Field Service Developer Guide

Version 68.0, Winter ‘27

Last updated: October 2, 2026

© Copyright 2000–2026 Salesforce, Inc. All rights reserved. Salesforce is a registered trademark of Salesforce, Inc., as are other names and marks. Other marks appearing herein may be trademarks of their respective owners.

CONTENTS

Chapter 1: Field Service Developer Guide 1

Get Ready to Develop with Field Service 2
API End-of-Life Policy 2
Field Service Data Objects 2
- Field Service Core Data Model 3
- Field Service Inventory Management Data Model 7
- Field Service Preventive Maintenance Data Model 9
- Field Service Product Service Campaign Data Model 11
- Field Service Warranty Management Data Model 11
- Field Service Pricing Data Model 12
- Field Service Object References 13

Field Service API 47
- Field Service Flow 47
- Field Service Mobile Settings 52
- Service Report Template 55
- Appointment Bundling REST APIs 61
- REST Reference 92

Field Service Metadata API 99
- Flow for Field Service 99

Field Service Tooling API 100
Field Service Invocable Actions 100
- Convert Time Zone 101
- Create Engagement Topic 102
- Get Agent Context 103
- Get Service Appointment Time Slots 104
- Get Current Time 106
- Get Engagement Topics 107
- Get Field Service Object Field 108
- Schedule Service Appointment 109

Field Service Apex Reference 110
- ConnectApi Namespace 111
- FSL Apex Namespace 114

Field Service Custom Triggers 171
Field Service Code Examples 185
- Create a Service Report with Apex 185
- Generate Work Orders on Maintenance Plans with Apex 187
- Code Examples: Dispatcher Console Custom Actions 188
- Create Service Appointment Lists in the Dispatcher Console 192

Field Service Mobile App 193

Get Started with Lightning Web Components in the Field Service Mobile App 193
Build Custom Lightning Web Components for Service Documents 207
Configure Deep Linking for the Field Service Mobile App 214
Add Lighting Web Components for Plug-Ins to the Field Service Mobile App 221

Index 232

CHAPTER 1 Field Service Developer Guide

In this chapter ...

- Get Ready to Develop with Field Service
- API End-of-Life Policy
- Field Service Data Objects
- Field Service API
- Field Service Metadata API
- Field Service Tooling API
- Field Service Invocable Actions
- Field Service Apex Reference
- Field Service Custom Triggers
- Field Service Code Examples
- Field Service Mobile App

Field Service is a powerful, highly customizable toolbox of features that you can use to set up and manage a multiplatform, mobile service operation. This guide offers reference information and code examples to help you understand the structure of Field Service features and customize them with code.

This guide includes:

- Object diagrams that illustrate the relationships between field service data objects
- Reference information for field service objects in the REST, Metadata, and Tooling APIs
- Apex reference information for the Field Service managed package
- Apex code examples that you can use to complete common field service tasks

If you're new to field service, we recommend checking out Get on the Road with Field Service in Trailhead. Then, find curated links to documentation, blogs, Trailhead modules, and other resources through the Field Service learning map (English only)

Happy coding!

SEE ALSO:

Install the Field Service Managed Package

Field Service Mobile App

Get Ready to Develop with Field Service

To prepare to work programmatically with field service features, verify that Field Service is enabled in your org.

1. From Setup, enter `Field Service Settings` in the Quick Find box, then select **Field Service Settings**.
2. Verify that Field Service is enabled.
3. Click **Save**.

You now have access to Salesforce's standard field service objects, which are listed in this guide. However, that's just the beginning! To get your feet wet before you start working programmatically with Field Service, it's a good idea to walk through the setup tasks outlined in Set Up Field Service.

API End-of-Life Policy

Salesforce is committed to supporting each API version for a minimum of 3 years from the date of first release. To improve the quality and performance of the API, versions that are over 3 years old sometimes are no longer supported.

Salesforce notifies customers who use an API version scheduled for deprecation at least 1 year before support for the version ends.

Note: Version 20.0 of REST API and SOAP API have now been deprecated and are no longer supported. You can continue to access this legacy API version until Summer '22 is released, at which point this legacy version will be retired and will become unavailable. For more information, see this Knowledge Article: Salesforce Platform API Versions 7.0 through 20.0 Retirement.

Note: Versions 21.0 through 30.0 of REST API and SOAP API will be deprecated in the Summer '22 release. For more information, see this Knowledge Article: Salesforce Platform API Versions 21.0 through 30.0 Retirement.

Field Service Data Objects

Field Service is based on a suite of standard and custom Salesforce objects that relate to each other in different ways. These objects also serve as the foundation of the Field Service managed package and mobile app.

The following object relationship diagrams and reference information are your field service object dictionary. Use them to guide your implementation decisions. You can create, retrieve, update, or delete field service records and objects, through our SOAP and REST APIs. You can query these objects using SOQL (Salesforce Object Query Language).

IN THIS SECTION:

Field Service Core Data Model

Use Field Service's core objects to complete essential field service tasks such as managing work orders, defining your service territories, and tracking your workforce.

Field Service Inventory Management Data Model

Use Field Service's inventory objects to track the storage, request, consumption, return, and retirement of items in your inventory.

Field Service Preventive Maintenance Data Model

Manage periodic maintenance with the help of maintenance plans, which define the maintenance schedule for specific assets. Maintenance plans typically reflect the terms in a customer's service contract or entitlements.

Field Service Product Service Campaign Data Model

Use product service campaign to record the actions to address situations such as product recalls, manual firmware upgrades, safety or compliance audits, or end-of-life communications. Assets affected are associated with a campaign using product service campaign items. Campaign and campaign items can then be associated with work orders and return orders as needed to complete the work.

Field Service Warranty Management Data Model

Use warranty items to record details of the labor, parts, and expenses, along with any exchange options, that are provided to rectify issues with products sold or installed. Create standard warranties for products and product families and, for products you install, record details of additional or extended warranties along with exclusions and void terms.

Field Service Pricing Data Model

Link work orders to products or assets in your org to track product pricing and work being performed on your customers' installed products.

Field Service Object References

Object reference for standard and custom Salesforce objects used in Field Service.

SEE ALSO:

Salesforce SOAP API Developer Guide

Salesforce REST API Developer Guide

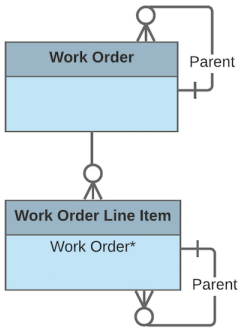
Salesforce SOQL and SOSL Reference

Field Service Core Data Model

Use Field Service's core objects to complete essential field service tasks such as managing work orders, defining your service territories, and tracking your workforce.

