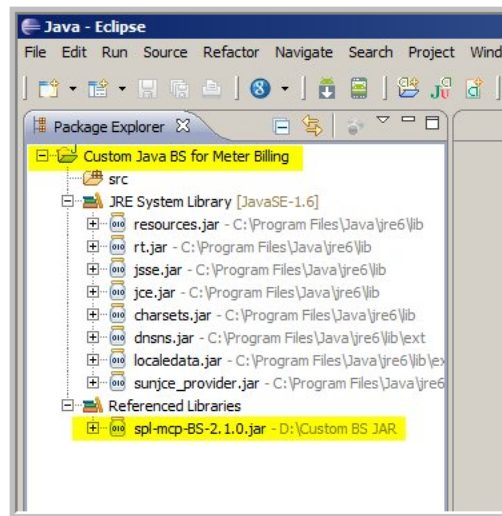
6.   Right Click -> Project properties -> Java Compiler to change the Java project's code compliance to JDK 1.4 (to ensure same code that's written in this project can work on Windows Mobile/Laptop and Android).
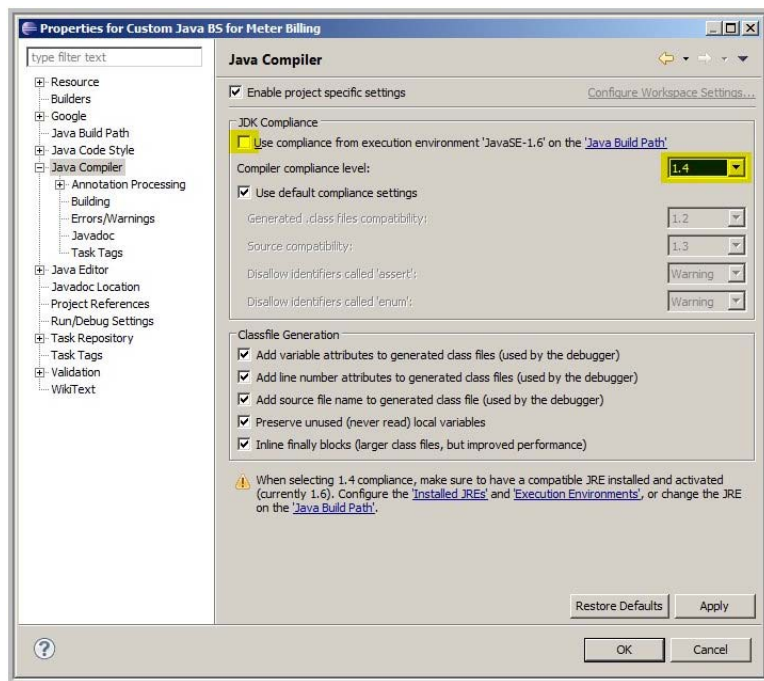
The Java project is successfully setup.

Write the Java Business Services

To write the java business services you must do the following:

- Code the Business Service Implementation
- Create Business Service Metadata

Coding the Business Service Implementation

Creating a new Java Class file to write the Java Business Service.
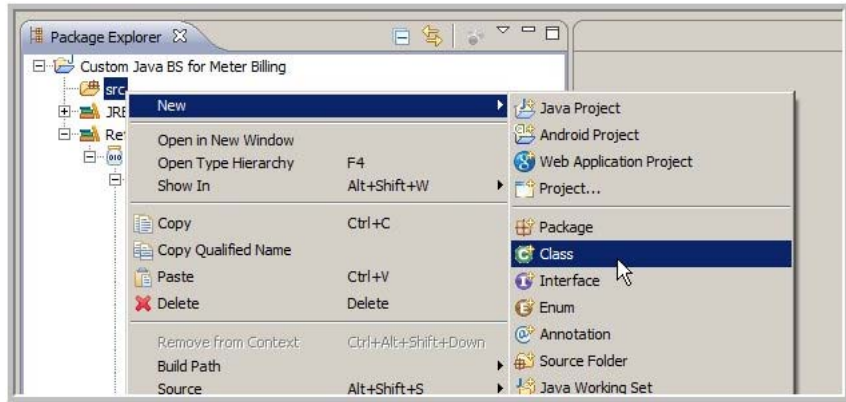
1. Enter the package name, the class name, and **IMCPBusinessService** as an interface this class implements and click "Finish".
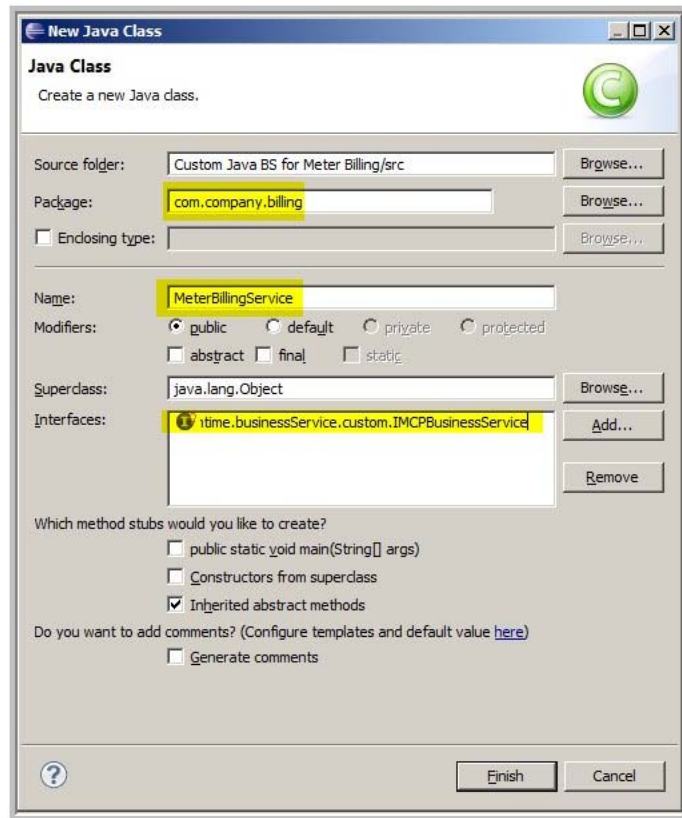
A Java Business class has been created as below.
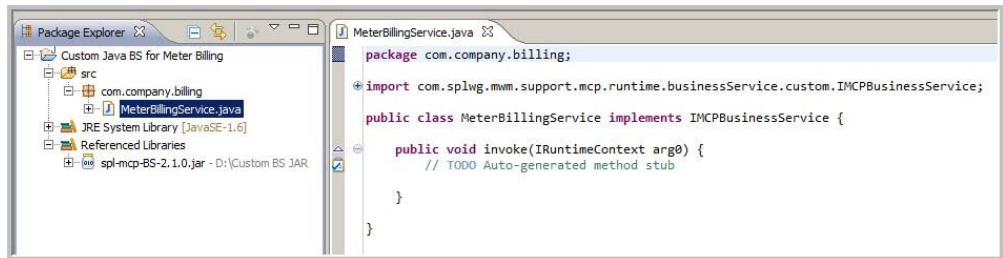
2. Implement the Java Business Service using the Custom Java Business Services APIs.

```
package com.company.billing;
```

```
import
com.splwg.mwm.support.mcp.runtime.businessService.custom.BusinessServiceError;
…… Other imports here.

public class MeterBillingService implements IMCPBusinessService {

        // This will store the RuntimeContext instance for the runtime.
        // This will hold a reference to the XML DOM
        IRuntimeContext mContext = null;

        // Provides access to MDT JBS API methods.
        IMCPCallback mcpCallback = null;

        // Just hard-coding the value for the sake of the example
        private static int COST_PER_UNIT = 4;

        private static Class clazz = MeterBillingService.class;

        /**
         * Mandatory method - is a start point for any Custom Java Business
Service.
         */
        public void invoke(IRuntimeContext context) {
                mContext = context;
                mcpCallback = context.getMCPCallback();

                mcpCallback.logInfo("Reached MeterBillingService ", clazz);

                double billAmount = calculateBill();

                mcpCallback.logInfo("Bill Amount = " + billAmount, clazz);

                mcpCallback.logInfo("End of MeterBillingService ", clazz);

        }

        // Setter to set the Runtime Context (used only in Unit testing -
where a RuntimeContext)
        public void setRuntimeContext(IRuntimeContext pContext) {
                mContext = pContext;
        }

        public String[] getUserInformation() {
                String[] details = mcpCallback.getUserInfomation();
                mcpCallback.logInfo("User name = " + details[0] + " Password
= " + details[1], clazz);
                                        return details;
        }

        public double calculateBill() {

                DataElement currentElement = mContext.getJBSDOM();
                DataElement bsData =
currentElement.getChildElement("bsData");
                double meterReadStart =
bsData.getChildElement("meterReadStart").getDoubleValue();
                double meterReadEnd =
bsData.getChildElement("meterReadEnd").getDoubleValue();
                double billAmount = COST_PER_UNIT * (meterReadEnd -
meterReadStart);
```

```
                    DataElement meterBillAmount =
bsData.getChildElement("meterBillAmount");

                    if(meterBillAmount == null) {
                            if(mcpCallback.isEligibleForDebug()) {
                                    mcpCallback.logDebug("Output Element -
meterBillAmount not previously found, adding it now", clazz);
                            }

bsData.addChildElement("meterBillAmount").setDoubleValue(billAmount);
                    }  else {
                            if(mcpCallback.isEligibleForDebug()) {
                                    mcpCallback.logDebug("Output Element -
meterBillAmount previously created, skipping creation", clazz);
                            }
                            meterBillAmount.setDoubleValue(billAmount);
                    }

                    mcpCallback.logInfo("Reached MeterBillingService ", clazz);
                    return billAmount;
            }
}
```

Creating Business Service Metadata

1. Open the Online ORS/MWM web application.

2. Create a new Business service by navigating to:

   Admin Menu → B → Business Service (+)

3. Enter a business service name and other details.

   Business Service Name/Code – Prefixed by **CM**.