Note: Asterisks in the following diagrams signify required fields.

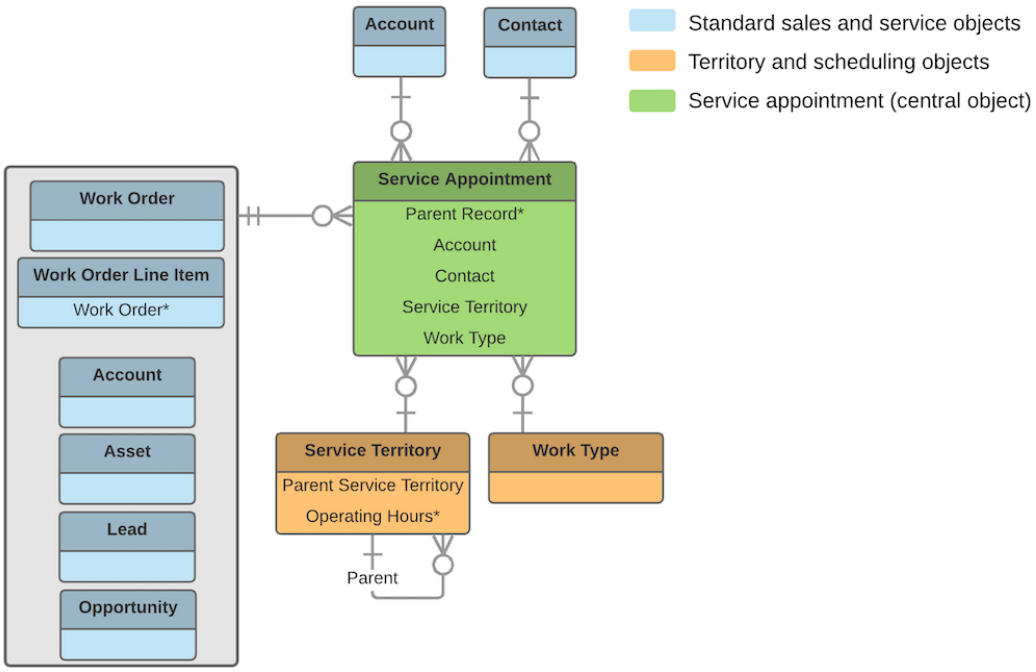
Work orders represent work to be completed for your customers, and are central to field service operations in Salesforce. To divide the work further for billing purposes or to track subtasks, add work order line items, which are child records of work orders.



Work orders offer a great deal of flexibility. They can be associated with many types of records, including:

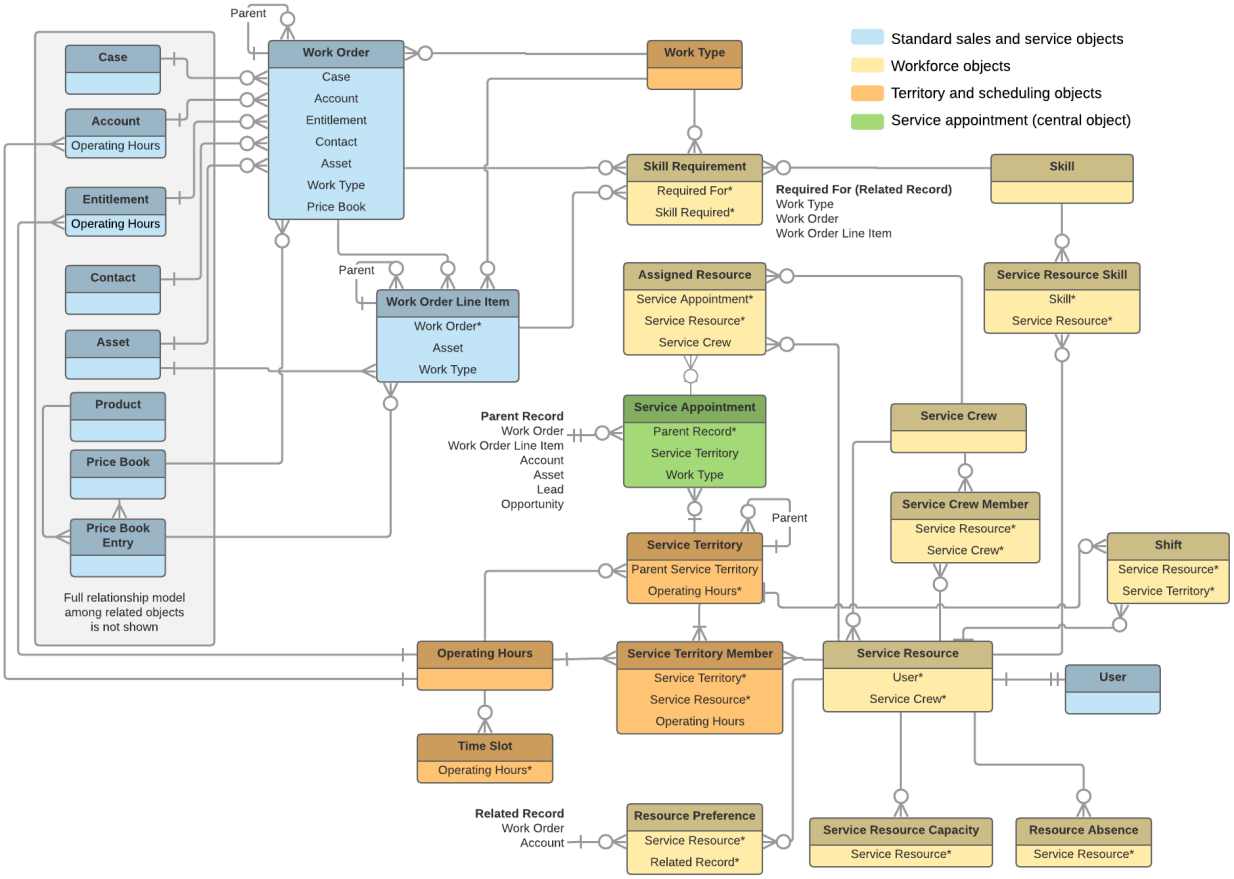
- *Assets*, to track work performed on a specific asset
- *Cases*, to indicate that the work is being performed as part of a customer case
- *Accounts* and *contacts*, representing the customer
- *Entitlements* and *service contracts*, to indicate that the work is being done to fulfill a service-level agreement

While work orders describe the work to be performed, *service appointments* represent the visits your team makes to the field to perform the work. They include scheduling settings such as an arrival window, scheduled start and end times, and appointment duration. Every service appointment has a parent record. The parent record is typically a work order or work order line item, but you can also add child service appointments to accounts, assets, leads, or opportunities to track related visits. A record can have multiple child service appointments; for example, a work order may have two service appointments if two visits were needed to complete the work.



If your team often performs the same tasks for multiple customers, create *work types* to standardize your field service work. Work types are templates that can be applied to work orders and work order line items. On a work type, you can define the duration of the work and add *skill requirements* to indicate the level of expertise needed to complete the work. You can also opt to auto-create a child service appointment on any record that uses the work type.

Work orders and service appointments are supported by a variety of objects that control when and where the work occurs, the nature of the work, and who performs it.



Who Performs the Work

Members of your mobile workforce are represented in Salesforce as *service resources*. A service resource represents an individual technician who can be assigned to a service appointment. You can also create *service crews*, which are groups of service resources with complementary skills and experience that can be assigned to appointments as a unit.

To assign a service resource to a service appointment, create an *assigned resource* record. Assigned resources contain lookups to a service resource and a service appointment. To assign a service crew to an appointment, first create a representative service resource record with a resource type of Crew. Then, create an assigned resource record that looks up to the Crew service resource.

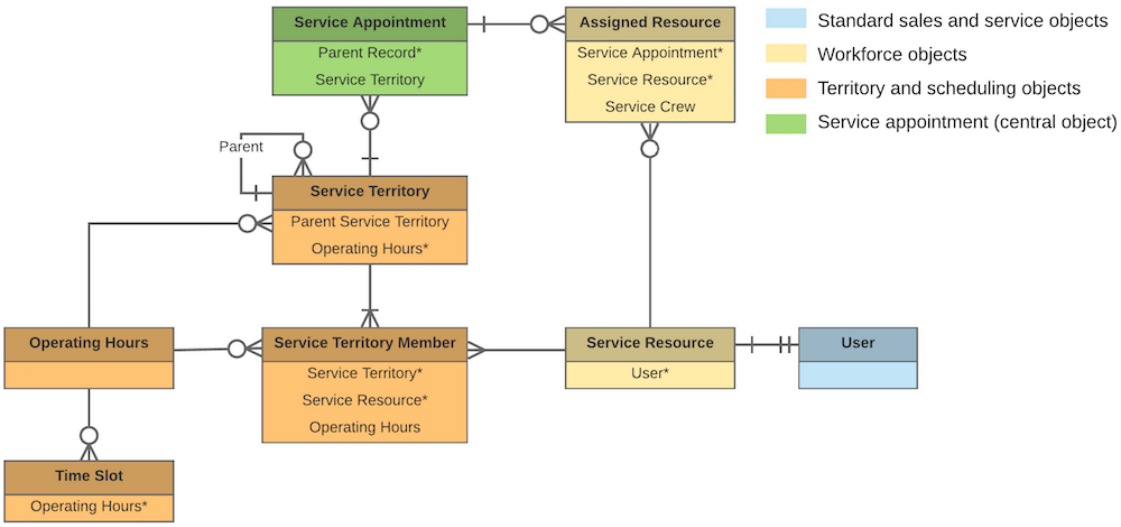
Service resources come with several objects that let you define their skills and availability:

- *Service resource skills* represent a service resource's certifications or levels of expertise.
- *Resource capacity* records track the hourly or job-based capacity of contractors.
- *Resource absences* represent time when a service resource needs to miss work.
- *Resource preferences* designate specific service resources as preferred, required, or excluded on a work order or account.

Where the Work Occurs

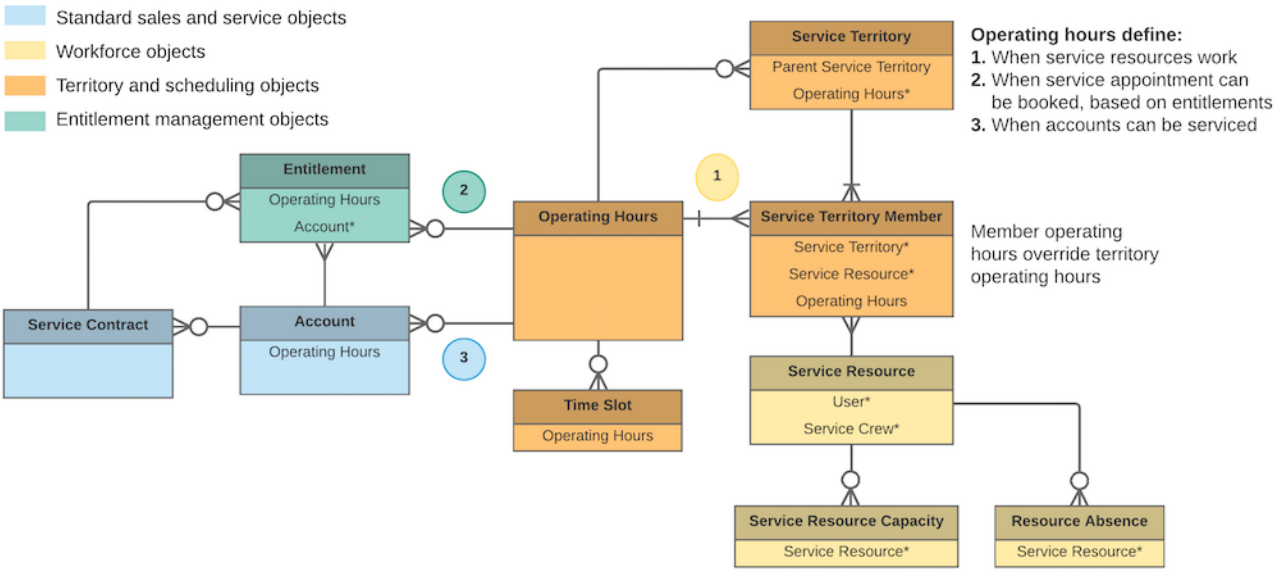
Service territories are the places where your team can perform field service work and are a way to organize your service resources. They typically represent geographic territories such as cities or counties but may also represent functional divisions like sales versus service.

A *work order* can be associated with one service territory. *Service resources* are assigned to one or more service territories as *service territory members* to indicate that they are available to work in the territory.



When the Work Occurs

Operating hours indicate when your team can perform field service work. They can be assigned to accounts, service territories, and service territory members. To add detail to operating hours, create *time slots*, which represent the hours of operation in a particular day.