   Example: CM-MeterBillingBusinessService

   Description and Detailed Description – Describe the business service's usage.

   Service Name – For developing Custom Java Business services, always use **M1-MCPCUSTOMBS**.
   This is somewhat different from how the online ORS application business services are written in which the business service implementation class and its service program are plugged-in through this field.

   Application Service – You can use **F1-DFLTAPS** (this is not used on the MDT Runtime).

4. Click on the "Schema" TAB to provide a Business service schema. Through this schema you will map this business service metadata to the Business service Java Implementation. There are two elements in the schema which are important and should always be included.

   bsData – Any input/output for this business service is passed inside this.
   An XPath example – "ZZ-MeterBillingBusinessService/bsData/meterReadStart"

   bsClassName – This mentions the Java class name (without the .java or the .class extension). The class-name should be mentioned in the **"default"** attribute and is mandatory. In this example you can see that the Java class created in the coding phase is used here.
   "**com.company.billing.MeterBillingService"** is the Business Service Java Class Name.

```
<schema>
<bsData type="raw" mapXML="RAW_XML'/>
<bsClassName default="com.company.billing.MeterBillingService"
required="true" mapField="JAVA_CLASS_NAME"/>
```

```
</schema>
```

Example Usage of the Business Service

1. Create a new UI Map which uses the business service.
    Admin Menu → U → UI Map (+)

    Enter the UI map details

    •   UI Map Name/Code – usually starts with CM – Example: CM-MeterReadUIMap

    •   Description and Detailed Description – Describes the usage of the UI map.
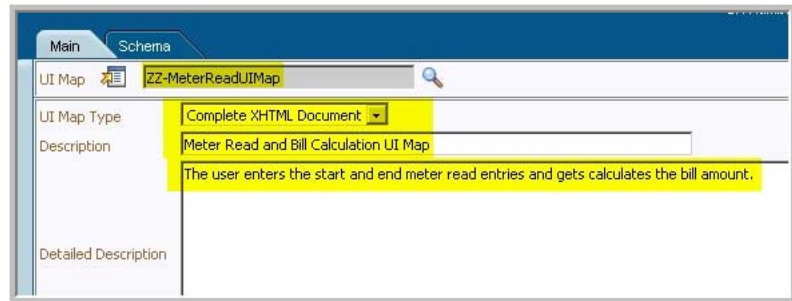
2. Enter the UI Map schema.
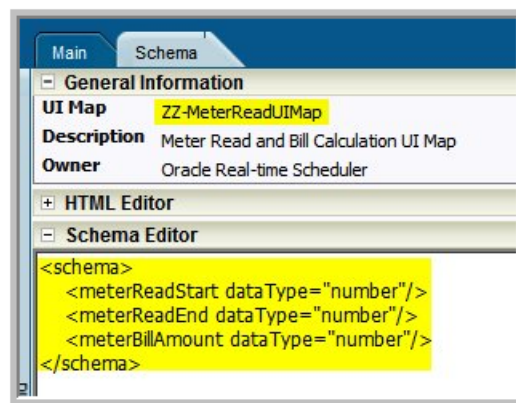
3. Generate HTML for an Input Map using the dashboard zone by clicking the Input Map button.

    You can now see that the HTML is generated in the "HTML Editor" section. Click "Save" again.

    The generated HTML has to be modified to make it XHTML complaint by closing all unclosed HTML tags.

Example:

Original<link href="cisDisabled.css" type="text/css" rel="stylesheet">

New<link href="cisDisabled.css" type="text/css" rel="stylesheet"/>
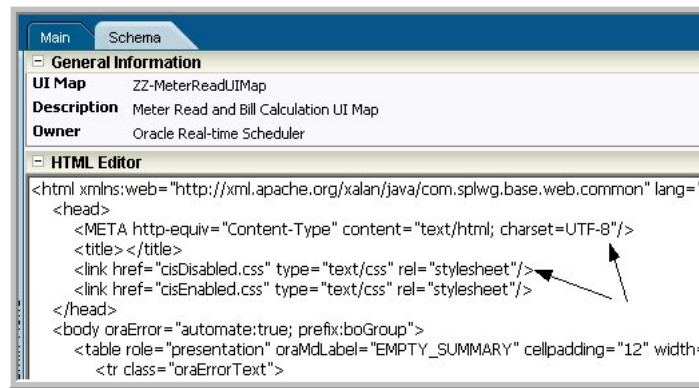
4. Modify the HTML buttons to call the Custom Business Services.

```
<td class="oraSectionStart oraEmbeddedTable" colspan="2">
  <table role="presentation" oraMdLabel="EMPTY_SUMMARY" cellspacing="2">
    <tr>
      <td><input onClick="calculateMeterRead();" value="Calculate Meter Bill Amount" class="oraButton" type="button"/>
    </tr>
  </table>
</td>
```

5. Write Java Script method to calculate Meter Read (which calls a service script).

   Note: oraPrepareToSave() suppresses the "unsaved data changes warning".
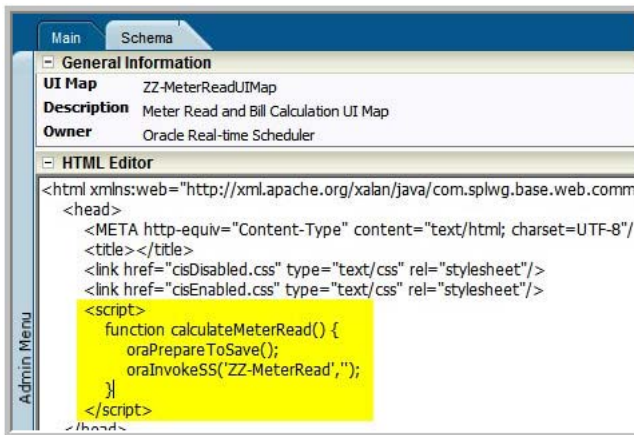
6. Create a service script that the UI map calls for Meter Read calculation.
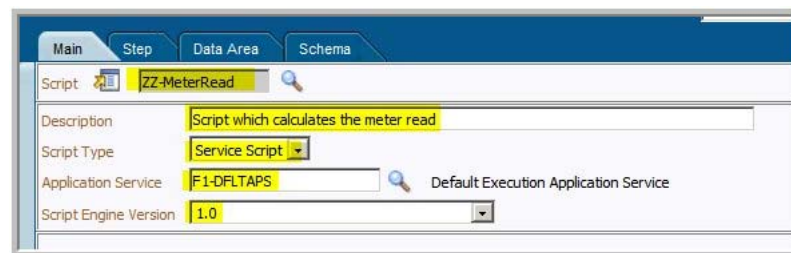
Since the service script calls the BS, include the BS schema.

Service script has the same fields as the UI map, so Include Map is used here. For this reason, it is not necessary to explicitly include the UI Map in the "Data Area" section above.
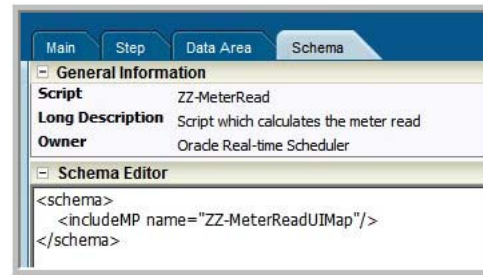
The include helps pass the data between the UI map and the service script and back to the UI Map. At runtime, the service script schema resembles the example below:

7. Add script steps to set the input data to the BS and call the BS and fetch the output to be passed to the UI Map.

    •   Pass the inputs received from the UI Map to the Business service.
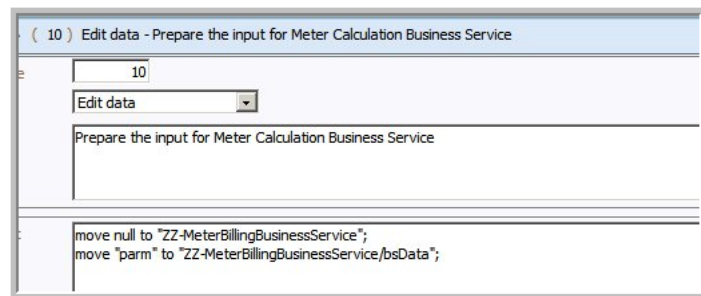
    •   Call the custom meter business service.
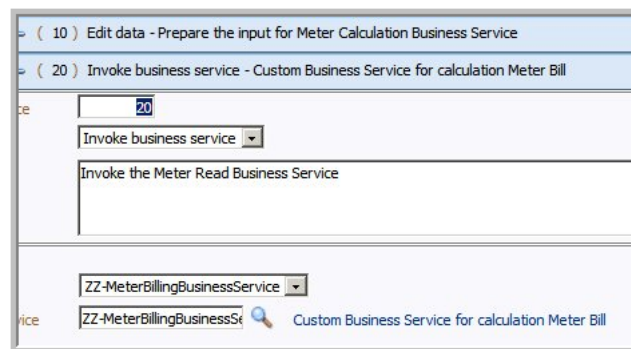
- Get the BS output and set back to script schema.
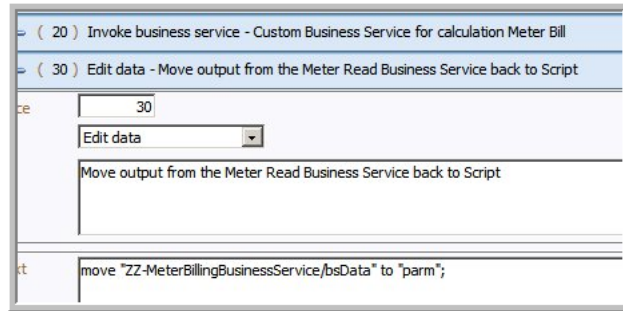
- Terminate with the same UI Map by passing the script schema to the UI map.
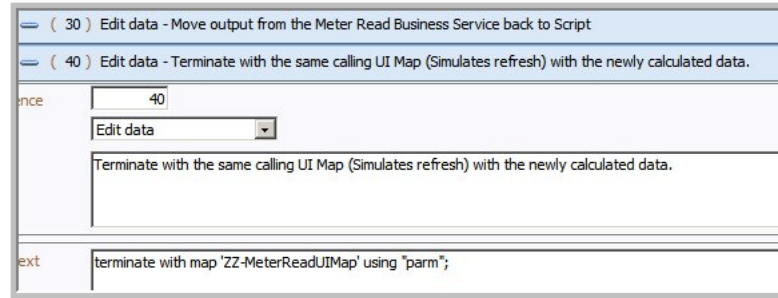
8. Create a new script which is used as the initial service script.
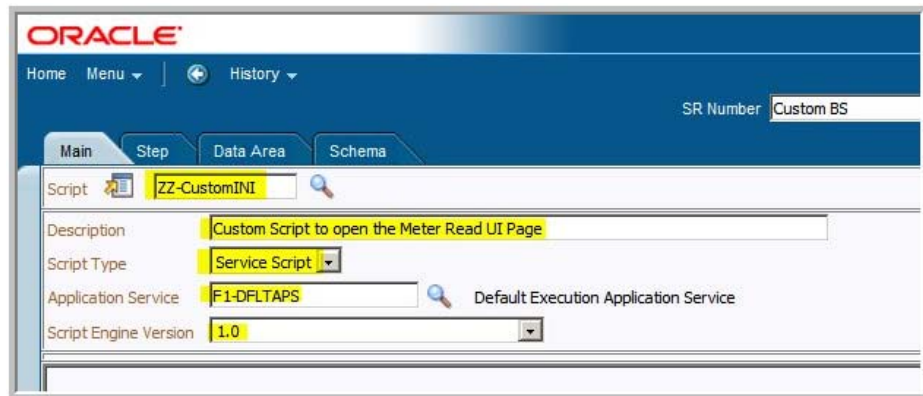
9. Add the UI map created to the script.
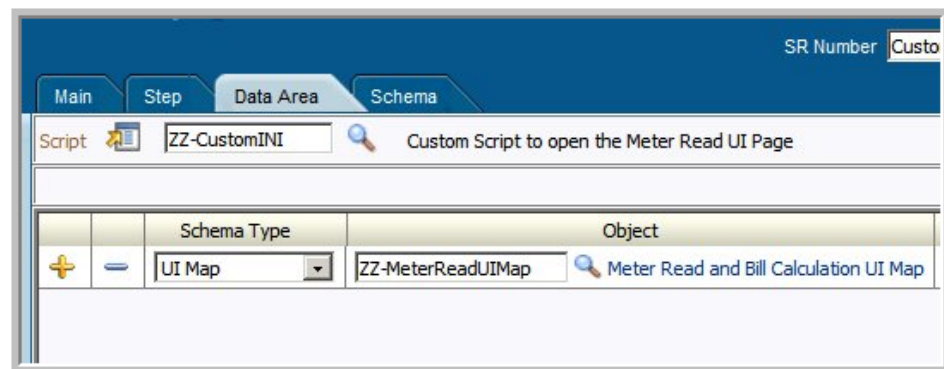
10. Create a script step to terminate with the map.

11. Now that all the metadata for the example is created, prepare this for Deployment.

- Create a Deployment Part to include the items created

- 2 Service scripts (Initial Service Script & Script to call BS)

- 1 Business service – The custom business service that was created

- 1 UI map to record and display the calculation.

12. Create a Deployment Type with this Deployment Part.

13. Run the Batch Job to generate Deployment for this Deployment Type.

14. Activate the deployment after the batch job status is "ended" and there are no errors observed in the Batch Run Tree.

Compile and Building the Custom Java Business Services

To compile and build the custom Java business services you must build the Java code into JAR and APK.

Building the Java code into JAR and APK

MDT runtimes require the Custom Java business services compiled in different formats.

- Windows Mobile MCP

- Windows Laptop MCP – Supports multiple JAR files

- Android MCP - Supports only 1 APK, so classes from multiple JARS have to be merged into a Single APK file.

You can maintain all 3 using ant build scripts in the same project location.
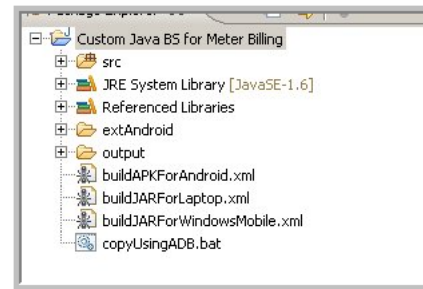
Windows Laptop – Compiling the Business Service code into a JAR File

This simple Ant script compiles the Java code into a JAR File and copies it into the c:\MWMApp\ext. The significance of the "ext" folder is explained in the "Testing the Java Business Services inside MDT Runtime" section.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project basedir="." default="buildjar" name="MCP-BS-Builder">
<description>
Ant build script to build Custom BS implementation's JAR
</description>
    <property name="output.dir" value="bin" />
    <property name="target.dir" value="C:\MWMApp\ext" />
    <property name="jar.name" value="meter-read" />
    <target name="buildjar">
        <delete file="${target.dir}/${jar.name}.jar" />
        <jar destfile="${target.dir}/${jar.name}.jar"
basedir="${output.dir}" />
    </target>
</project>
```

Windows Mobile – Compiling the Business service code into a JAR File

This simple Ant script compiles the Java code into a JAR File and copies it into the device's ext folder.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project basedir="." default="buildjar" name="MCP-BS-Builder">

<description>
Ant build script to build Custom BS implementation's JAR
```

```
                              </description>

            <property name="output.dir" value="binCVM" />
            <property name="target.dir" value="extCVM" />
            <property name="jar.name" value="meter-read" />
<property name="cvm.dir" value="D:\Custom BS JAR\CVM" />

            <target name="clean">
                <delete dir="${output.dir}" />
                <mkdir dir="${output.dir}"/>
            </target>

            <target name="compile">
                <javac srcdir="src"
                  destdir="binCVM"
                  compiler="javac1.4"
                  debug="on"
                  debuglevel="lines,vars,source"
                  source="1.4"
                  target="1.4"
                  executable="C:\Program Files\java\jdk1.6.0_25\bin"
                  failonerror="true"
                  listfiles="true"
                  >
                <bootclasspath path="${cvm.dir}\lib\btclasses.zip" />
                <classpath path="D:\Custom BS JAR\spl-mcp-BS-
2.1.0.jar;${cvm.dir}/lib/charsets.jar;$${cvm.dir}/
libcharsets.jar;${cvm.dir}/lib/jaas.jar;${cvm.dir}/lib/
jce.jar;${cvm.dir}/lib/jsse-cdc.jar;${cvm.dir}/lib/
localedata.jar;${cvm.dir}/lib/foundation_jdbc.jar;${cvm.dir}/lib/
sunrsasign.jar" />
                <extdirs path="${cvm.dir}/lib/ext" />

                </javac>
            </target>

            <target name="deploy">
 <exec executable="c:\Program Files\Windows Mobile Developer Power
Toys\/CECopy/cecopy.exe" failifexecutionfails="true"
failonerror="true">
 <arg value="${target.dir}"/>
<arg value="dev:\MWMApp\ext"/>
</exec>
            </target>

            <target name="buildjar">
             <antcall target="clean" />
             <antcall target="compile" />

                <delete file="${target.dir}/${jar.name}.jar" />
                <jar destfile="${target.dir}/${jar.name}.jar"
basedir="${output.dir}" />

                <antcall target="deploy" />
            </target>
</project>
```

Android Runtime – Supports 1 APK file (JAR converted into APK)

This section describes how to compile and test on Android MDT Runtime.