If a customer *entitlement* includes terms regarding when the customer has access to field service, you can track these hours on the entitlement using the **Operating Hours** field (API name: `SvcApptBookingWindowsId`). For example, if Customer A is entitled to service Monday through Friday from 8 AM to noon, but Customer B is entitled to 24/7 service, you can create operating hours for each customer and assign them to the related entitlement.

SEE ALSO:

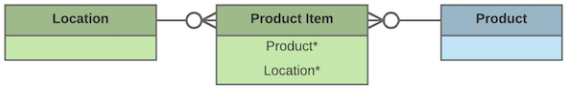
Guidelines for Creating Operating Hours for Field Service

Field Service Inventory Management Data Model

Use Field Service's inventory objects to track the storage, request, consumption, return, and retirement of items in your inventory.

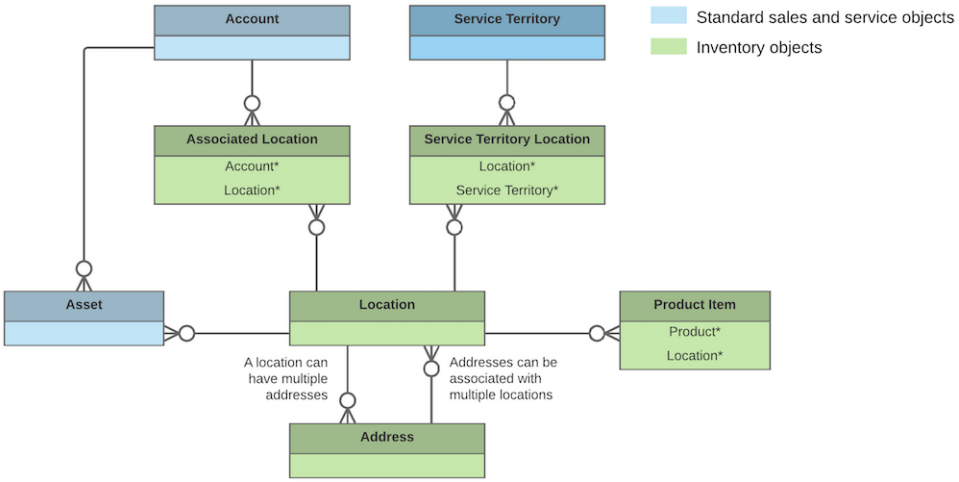
Note: Asterisks in the following diagrams signify required fields.

Inventory management starts with *product items*, which represent the stock of a particular product at a particular location. Each product item is associated with a *product* and a *location* in Salesforce. For example, if you have 50 hammers stored in your Warehouse A location and 200 stored in Warehouse B, create one product item for each location. Product items list a quantity at the location that is updated automatically when inventory is transferred or consumed.



If the Inventory Location option is selected on a location, it means that inventory can be stored there. Product items can be associated only with inventory locations.

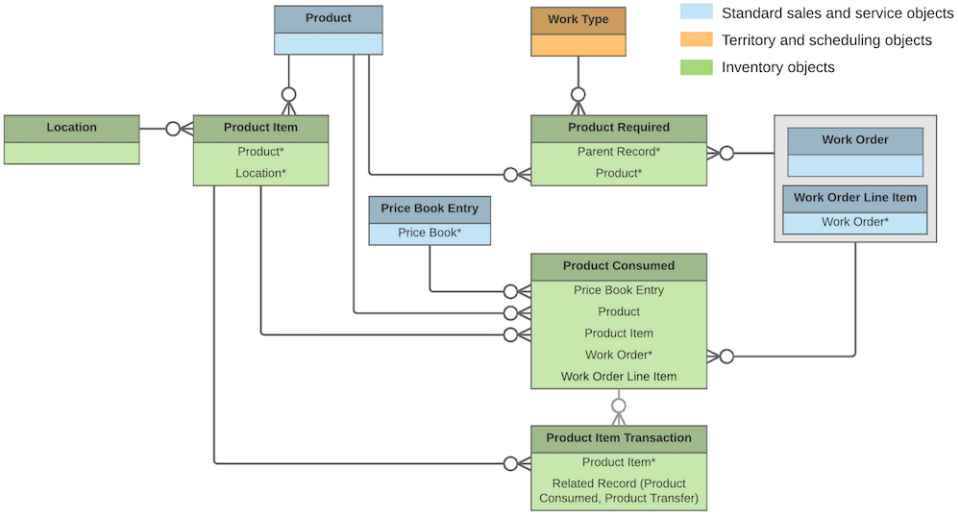
You can link a location to multiple *accounts* and *service territories*. For example, if a location is a shopping mall, you can choose to associate it with every account that operates a store in the mall. You can also create *addresses* for a location, such as a mailing and home address. And to keep track of customer sites, create *associated locations*, which contain lookups to an *account* and a *location*.



If a particular product is needed to complete a field service job, add *products required* to ensure that the assigned service resources arrive prepared. Products required can be child records of *work orders*, *work order line items*, and *work types*. Work order and work order line items inherit their work type's products required.

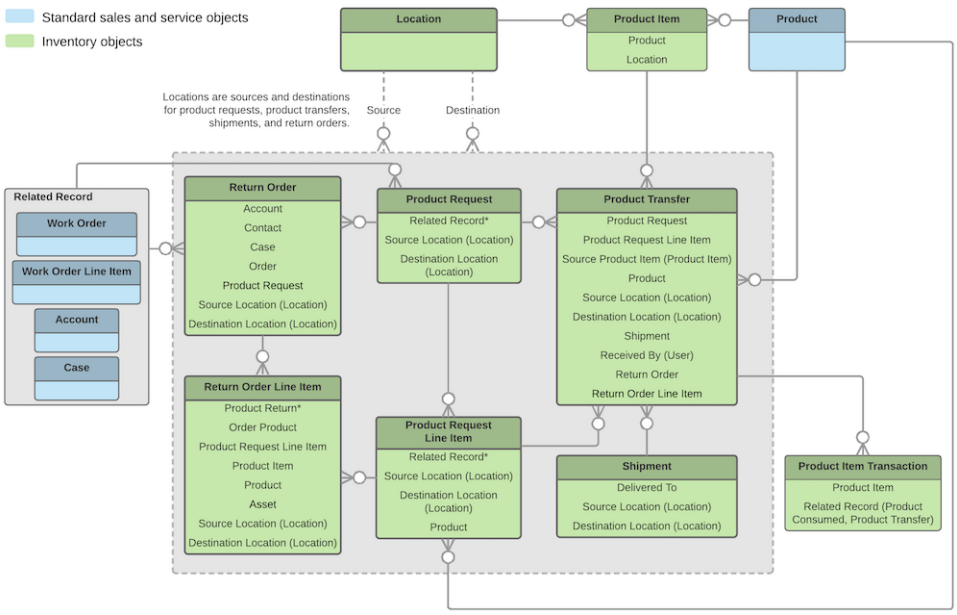
When a product is consumed during the completion of a work order, track its consumption by creating a *product consumed* record. You can add products consumed to work orders or work order line items. Track product consumption at the line item level if you want to know which products were used for each line item's tasks.