Prerequisites:

- Eclipse Android SDK (the latest version Rev 15)

- Make sure the Android's platform-tools directory is in the system PATH (or you can point the individual executables in the Antd file).

This simple Ant file creates a APK from the class files and then copies it into the Android Emulator. For using this apk with Android device. Please refer to the section titled " Testing Inside MDT Runtime."

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<project basedir="." default="copyToDevice" name="JAR To Dex
Converter">

<description>
Ant build script to build dex files.
</description>

 <property name="classes.dir" value="bin" />
 <property name="target.dir" value="D:\CustomBS\Custom Java BS for
Meter Billing\extAndroid\" />
 <property name="apk.name" value="meter-read.apk" />

 <target name="buildAPK">
<exec executable="dx.bat">
<arg value="--dex" />
<arg value="--output=${target.dir}/${apk.name}" />
<arg value="--positions=lines" />
<arg path="${classes.dir}" />
</exec>
 </target>

 <target name="copyToDevice" depends="buildAPK">
<exec executable="copyUsingADB.bat">
    <arg value="${apk.name}" />
</exec>
 </target>
</project>
```

Test Inside MDT Runtime

For Laptop and Windows Mobile MDTs, in the above Compiling and Building step we created a JAR file (multiple JAR files are supported too). Android requires an APK file created which was also explained above. (supports 1 apk file only).

If you have used a similar script as shown in the compiling and building step, then you would have noticed that the JAR file (meter-read.jar) is already copied into the MDT Runtime's ext folder.

Setting up Laptop MDT

For Laptop, the default path where the MSI is installed is C:\ORSApp directory (it can be different too based on installation time preference).

In this installation folder you'll find an "**ext**" folder. Copy the meter-read.jar into the "ext" folder.

Setting up Windows Mobile

For Windows mobile, the default path where the CAB is installed is \ORSApp on the device.

In this installation folder you'll find an "**ext**" folder. Copy the meter-read.jar into the "ext" folder.

Setting up Android

For Android, the folder where the APK has to be copied is different from Laptop and Windows Mobile.

Android Emulator – The provided ant build file copies the APK file into Android's directory – "**/data/data/com.splwg.base.android/ext**". This folder is writable and is accessible in the case of emulator only. If you wish to copy the file manually, you can use these commands.

```
C:\>adb shell
$cd /data/data/com.splwg.base.android/
$mkdir ext
$exit
C:\>adb push meter-read.apk /data/data/com.splwg.base.android/ext/
```

Android Device – The provided ant build script can compile the APK file and it places it in extAndroid folder of the current Eclipse Project. To import the APK file into the folder "/data/data/com.splwg.base.android/ext", you need to follow these steps

1. Connect the Android device to the computer and go into USB Disk drive mode

2. Copy the meter-read.APK file into a new the sdcard into a new folder structure ORSApp\ext\

3. Disconnect the device from the computer (or move into Charge Only mode)

4. Use Oracle ORS Tools shortcut icon to "Import" and the APK is copied across into the required folder /data/data/com.splwg.base.android/ext.

Using MDT Runtime after the Initial Setup

After the initial setup steps as explained above, use the MDT Runtime as follows:

- For all the MDT Runtimes, the application is launched and the new Deployment created in the step Creating Business Service Metadata is downloaded.

- After the download, the usage of this business service is transparent and will work as coded into the UI Maps like any other Business service called in the system.

- Entering "meterReadStart" and "meterReadEnd" and clicking the "Calculate Meter Bill Amount" button will calculate the "meterBillAmount" as 40.0 (20 – 10 * 4). This calculation was done using the Meter Read Custom Java Business.

- The test can be repeated multiple times by modifying the start and end values.

Laptop

Windows Mobile
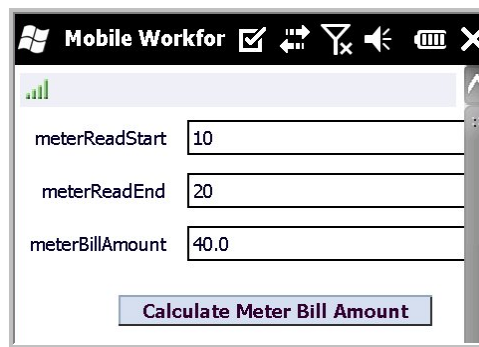
Android
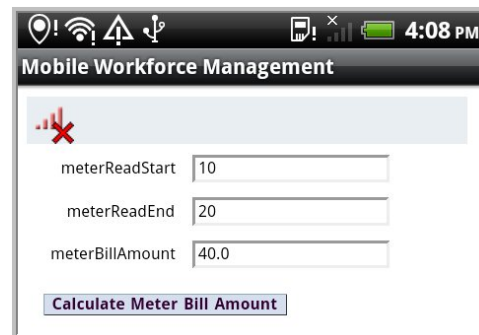
JUnit Test the Java Business Services

To complete JUnit testing of the Java business services you must:

- Create the JUnit Project
- Write a JUnit Test Case

Creating the JUnit Project

This section provides steps for creating a new Java Project for writing Junit tests.
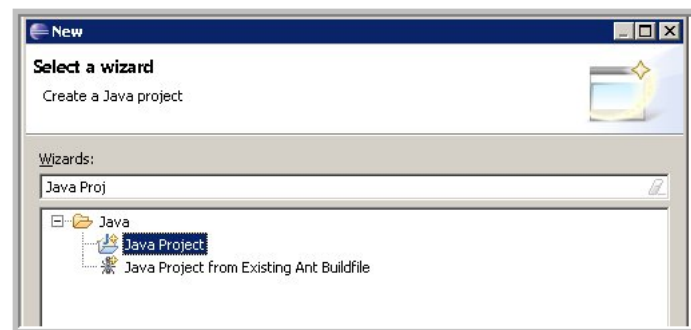
1. Enter a project name.

(For example: Tests for __original project name __)
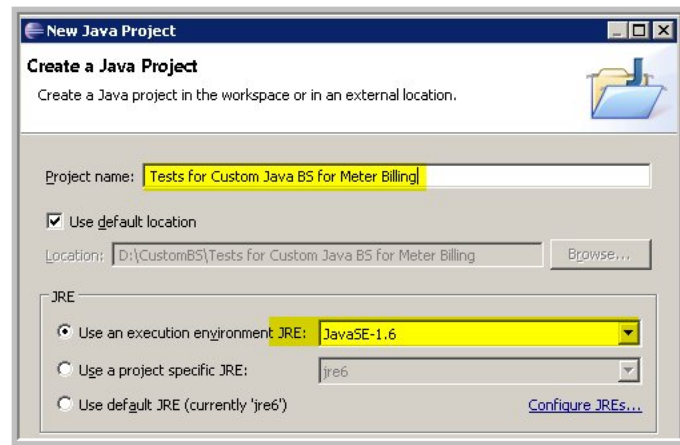
2. Add the original project as a dependency for the new Test project.
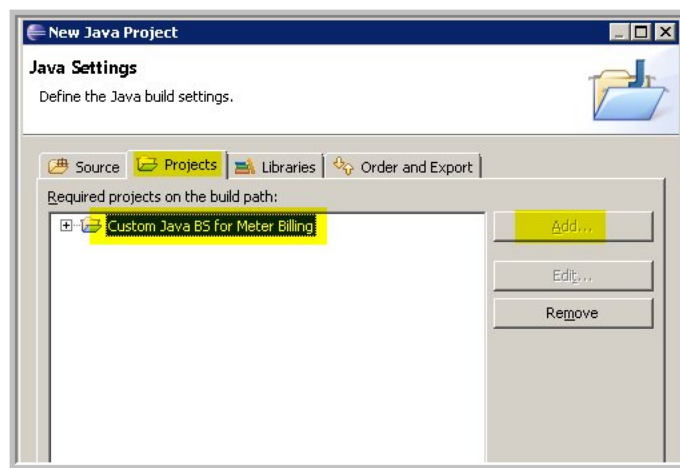
3. Add the JUnit3 Library to the current project from the "Libraries" tab.
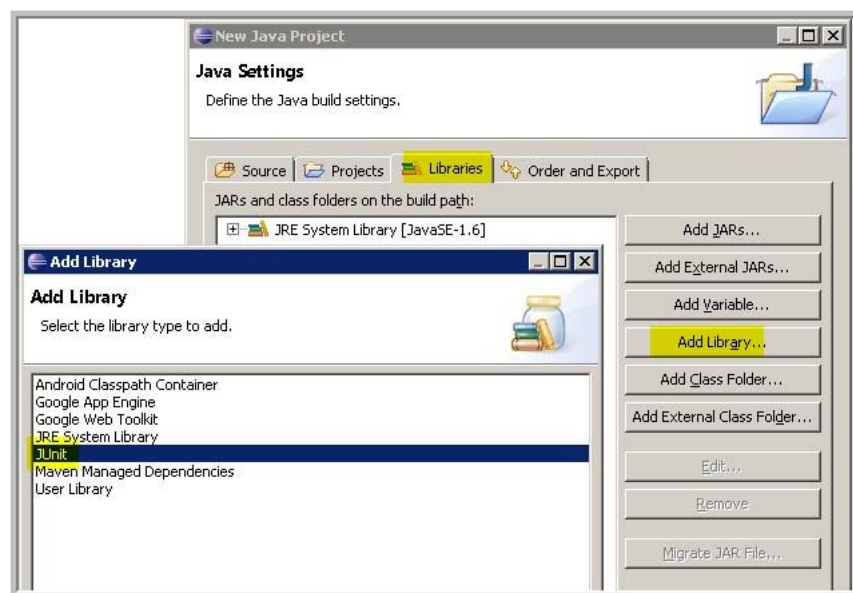
4. After adding the Junit3 Library, click "Finish" and the project is ready.
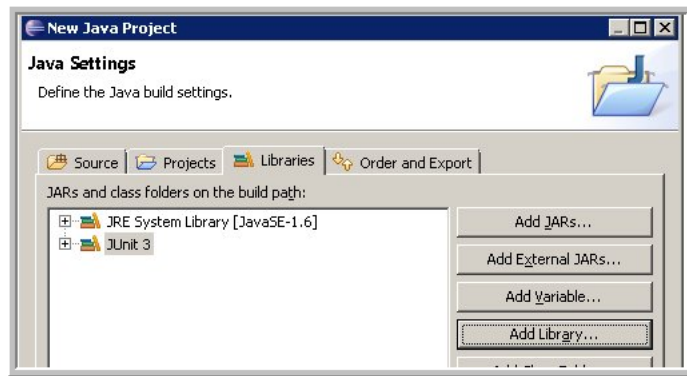
5. Add the dependency JARS to the JUnit project.

    MDT Java BS API - spl-mcp-bs-2.1.0.jar – As used to create the Java business service.

    DOM4J - dom4j-1.6.1.jar – Can be downloaded from DOM4j's official download page

    http://sourceforge.net/projects/dom4j/files/dom4j/
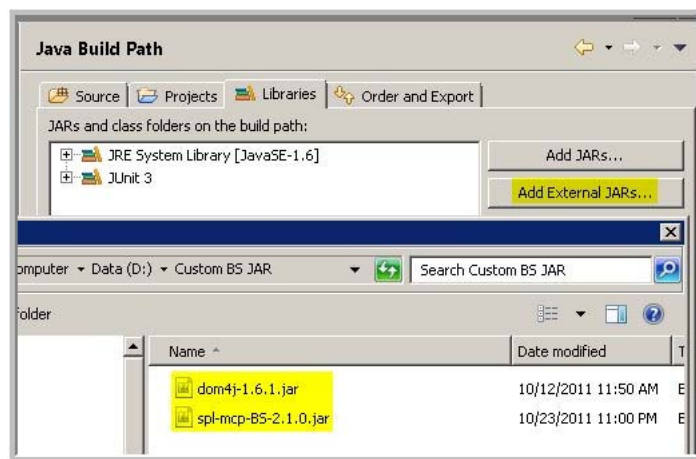
The JUnit test project is now ready to write JUnit test cases.
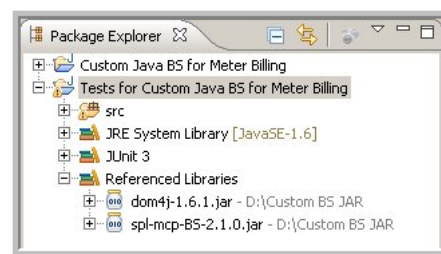
Writing the JUnit Test Case

1. Create a new Java Class with the same package as the Original Business service class.

The new Java class is a JUnit java class, so it should extend JUnit's TestCase class.
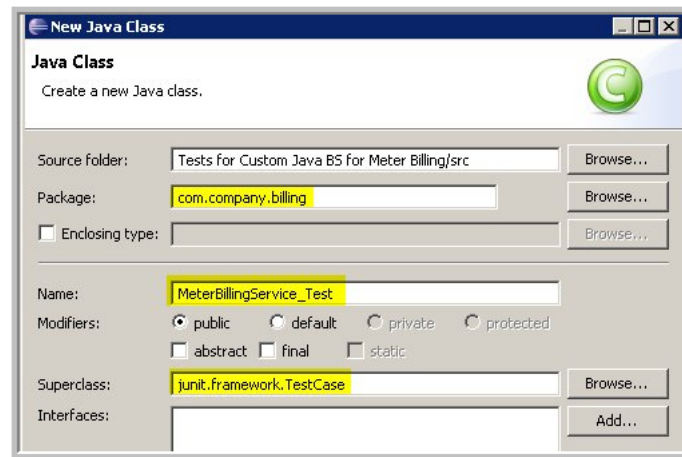
2. Write a MCPCallbackMock Custom implementation class like below.

This mock class can over-ride any methods explained in the IMCPCallback section. For example, here the implementation over-rides some of the logger methods to print to System OUT and return fake user credentials to the test case.