The way you use products consumed depends on how closely you want to track the state of your inventory in Salesforce. If you want to track the entire life cycle of items in your inventory, including their storage, transfer, and consumption, link your product consumed records to product items. This approach ensures that your inventory numbers update automatically to reflect the consumption of products from your inventory. If you want to track product consumption only, however, specify a **Price Book Entry** on each product consumed record and leave the **Product Item** field blank.



The movement of items into and out of your inventory and between locations is tracked using these objects:

- *Product requests* are orders for products, which you might create when stock is running low.
- *Product request line items* are subdivisions of a product request.
- *Product transfers* track the movement of product items between inventory locations.
- *Shipments* represent the shipment of product items between locations.
- *Product item transactions* describe actions performed on a product item. They're auto-generated records that help you track when inventory is replenished, consumed, or adjusted.
- *Return orders* track the return of a product item due to damage, order errors, or other reasons.
- *Return order line items* are subdivisions of a return order.



SEE ALSO:

- Set Up Your Field Service Inventory
- Guidelines for Transferring Inventory
- Guidelines for Consuming Inventory
- Common Inventory Management Tasks

Field Service Preventive Maintenance Data Model

Manage periodic maintenance with the help of maintenance plans, which define the maintenance schedule for specific assets. Maintenance plans typically reflect the terms in a customer's service contract or entitlements.

Note: Asterisks in the following diagram signify required fields.

A *maintenance plan* can be linked to an *account* and *contact*—representing the customer—and, if the customer has one, to a *service contract*.

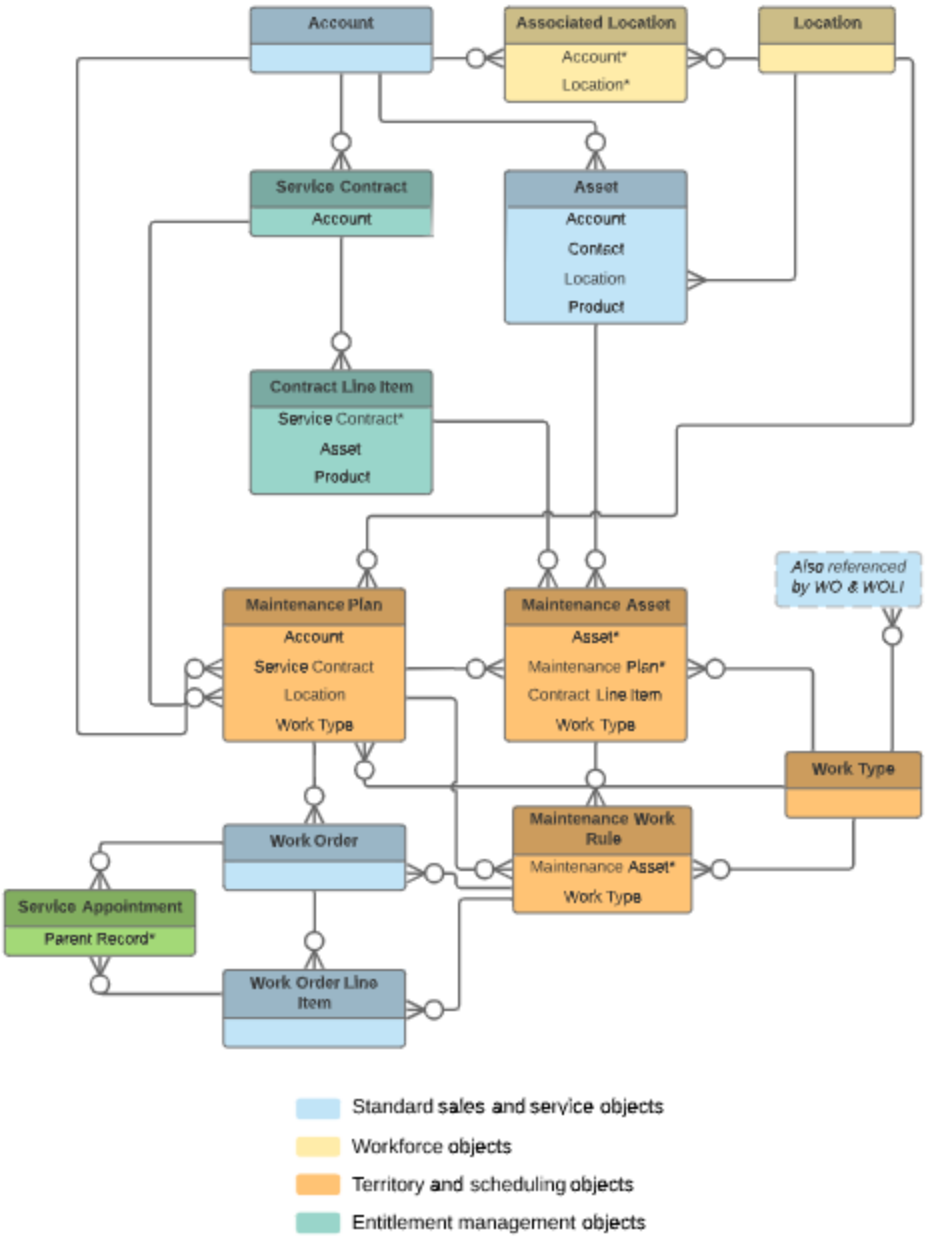
A maintenance plan can cover multiple *assets*. For example, you can create a maintenance plan to track the monthly maintenance of 20 laser printers installed in a customer's office building. The assets covered by a maintenance plan are represented as child records of the maintenance plan called *maintenance assets*. For more complex reoccurring maintenance, you can define *maintenance work rules* for most assets and maintenance plans. With this feature you can, for example, set up a schedule for minor monthly maintenance and a major yearly service. You can also link a maintenance plan to a *location* to indicate where the assets are installed. For example, link a plan to a location of the Site type that represents the office building where the printers are installed.

After you create a maintenance plan or maintenance work rules, it's time to generate *work orders* for the planned maintenance visits. Maintenance plans come with a Generate Work Orders quick action, which can also be called with Apex code. Your maintenance plan settings determine how many work orders and work order line items are generated at once and what their settings are. You can opt to generate one work order per maintenance asset for each visit, or a parent work order for each visit with one work order line item per maintenance asset.

Maintenance plans, maintenance assets, and maintenance work rules can be associated with a *work type*:

- If you specify a work type on a maintenance plan, the plan's work orders use that work type.

- If you specify a work type on a maintenance asset, generated work orders that are associated with the maintenance asset use the maintenance asset's work type.
- If you specify a work type on a maintenance work rule, generated work orders that are associated with the maintenance work rule use the maintenance work rule's work type.



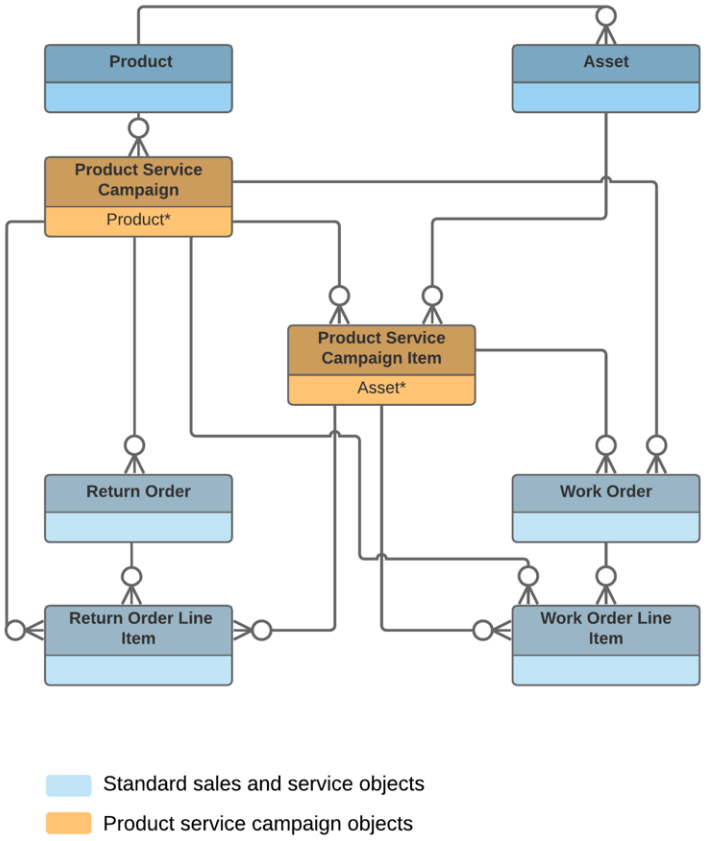
SEE ALSO:

Generate Work Orders on Maintenance Plans with Apex

Field Service Product Service Campaign Data Model

Use product service campaign to record the actions to address situations such as product recalls, manual firmware upgrades, safety or compliance audits, or end-of-life communications. Assets affected are associated with a campaign using product service campaign items. Campaign and campaign items can then be associated with work orders and return orders as needed to complete the work.

Note: Asterisks mean these fields are required.

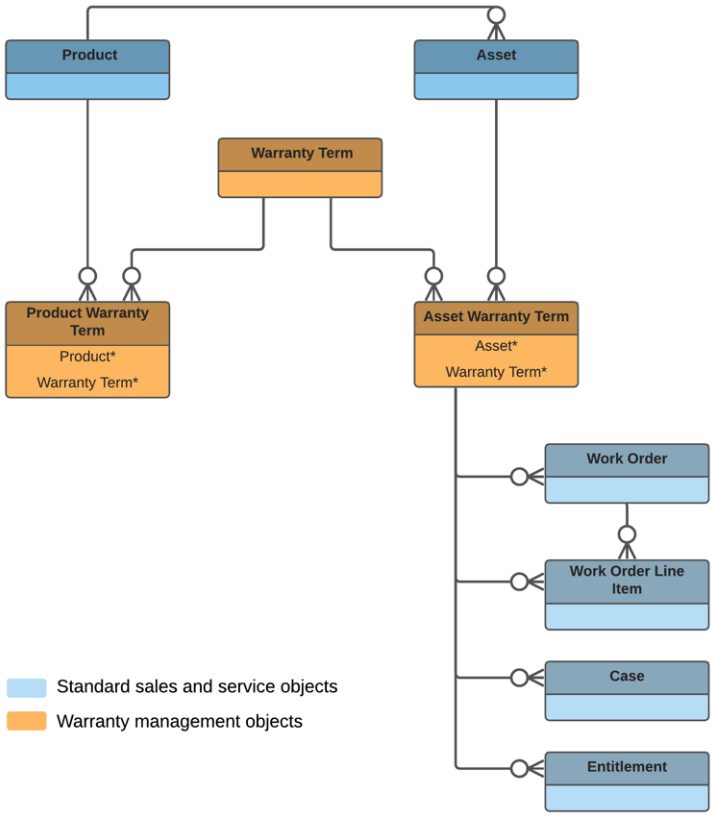


Field Service Warranty Management Data Model

Use warranty items to record details of the labor, parts, and expenses, along with any exchange options, that are provided to rectify issues with products sold or installed. Create standard warranties for products and product families and, for products you install, record details of additional or extended warranties along with exclusions and void terms.

Note: Asterisks in the following diagram signify required fields.

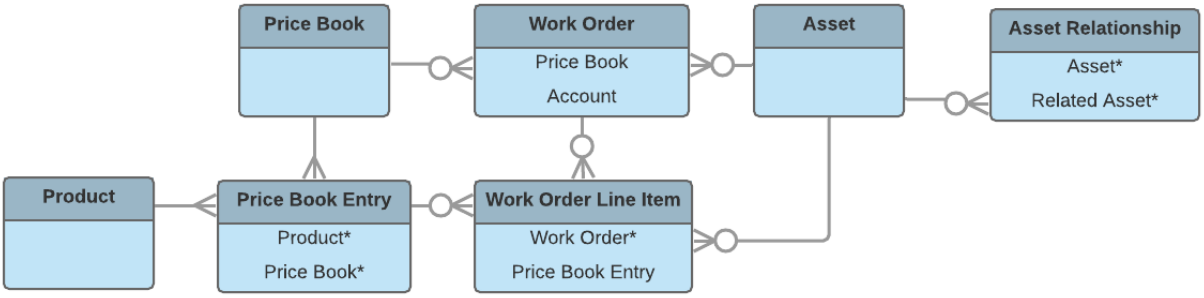
A *warranty term* is linked to a *product warranty term* to define the standard warranty offered for a product or product family. When a product is installed, *asset warranty term* details are created from the standard warranty. An asset warranty term can be associated with a work order, work order line item, case, or entitlement to track actions related to the fulfillment of the warranty term provisions.