```java
package com.company.billing;
import
com.splwg.mwm.support.mcp.runtime.businessService.custom.MCPCallbackMock;
public class MeterReadMCPCallbackMock extends MCPCallbackMock {

    @Override
    public String[] getUserInfomation() {
        return new String[] {"FAKEUSER", "FAKEPASSWORD"};
    }

    @Override
    public boolean isEligibleForDebug() {
        return true;
    }

    @Override
    public void logDebug(String message, Class classReference) {
        System.out.println(classReference.getName() + " DEBUG: "
+ message);
    }

    @Override
    public void logInfo(String message, Class classReference) {
        System.out.println(classReference.getName() + " INFO: " +
message);
    }
}
```

3. Create the actual test case which calls the Meter Bill amount business service in the JUnit test environment.

The test case uses the new MeterReadMCPCallback mock class created above. If the business service does not call MCP APIs (i.e does not use MCPCallback methods), then one can use the default provided: MCPCallbackMock class instead.

The test case has 2 test methods which test the public methods of the business service. The test case input is a "sample" input as how the business service Java class receives the input in MDT Runtime and then calls the calculateBill() method. The output is read through the Business service's output XML and is validated.

This is how the Java Business Service can be tested outside the MCP Runtime.

```java
package com.company.billing;

// Imports go here.

public class MeterBillingService_Test extends TestCase {

public void testMeterBillCalculation() {

MeterBillingService meterBillingService = new MeterBillingService();
RuntimeContextMock context = new RuntimeContextMock();
MeterReadMCPCallbackMock mcpcallback = new MeterReadMCPCallbackMock();
context.setMCPCallback(mcpcallback);
context.setJBSDOM("<root>" +
"<bsData>" +
"<meterReadStart>10</meterReadStart>" +
"<meterReadEnd>20</meterReadEnd>" +
"</bsData>" +
"</root>");
meterBillingService.setRuntimeContext(context);
double billOutput = meterBillingService.calculateBill();
assertTrue("Bill Amount returned by API = " + billOutput, billOutput ==
40);

        DataElement out = context.getJBSDOM();
    DataElement meterBillAmount =
out.getChildElementForXPATH("bsData/meterBillAmount");
double bsOutput = meterBillAmount.getDoubleValue();
assertTrue("Bill Amount returned by BS Output = " + bsOutput, bsOutput
== 40);

}

public void testUserDetails() {

MeterBillingService meterBillingService = new MeterBillingService();
RuntimeContextMock context = new RuntimeContextMock();
MeterReadMCPCallbackMock mcpcallback = new MeterReadMCPCallbackMock();
context.setMCPCallback(mcpcallback);
context.setJBSDOM("<root>" +
"</root>");
meterBillingService.setRuntimeContext(context);

String[] userInfo = meterBillingService.getUserInformation();
assertTrue("Required User found ", "FAKEUSER".equals(userInfo[0]));
assertTrue("Required Password found ",
"FAKEPASSWORD".equals(userInfo[1]));