Field Service Pricing Data Model

Link work orders to products or assets in your org to track product pricing and work being performed on your customers' installed products.

Note: Asterisks in the following diagram signify required fields.



If you've set up a product catalog in Salesforce to track the goods and services your business offers, you can associate items in your price books with work orders and their line items, similar to the way you can associate products with opportunities or orders. If you specify a *price book* on a *work order*, this allows you to link each *work order line item* to a *price book entry* (product) from the price book. List price, discount, and quantity are defined at the line-item level.

For example, if you create a work order for a solar panel installation, select a price book in the Price Book lookup field on the work order. Then, use the Price Book Entry lookup field on its work order line items to select goods or services listed in your price book, such as Site Assessment, Solar Panel, and Inverter. A quick glance at a completed work order's line items shows you which products from your product catalog were sold as part of the work order.

After a product is purchased and installed for a customer, it is typically tracked as an *asset* in Salesforce. The Asset lookup field on work orders and work order line items allows you to track work being performed on a specific asset. It also makes it possible to view a history of all work completed on the asset.

If an asset is replaced or upgraded, the relationship between the old and new asset is tracked in an *asset relationship* record. An asset relationship lists a start and end time—for instance, if the replacement asset is being leased—and a relationship type, which must be defined by the admin.

SEE ALSO:

Work Order Pricing Guidelines for Field Service

Equal Asset Relationships

Field Service Object References

Object reference for standard and custom Salesforce objects used in Field Service.

IN THIS SECTION:

Field Service Standard and Custom Objects

A list of standard Salesforce objects and Salesforce-managed custom objects used in Field Service.

Field Service Custom Fields on Standard Objects

A list of custom fields on standard Salesforce objects installed with the Field Service managed package.

Supplementary Field Service Objects

A list of Field Service objects that support history tracking or sharing.

Field Service Standard and Custom Objects

A list of standard Salesforce objects and Salesforce-managed custom objects used in Field Service.

The following list links to reference information for the standard objects and Salesforce-managed custom objects. Some objects are specific to Field Service, while others are used across a variety Salesforce features.

Note: Most objects are available only if Field Service is enabled. Objects not tied to Field Service enablement are shown with an asterisk (*).

- Address

  Represents a mailing, billing, or home address.

- ApptBundleAggrDurDnscale

  Sums the duration of the bundle members, reduced by a predefined percentage.

- ApptBundleAggrPolicy

  Policy that defines how the property values of the bundle members are aggregated and assigned to the bundle.

- ApptBundleConfig

Represents the general parameters that define the behavior of the bundle.

- ApptBundlePolicy

  Policy that defines how the bundling of service appointments should be handled.

- ApptBundlePolicySvcTerr

  Represents a link between the BundlePolicy and the ServiceTerritory.

- ApptBundlePropagatePolicy

  Policy that defines which property values are inherited from the bundle to the bundle members or are assigned as constant values in the bundle members.

- ApptBundleRestrictPolicy

  Policy that defines the restrictions that are considered while forming a bundle.

- ApptBundleSortPolicy

  Policy that defines the properties by which the bundle members are sorted within the bundle. Can also be used in the automatic mode for determining the order of the automatic selection of bundle members.

- AppExtension

  Represents a connection between the Field Service mobile app and another app, typically for passing record data to the Salesforce mobile app or other apps.

- Asset*

  Represents an item of commercial value, such as a product sold by your company or a competitor, that a customer has purchased.

- AssetAccountParticipant on page 21

  Represents a junction between the Asset and Account objects describing the association between a participating account and an asset.

- AssetAttribute

  Stores asset attributes to track and analyze asset conditions to improve their uptime.

- AssetContactParticipant on page 23

  Represents a junction between the Asset and Contact objects describing the association between a participating contact and an asset.

- AssetDowntimePeriod*

  Represents a period during which an asset is not able to perform as expected. Downtime periods include planned activities, such as maintenance, and unplanned events, such as mechanical breakdown.

- AssetRelationship*

  Represents a non-hierarchical relationship between assets due to an asset modification; for example, a replacement, upgrade, or other circumstance. In Revenue Lifecycle Management, this object represents an asset or assets grouped in a bundle or set.

- AssetWarranty

  Defines the warranty terms applicable to an asset along with any exclusions and extensions.

- AssignedResource

Represents a service resource who is assigned to a service appointment in Field Service and Lightning Scheduler. Assigned resources appear in the Assigned Resources related list on service appointments.

- AssociatedLocation

  Represents a link between an account and a location in Field Service. You can associate multiple accounts with one location. For example, a shopping center location may have multiple customer accounts.

- AttributeDefinition

  Represents a product, asset, or object attribute, for example, a hardward specification or software detail.

- AttributePicklist

  Represents a custom picklist for an asset attribute.

- AttributePicklistValue

  Represents the values of an asset attribute picklist.

- ContractLineItem*

  Represents a product covered by a service contract (customer support agreement).

- ContractLineOutcome

  Represents information on a contract line outcome's captured data and other related parameters that are used when capturing data.

- ContractLineOutcomeData

  Represents the contract line outcome's captured data. It stores the data that was captured between the contract line outcome's start date and end date.

- DigitalSignature

  Represents a signature captured on a service report in field service.

- Entitlement*

  Represents the customer support an account or contact is eligible to receive.

- EntityMilestone*

  Represents a required step in a customer support process on a work order. The Salesforce user interface uses the term "object milestone.

- Expense

  Represents an expense linked to a work order. Service resource technicians can log expenses, such as tools or travel costs.

- ExpenseReport

  Represents a report that summarizes expenses.

- ExpenseReportEntry

  Represents an entry in an expense report.

- FieldServiceMobileSettings

  Represents a configuration of settings that control the Field Service iOS and Android mobile app experience.

- FldSvcObjChg

  Represents a change made to one of a service appointment's tracked fields.

- FldSvcObjChgDtl

  Represents the details of a change made to one of a service appointment's tracked fields.

- FSL__Time_Dependency__c

  Represents a dependency between two service appointments. This object is used to define scheduling relationships between two appointments. It allows you to determine the order and timing in which dependent appointments should be scheduled.

- GeolocationBasedAction

  Represents a geolocation-based action, which is an action that's triggered when a user enters, exits, or is within the area of the associated object.

- LinkedArticle

  Represents a knowledge article that is attached to a work order, work order line item, or work type.