}
}
```

Java Business Service API Reference

This section explains all the Classes and APIs provided as a separate MDT Java Business JAR (spl-mcp-BS-2.1.0.jar) which help in implementation of MDT Custom Business Services. The custom business service implementations shouldn't refer to any other classes in MCP Runtime except for these classes.

The development environment is the same for developing for all the MDT Runtimes i.e WM/Laptop and Android too. In the case of Android, the custom BS class files will be converted into an APK file.

IMCPBusinessService (Custom BS Classes should implement this)

This interface contains one method with the following signature

void invoke(RuntimeContext context)

The custom business services will provide an implementation for this method by using the various other Custom Business Service APIs explained below to operate on the DOM.

BusinessServiceError (Exception)

This exception will be created and thrown from within the custom BS error handling code. The runtime implementation of this exception uses the existing BSException: (**com.splwg.mwm.support.mcp.runtime.businessService**)

There are ways for the user to show an Error to the User :

- BusinessServiceError (String message)

   Uses string input as message that is shown to the use

- BusinessServiceError (Integer msgCategory, Integer msgNumber, String[] params)

   Uses MDT-FormatMessage BS (internal methods) to form a Language specific message.

IRuntimeContext (RuntimeContext Interface)

This interface abstracts out the actual calls to RuntimeContext. An instance is set before the BS call is made. If the BS is executing in a junit environment, then an instance of custom RuntimeContextMock instance is set which contains the appropriate test DOM (DataElement)

RuntimeContext (Input/Output to the Custom BS)

This is passed as an input to the Custom Business Service:

Member Variables

- **DataElement** (DOM structure) - Passed as an input to the Custom BS

- **MCPCallBack** (Utility API) - Helps to interact with MCP Runtime (ex: executeSS, logging)

Methods

- **public DataElement getJBSDOM()** – Returns the DataElement variable.

- **public MCPCallBack getMCPCallBack()** – Returns the MCPCallBack variable.

RuntimeContextMock (Mock Implementation)

This is the default implementation of the RuntimeContext interface. Customers may extend this class to override the default implementation if necessary. This class is only used in custom BS development and testing. It is not used by the MCP Runtime.

Methods

- **setJBSDOM(DataElement)** – Sets the data element to given Data Element.
- **setJBSDOM(String)** – Parses and sets the data element to the given XML.
- **setMCPCallback(MCPCallback)** – Sets the MCP Callback object.

IMCPCallback (Default MCPCallback interface)

This interface defines the callback methods that the Custom BS can interact with. The implementation is provided in the runtime by MCPCallback or MCPCallbackMock based on the execution time. For methods to be implemented, check the MCPCabllback class below:

- **MCPCallback** – When the BS runs as a part of MCPRuntime
- **MCPCallbackMock** – when the BS runs as a test program, an instance of MCPCallbackMock is set to RuntimeContextMock which is passed to the custom BS.

MCPCallback (Utility methods to call MCP Runtime)

This contains utility methods for the Custom Business service to interact with MCP Runtime to execute Scripts, other BS, BO operations, logging, querying the database etc.

Methods

1. **InvokeSS** – Allows custom BS to invoke a SS. The implementation invokes ServiceScriptDispatcher with the Script name and the input DOM**:**

   Parameters:
   - Script Name - (input) - String
   - Data Area – (input/output) - JBS DOM

   Example:- MCPCallback.invokeSS("M1-MCPTskLst", dataElement);

2. **InvokeBS** – Allows custom BS to invoke a BS. The implementation invokes BusinessServiceDispatcher which translates to executing the appropriate Business service. This can be used to call MCP Business services and another "Custom BS" implementation also.

   Parameters:
   - BS Name - (input) - String
   - Data Area – (input/output) - JBS DOM

   Example:- MCPCallback.invokeBS("M1-MDTGetMDTInfo", dataElement);

3. **InvokeBO** – Allows custom BS to invoke a BO. Based on the action, this will call BusinessObjectProcessor.java's appropriate method add, replace, delete or read. In all the actions, the input DataElement is updated and can be read in the following step**:**

   Parameters:
   - BO Name - (input) - String
   - Data Area – (input/output) - JBS DOM
   - Action – (input) – String – "READ", "ADD", "REPLACE", "DELETE"

Example:- MCPCallback.invokeBO("M1-Assignment", dataElement, "ADD");

4. **QueryService** – Allows custom BS to query BO XML. This will directly call the MCP Query Business service's supporting Java methods for faster result (instead of going through the Business Service Dispatcher).

   Parameters:

   - MO ID – (input) - String

   - List of Where Clause Name – (input) - Array [Where Names]

   - List of Where Value Pairs – (input) – Array [Where Values]

   - List of Sort Columns / Order Pairs - (input) – Array [Column, Order]

   - List of BO Name / Raw XML - (output) – Array [BO Name, JBS DOM]

   Example:-

   String[] whereClause = new String[] {"boStatus", "IN_SERVICE"};

   String[] sortColumn = new String[] {"CRE_DTTM", "ASC"};

   Object[] response = MCPCallback.queryService("M1_CREWSHIFT" whereClause, sortColumn);

   Response:

   response[0] (String) – M1_CREWSHIFT

   response[1] (JBS DOM - DataElement) –

   <root>

   <shiftId>11223344</shiftId>

   <boStatus>IN_SERVICE</bostatus>

   </root>

5. **GetUserInfomation** – Allows custom BS to access user / pass for current user.

   These details are fetched from MCP Runtime's NetworkManager.

   Parameters:

   - UserName – (output) – String

   - Password – (output) – String

   Example:-

   - String[] output = MCPCallback.getUserInformation();

   - System.out.println("Username = " + output[0]);

   - System.out.println("Password = " + output[1]);

6. **Info – Logs message at info level**
   Integrates with the existing MCP Logger framework.

   Parameters

   - message – (input) – String

   - Class – (input) – Class

   Example:- MCPCallback.logInfo("Info Message to be logged", CustomBSProgram.class);

7. **Debug – Logs message at debug level.**
   Integrates with the existing MCP Logger framework

Parameters

- message – (input) – String

- class – (input) – Class

Example:-MCPCallback.logDebug("Debug Message to be logged", CustomBSProgram.class);

8. **Error** – Logs message at error level. Integrates with the existing MCP Logger framework

    Parameters

    - message – (input) – String

    - class – (input) – Class

    - Example:-

    - MCPCallback.logError("Error Message to be logged", CustomBSProgram.class);

9. **isEligibleForDebug()** - Allows developer to determine if log level is set to debug before doing any debug logging. This is helpful on the device side by reducing the memory foot print by skipping the costlier MCPCallback.logDebug() call which requires String input.

    Returns Boolean

    Example:

```
if(MCPCallback.isEligibleForDebug()) {
MCPCallback.logDebug("Debug Message to be logged",
CustomBSProgram.class);
                  }
```

MCPCallbackMock (Mock implementation)

This is the mock implementation of the MCPCallback interface. Customers may extend this class to override the default implementation if necessary. This class is only used in custom BS development and testing. It is not used by the MCP Runtime.

Methods

This class will implement each of the interface methods as no op (empty stub) methods and the customer can override the methods to return specific output. For example, the customer may route all the log calls to System Console instead of to a log file.

Frequently Asked Questions

How to tell if the Custom Java BS JAR/APK Loaded

This can be confirmed from the logs by such log statements in MDT Debug Log level

- DEBUG: Custom JAR found = file:C:\MWMApp\ext\meter-read.jar

- DEBUG: Successfully loaded 1 JARS from path C:\MWMApp\ext

How to tell if the Business Service is invoked correctly

By logging the Business service input and exit points in MDT Debug Log level MCPCallback.logDebug(). You can check the logs to check if the BS call is successful

Logging Practices

Minimize logging in INFO log level i.e MCPCallback.logInfo().

DEBUG logs are written only in MDT log level DEBUG. To optimize the over-head of extra strings getting created when the Log level is INFO, an example is provided in the API section of this document in which the MCPCallback.isEligibleForDebug() check is used before logging.

Appendix F

Glossary

Acknowledgement Required
Indicates that the crew is required to manually confirm receipt of activities of this type.

activity time window
The time period during which an activity can be scheduled. See also effective time window (ETW), preferred time window (PTW), arrival time window (ATW), and service time window (STW)

activity type
Defines the nature of the task to be performed at a specific location.

alert
A situation in which predefined condition(s) have been met that require user attention or intervention.

algorithm
A set of rules for solving a problem in a finite number of steps.

allocate
A user-initiated action that locks an activity to a particular crew shift. See also assign and unassign.

allocation priority profile
See priority profile

alternate ID
A different way of uniquely identifying an activity, in addition to its system-generated primary key. A short and simple alternate ID is typically used if an activity is dispatched to a crew by voice and the activity's system-generated key is too long to communicate verbally.

application service
Defines the actions (access modes) supported by a securable function in the system. An application service exists for every transaction and zone in the system, and every business object must reference an application service. Application security is defined by granting or denying user groups access to specific application services.

appointment
An activity for which an arrival time window has been set in advance.

Appointment Booking Group
Defines valid time windows for appointment booking.

arrival time window

A period of time within an activity's effective time window (ETW) during which the activity is allowed to be scheduled, based on applicable restrictions or conditions. See also effective time window (ETW), preferred time window (PTW), and service time window (STW)

assign

An action, performed by the scheduler, that links an activity to a particular crew shift. Assigning is done only by the system. A user can unassign an activity, which unlinks it from its current crew shift and makes it available for rescheduling to any other crew shift, and then the user can allocate it to a different shift; however, a user cannot assign activities.

assignment

A crew's copy of an activity. It is used by the crew to record status changes and completion information. If the crew uses a mobile data terminal (MDT) to work their schedule, a copy of the assignment is dispatched to the crew's mobile device.

assignment in progress

An activity status indicating that a copy of the activity, referred to as an assignment, has been created. The activity remains in this status until the crew completes the assignment or returns it.

assumed equipment

Equipment (capabilities) typically associated with vehicles of a particular type.

assumed skills

Skills (capabilities) typically associated with mobile workers of a particular type.

auto dispatch

A system feature that provides automatic dispatching of activities to crews once they are logged on. Also refers to an activity type attribute that enables or disables auto dispatch for specific activity types. The Drip Mode attribute defines auto dispatch settings at the crew shift level.

base service areas

Service areas within which a mobile worker typically works.

base time

The time zone defined on installation options, which is typically the server time zone.

being scheduled

An activity status indicating that the activity has been sent to the scheduler and is in the process of being scheduled. An activity also transitions back to this status if its current assignment to a crew has been recalled or returned.

BPA

Business Process Assistant. See BPA script.

BPA script

A set of commands that run on the client's browser and guide a user through a business process.

break

A planned time period within a crew shift during which the crew is eligible for taking a break. During a break, the crew may not carry out work or travel.

break duration

Duration of the break.

break window duration
Duration of the period of time during which the actual break can be taken. For example, a 15-minute morning break might have a 60 minute window during which the break can be taken.

calculated
Computed; arrived at or determined by mathematical calculation.

cancel
An action that typically terminates the lifecycle of a business entity. For example, canceling an activity stops the effort of scheduling and performing it.

cancelled (status)
The status resulting from a cancel action. This is typically a final state. For activities, a cancelled status indicates that the host system or the dispatcher has cancelled the activity and it is no longer being considered for scheduling. For crew shifts, a cancelled status indicates that the shift was manually cancelled.

capability
An ability associated with a resource that qualifies the resource for a certain type of work.

capability type
A type of ability that qualifies a resource for a certain type of work. Capability types may define human skills or equipment.

class
A broad grouping or categorization.

clustered tasks
If a crew shift contains multiple tasks of the same class and status that are back-to-back (with only a short period of idle between), then the system collapses the individual tasks into one clustered task in the Scheduling Gantt task area. You cannot take action on clustered tasks; you must first expand the shift to uncluster the tasks.

collision
A clash; a conflict; a meeting of objects that causes an adverse impact. See collision detected.

collision detected
An activity status that occurs when more than one completed assignment is received by the server for the same activity.

Common Dispatching Functionality
A component of Oracle Real-Time Scheduler that provides functionality for dispatchers to effectively monitor and manage workforces and activities.

Common Dispatching Interface
The primary point of user interaction for the Common Dispatching Functionality (CDF).

complete
An action that finalizes the processing of a business entity and ends its lifecycle.

complete (activity status)
An activity status indicating that the requested work has been done and completion details are available to be sent back to the requesting host system. A completed assignment has been received (one and only one) for an activity and all other assignments for that activity are finalized. The activity remains on the mobile device until logoff. If an activity was in a Collision Detected status

and the dispatcher has identified which completed assignment to use, the activity transitions to the Completed status.

complete (shift status)
A crew shift status indicating that the crew is done working. This is also known as the End Of Shift. A shift cannot be completed if any of its tasks are not completed or returned.

completion remarks
Comments typically entered by a crew when an activity is completed.

crew
A uniquely named group of resources (mobile workers and vehicles) scheduled to perform work.

crew allocation
The specific mobile worker(s) and vehicle(s) assigned to a crew on a particular shift.

crew shift
A planned period of time in which a crew is available to work.

crew shift template
A tool for quickly generating crew shifts based on pre-defined parameters.

crew size
The minimum number of mobile workers needed to perform activities of a particular type.

dashboard
A portal that always appears on the desktop. Its zones contain tools and data used to complete common tasks.

deadreckoning
A method of estimating the current position of a moving vehicle where GPS is not available or updates are received infrequently.

deployment
A specific cut or build of a deployment type for a specific language; a packaged application to be delivered to a mobile device terminal (MDT); also refers to the gathering up of application components (maintenance objects, business objects, UI maps, SS, etc.) and their preparation for delivery to and usage on the MDT.

deployment part
A collection of deployment items, such as maintenance objects, business objects, UI maps, etc.

deployment type
Defines attributes of an application to be deployed, including authorized user groups, supported MDT types, messages used, and objects included.

detected
Discovered; found.

disable
To make unavailable or inactive.

dispatch
An action, either system- or user-initiated, that locks an activity's sequence in the shift schedule and marks it as ready to be sent to the crew. See dispatched status.

dispatched (status)

An assignment status indicating that the assignment has been synchronized to the crew and the activity is now locked to its assigned crew shift and sequence. If required, manual acknowledgement has been given.

dispatch area

A predefined set of service areas and service classes monitored by one or more dispatchers.

dispatcher

An individual responsible for monitoring and managing day-to-day field operations.

drip mode

A crew shift attribute that indicates how the system should handle dispatching of activities to that crew shift. Valid options are:

- **None:** Automatic dispatching is disabled. All scheduled activities must be manually dispatched to the crew's shift by the dispatcher or by another system-invoked process.

- **Auto All:** All scheduled activities are automatically dispatched to a shift once it has started. This applies to any additional activities being scheduled throughout the duration of that shift. This excludes any activities where the Auto Dispatch attribute is set to No for the Activity Type.

- **Standard (Drip Mode):** Only a fixed number of scheduled activities are dispatched to the crew at any given point of time. The fixed number is specified as the Drip Horizon.

drip horizon

The number of activities to dispatch when drip mode is set to Standard or Hybrid.

effective date

The date on which a characteristic, parameter value, or other element becomes effective. See also arrival time window (ATW), preferred time window (PTW), and service time window (STW)

effective time window

The time period within which an activity should be scheduled, based on values sent from the host.

emergency task

A task of highest priority, indicating that its scheduling takes precedence over all non-emergency tasks.

en route

An assignment or POU status indicating that the crew is on its way to the service address or meeting address.

enable

To make available or active.

end odometer

The final odometer reading (total distance traveled by the vehicle) recorded at shift logoff.

equipment

Device or equipment used to perform certain types of work. In ORS, equipment is a capability that can be associated with one or more vehicles (resources).

estimated duration

The approximate length of time it should take to complete an activity.

field referenced activity

An activity created by a crew when connectivity problems prevent the crew from receiving an activity and require the dispatcher to voice dispatch it. The FRA references the original activity and, when connectivity is re-established, the FRA is synchronized with that activity on the server.

finalized

Indicates that an object's current status represents the end of its defined lifecycle. For example, in the base system an activity is finalized if it is in a canceled or completed state.

foreign key

A referential constraint between two tables. The foreign key identifies a column or a set of columns in one (referencing) table that refers to a column or set of columns in another (referenced) table.

foreign key reference

Defines the program used to construct a maintenance object's info string. It may also define the transaction used to display, add, and update instances of the maintenance object.

Gantt

A tool that allows dispatchers to view shifts in a timeline and perform actions on activities for those shifts.

Geocode

The latitude and longitude associated with an address, which is used for optimizing routes and displaying crews and activities on a map.

Global Positioning System

A navigation system based on a network of satellites that send out radio signals to pinpoint an entity's location.

holding scheduler

A holding scheduler is not associated with a schedule manager and therefore does not schedule shifts or tasks. It is designed to capture tasks and shifts that cannot be assigned to any online scheduler. One and only one holding scheduler should exist. Any task or shift attached to a holding scheduler is evidence of a scheduling error.

host external system

A computer system external to Oracle Real-Time Scheduler that sends work orders (activities) to and receives updated information from Oracle Real-Time Scheduler.

idle time cost

A scheduling factor that indicates whether or not crew idle time cost should be considered by the scheduler.

in service

A crew shift status indicating that the crew is considered to be performing work. The crew may be traveling to an activity location, working an activity, or idle between activities. The crew status transitions to In Service automatically when an activity is started or a POU or break is ended. The user can manually transition to this status.

issue

A situation, problem, or potential problem requiring attention or resolution.

issues detected

An activity status indicating that the activity is not valid and processing cannot proceed. Manual dispatcher intervention is required to resolve the issue. Once the problem has been fixed, the status of the activity can transition to Being Scheduled.

key performance indicators

Quantifiable measurements that support dispatcher decision making and in-day exception handling. A KPI may apply to activities and crew shifts that dispatchers are responsible for in their current shift.

late cost

A scheduling factor that defines a cost associated with late arrival to an activity. It is used to discourage scheduling of the activity outside its time window.

lead time

Number of minutes before a shift is planned to start that the reserved capacity is released.

limited time

Indicates an entity or capability that has an effective and expiration date.

location

A predefined physical location, such as a service center.

logoff delay

Number of minutes before the end of the shift that the scheduler should stop scheduling work. This is used to reserve time at the end of the shift for any end-of-shift work.

logon delay

Number of minutes after start of the shift that the scheduler should begin scheduling tasks to the shift. This is used to reserve some time at the beginning of the shift for performing review and preparation work.

location based services

Crew location and tracking services that utilize GPS data obtained from a mobile resource.

maintenance object

A group of tables maintained together within the system.

manual

Occurring as a result of a user action, rather than automatically by the system.

manual acknowledgement

See acknowledgement required.

MapViewer

A tool, based on Oracle Application Server MapViewer, that provides a visual representation of monitored activities, crews, and routes on a map in the Common Dispatch Interface.

message category

A set of application messages.

metadata

Data about data. Metadata may define formatting, validation, possible values, data type, etc.

missing equipment

Equipment typically associated with the vehicle type but missing from this vehicle.

missing skills

Skills that a mobile worker of the specific job type is assumed to have, but this mobile worker does not have.

mobile data terminal

A computerized device typically used to communicate with a central dispatch office.

mobile worker

An individual performing work as part of a crew. A mobile worker is a resource.

odometer

An instrument that indicates distance traveled by a vehicle.

on hold

An activity status indicating that the activity was placed on hold, either by the dispatcher or by the system based on a business rule. An activity may be placed on hold because something needs to be done before the order can be worked (e.g., parts, permit, etc.). An activity cannot be put on hold once an assignment has been created (unless the assignments are returned). The activity will remain in this status until a user removes the hold manually or a condition occurs that prompts the system to remove the hold, at which time it will return to its previous status.

on site

An assignment and POU status indicating that the crew is at the service site or event location site.

optimization area/set

A set of sub-regions, called optimization areas, within a scheduler area. The optimization set is used when a scheduler area has too many tasks for a single scheduler to manage. A geocode pair defines an optimization area boundary.

out of service

A crew shift status indicating that the crew is not performing work. The crew may be attending a meeting, taking a break or having a vehicle breakdown, etc. When a crew goes out of service, they provide a reason and an estimated duration. At logon, a shift is Out of Service until a task is started and then it switches to In Service. Starting a POU or break automatically sets the crew status to Out of Service. The user may manually transition to this status.

overridden POU

A POU that has been modified manually and no longer matches the POU template from which it was generated.

overridden shift

A crew shift that has been modified manually and no longer matches the template from which it was generated.

overtime cost

A scheduling factor that defines the cost of exceeding the planned shift duration.

pending (shift status)

A shift's initial status when it is first created, before it is considered planned and ready to be started.

pending dispatch

An assignment status indicating that the assignment has not yet been synchronized to the crew's mobile device or has not been manually acknowledged by the crew, if the activity requires manual acknowledgement.

period of unavailability

A period of time in which a crew is planned not to perform work, such as while attending a meeting.

planning horizon

The number of days into the future to plan for crew shifts and periods of unavailability. For example, if the planning horizon is set to 60, the system generates crew shifts and periods of unavailability (POU) sixty days into the future. Generation occurs in a sliding window in that each day generates data for the last day incrementally. At any given time, the number of days of planned shifts and periods of unavailability is equal to this value. Planning horizon should be equal to or greater than the scheduling horizon.

portal

A page in an application that contains one or more zones.

posting action

A posting action is associated with every scheduler registry entry. It identifies the type of change made to an entity that impacts scheduling, and is used to communicate this information to the scheduler manager.

postpone

A user-initiated action that occurs when a crew has not yet started work on an activity but must postpone the scheduled work.

postponed

An activity status resulting from the postpone action.

preferred time window

The time period within which it is ideal to schedule an activity, based on values sent from the host. See also effective time window (ETW), arrival time window (ATW), and service time window (STW)

preview calendar

Calendar based on the changes that have been entered but not yet applied.

primary key

A single column or set of columns in a table that uniquely identifies each row in the table.

priority profile

A predefined set of allocation factors used by the scheduler to prioritize the scheduling of an activity relative to other activities.

queue

A list of items, such as activities, that are awaiting processing.

queued for dispatch

An activity status indicating that the activity has been successfully scheduled to a particular crew shift and is no longer eligible for rescheduling during optimization, but has not yet been synchronized to the crew's mobile device.

recall

An action, typically initiated by a dispatcher, that requests the return of an assignment from its assigned crew.

recommended allocation

One or more mobile workers and/or vehicles identified as appropriate or desirable for a crew shift.

recycle (alert)

An action that reassigns an alert to another dispatcher if the currently assigned dispatcher does not respond to it within a configurable amount of time.

reference time

Time used by the scheduler as input to cost-calculations that are sensitive to the current time. Applies only to time-dependent costs.

relative efficiency

Scheduling factor that defines a mobile worker's efficiency as a percentage of the expected efficiency for mobile workers of this type.

relative speed

Scheduling factor that defines a vehicle's speed as a percentage of the expected speed for vehicles of this type.

reserve capacity

A scheduling concept allowing shifts to reserve a portion of their time for a specific service class of work.

reserve capacity percentage

Percentage of shift time to be reserved for work associated with the reserve capacity service class.

reserve capacity type

Service class for which a percentage of the crew shift's capacity is reserved.

resource

A workforce resource, such as a crew, vehicle, mobile worker, or dispatcher.

resource class

A broad classification of resource. Resource classes include mobile worker, vehicle, crew, and dispatcher, and may include additional user-defined classes.

resource planner

An individual responsible for allocating workforce resources to perform work across multiple shifts.

Resource Planning and Scheduling

A component of Oracle Real-Time Scheduler system that handles resource planning, resource management, service management, and scheduling.

returned

An assignment status indicating that the crew has returned the assignment or the assignment was recalled. This is the end of the assignment's lifecycle; the activity will be rescheduled.

scheduler

A component of the Oracle Real-Time Scheduler system that assigns tasks to crew shifts in such a way as to optimize the use of resources and reduce overall costs. An implementation typically runs multiples schedulers in parallel for performance and backup.

scheduler configuration

Defines parameters used to control the scheduler.

scheduler registry

Contains a record for each object currently owned by a scheduler.

scheduling horizon

The number of days into the future the scheduler will consider when scheduling crew shifts and activities. Only shifts that have already started or are planned to start within that time frame, as well as activities that may be scheduled during that time, are considered. For example, if the scheduling horizon is set to 21, the scheduler considers shifts and activities for the next 21 days. Oracle recommends that the Scheduling Horizon be kept as short as is practical while still meeting customer business requirements. The optimum value for scheduling horizon strikes a balance between long term planning and short term flexibility.

service area

A logical boundary of an organization's territory that may or may not be based on geography.

service class

A broad categorization of activity types, such as Meter Work or Emergency Work.

service level agreement

A part of a service contract where the level of service is formally defined.

service time window

A period of time within an activity's arrival time window (ATW) during which it is preferable to schedule the activity. See also effective time window (ETW), preferred time window (PTW), and arrival time window (ATW).

shift

Short version of crew shift.

shift cost

A scheduling factor that defines the cost of utilizing a shift.

shift cost profile

A preconfigured set of shift cost-based factors that affect scheduling decisions.

shift promotion cost

A scheduling factor that specifies the multiplier by which the shift promotion cost for an activity type is increased or decreased. This is used to discourage creation of additional shifts.

shift weekly template

A tool used to define a cyclical weekly pattern made up of crew shift templates.

site address

The address of the actual entrance to the site of service, which may be different than the service address.

site delay
The delay involved in arriving at this site.

Short Messaging Service
A service that allows the interchange of short text messages between mobile telephone devices.

shuffler
A shuffler changes the current solution (schedule). The scheduler operates by repeatedly calling a shuffler to change the current solution and calculating the new solution cost to determine whether the change was beneficial or not. The scheduler employs many shufflers, each with its own specialization.

skill
A capability that indicates the ability to perform certain types of work. Mobile workers have skills; mobile worker types have assumed skills; activity types require skills.

SLA Flexibility
A scheduling factor that defines the period of time within which an activity must be scheduled to avoid incurring the SLA Priority cost.

SLA Priority
A scheduling factor that defines the fixed cost that will be applied for working an activity outside its service time window's SLA Flexibility period.

snooze (alert)
To stop (sleep) for a certain amount of time before reissuing the alert.

solution
The scheduler (activity-to-shift assignments) generated by the scheduler.

solution cost
The value calculated by the objective function for a schedule. This value is indicative of how well the schedule attains its cost goals. The lower the value, the better.

sort sequence
The order in which items appear in a list.

start odometer
The initial odometer reading (total distance traveled by the vehicle) recorded at shift logon.

status reason
The reason why an entity transitioned to its current state.

suspend
A user-initiated action that occurs when a crew has started work on an activity but cannot complete the work due to time, material or other resource constraints. When the crew suspends an order, they must provide an estimate of the time required to complete the work.

suspended
An assignment status resulting from the suspend action. The assignment's life cycle is not terminated and remains on a crew shift until the assignment is completed, a dispatcher re-assigns the activity to another shift (a new assignment is created), or the crew logs off the shift and the assignment is returned. When a crew is ready to resume work on a suspended activity, it transitions to En Route status.

synchronize

To communicate data between the server and the MDT so that both have the same up-to-date information.

task

Any activity, POU, or break that occupies time on a crew shift's schedule.

task class

A classification of task. Task classes include activity, POU, and break. Within each task class, there can be any number of task types.

task type

Defines the attributes of a particular type of activity task, POU task, or break. Examples of activity tasks are Meter Read or Equipment Installation.

template

A pre-defined model or example used as a guide to make other objects. POU templates are used to generate actual POUs. Crew shift templates are used to generate crew shifts.

travel time cost

A scheduling factor that defines a cost associated with crew travel time.

UI map

An entity that contains predefined HTML for building a map zone. Its schema defines the fields whose values are inserted into the map at run time.

unassign

A user-initiated action that unlinks an activity from the crew shift to which it has been assigned by the scheduler.

user-defined

Configured or set by a user rather than being assigned by the system.

vehicle

A non-human mobile resource used by a crew.

window cost

A scheduling factor that defines the preference of an activity's time window relative to other time windows.

work calendar

An object that defines work holidays within a defined period of time.

work done

An assignment status indicating that the crew completed the work but has not entered the completion details yet. The crew may enter the details later in the day. This is typical to crews using SMS to work their schedule. This is the end of the assignment's lifecycle. The activity remains in the Assignment in Progress status while waiting for the completion details to be entered.

work sequence

The task's sequence within the crew shift.

zone

A section of a portal. Examples of zones include Query zones, List zones, and Actions zones.

Index