- Location

  Represents a warehouse, service vehicle, work site, or other element of the region where your team performs field service work. In API version 49.0 and later, you can associate activities with specific locations. Activities, such as the tasks and events related to a location, appear in the activities timeline when you view the location detail page. Also in API version 49.0 and later, Work.com users can view Employees as a related list on Location records. In API version 51.0 and later, this object is available for Omnichannel Inventory and represents physical locations where inventory is available for fulfilling orders.

- MaintenanceAsset

  Represents an asset covered by a maintenance plan in field service. Assets can be associated with multiple maintenance plans.

- MaintenancePlan

  Represents a preventive maintenance schedule for one or more assets in field service.

- MaintenanceWorkRule

  Represents the recurrence pattern for a maintenance record.

- MobileSettingsAssignment

  Represents the assignment of a particular field service mobile settings configuration to a user profile.

- OperatingHours

  Represents the hours in which a service territory, service resource, or account is available for work. OperatingHours is used by Field Service, Salesforce Scheduler, Salesforce Meetings, Sales Engagement, and Workforce Engagement.

- OperatingHoursHoliday

  Represents the day or hours for which a service territory and service resources exclusive to the service territory are unavailable in Field Service, Salesforce Scheduler, Salesforce Meetings, Sales Engagement, or Workforce Engagement.

- Pricebook2*

  Represents a price book that contains the list of products that your org sells.

- Product2*

  Represents a product that your company sells.

- ProductConsumed

  Represents an item from your inventory that was used to complete a work order or work order line item in field service.

- ProductConsumedState

  Represents the status of an item from your inventory that was used to complete a work order or work order line item in Field Service.

- ProductItem

  Represents the stock of a particular product at a particular location in field service, such as all bolts stored in your main warehouse.

- ProductItemTransaction

  Represents an action taken on a product item in field service. Product item transactions are auto-generated records that help you track when a product item is replenished, consumed, or adjusted.

- ProductRequest

  Represents an order for a part or parts in field service.

- ProductRequestLineItem

  Represents a request for a part in field service. Product request line items are components of product requests.

- ProductRequired

  Represents a product that is needed to complete a work order or work order line item in field service.

- ProductServiceCampaign

  Represents a set of activities to be performed on a product service campaign asset, such as a product recall for safety issues or product defects.

- ProductServiceCampaignItem

  Represents a product service campaign's asset.

- ProductServiceCampaignItemStatus

  Represents a status for a product service campaign item in field service.

- ProductServiceCampaignStatus

  Represents a status for a product service campaign in field service.

- ProductTransfer

  Represents the transfer of inventory between locations in field service.

- ProductWarrantyTerm

  Defines the relationship between a product or product family and warranty term.

- RecordsetFilterCriteria

  Represents a set of filters that can be used to match service appointments or assets based on your criteria fields.

- RecordsetFilterCriteriaRule

  Represents a rule using fields from the designated source object to create filters on the filtered, or target, object.

- RecordsetFltrCritMonitor

  Monitors whether the value of an asset attribute is within the threshold of a recordset filter criteria (RFC).

- ResourceAbsence

  Represents a time period in which a service resource is unavailable to work in Field Service, Salesforce Scheduler, or Workforce Engagement.

- ResourcePreference

  Represents an account's preference for a specified service resource on field service work.

- ReturnOrder

  Represents the return or repair of inventory or products in Field Service, or the return of order products in Order Management.

- ReturnOrderLineItem

  Represents a specific product that is returned or repaired as part of a return order in Field service, or a specific order item that is returned as part of a return order in Order Management.

- SerializedProduct on page 25

  Records serial numbers for each individual product in an inventory.

- SerializedProductTransaction

  Represents transactions performed on a serialized product.

- ServiceAppointment

  Represents an appointment to complete work for a customer in Field Service, Lightning Scheduler,Intelligent Appointment Management, and Virtual Care.

- ServiceAppointmentStatus

  Represents a possible status of a service appointment in field service.

- ServiceContract*

  Represents a customer support contract (business agreement).

- ServiceCrew

  Represents a group of service resources who can be assigned to service appointments as a unit.

- ServiceCrewMember

  Represents a technician service resource that belongs to a service crew.

- ServiceReport

  Represents a report that summarizes a work order, work order line item, or service appointment.

- ServiceReportLayout

  Represents a service report template in field service.

- ServiceResource

  Represents a service technician or service crew in Field Service and Salesforce Scheduler, or an agent in Workforce Engagement.

- ServiceResourceCapacity

  Represents the maximum number of scheduled hours or number of service appointments that a capacity-based service resource can complete within a specific time period.

- ServiceResourceSkill

  Represents a skill that a service resource possesses in Field Service and Lightning Scheduler.

- ServiceTerritory

Represents a geographic or functional region in which work can be performed in Field Service, Salesforce Scheduler, or Workforce Engagement.

- ServiceTerritoryLocation

  Represents a location associated with a particular service territory in field service.

- ServiceTerritoryMember

  Represents a service resource who can be assigned in a service territory in Field Service, Salesforce Scheduler, or Workforce Engagement.

- Shift

  Represents a shift for service resource scheduling.

- ShiftPattern

  Represents a pattern of templates for creating shifts.

- ShiftPatternEntry

  ShiftPatternEntry links a shift template to a shift pattern.

- ShiftTemplate

  Represents a template for creating shifts.

- Shipment

  Represents the transport of inventory in field service or a shipment of order items in Order Management.

- Skill*

  Represents a category or group of Chat users or service resources in Field Service or Workforce Engagement.

- SkillRequirement

  Represents a skill that is required to complete a particular task in Field Service, Omni-Channel, Salesforce Scheduler, or Workforce Engagement. Skill requirements can be added to pending service routing objects in Omni-Channel. They can be added to work types, work orders, and work order line items in Field Service and Lightning Scheduler. And they can be added to job profiles in Workforce Engagement.

- TimeSheet

  Represents a schedule of a service resource's time in Field Service or Workforce Engagement.

- TimeSheetEntry

  Represents a span of time that a service resource spends on a field service task.

- TimeSlot

  Represents a period of time on a specified day of the week during which work can be performed in Field Service, Salesforce Scheduler, or Workforce Engagement. Operating hours consist of one or more time slots.

- TravelMode

  Represents a travel mode used for travel time calculations. The records include information about the type of transportation (such as Car or Walking), whether a vehicle can take toll roads, and whether a vehicle is transporting hazardous materials.

- WarrantyTerm

  Represents warranty terms defining the labor, parts, and expenses covered, along with any exchange options, provided to rectify issues with products.

- WorkCapacityAvailability

  Represents the available work capacity for a specific time and service territory.

- WorkCapacityLimit

  Represents the capacity limit in a specific service territory for a workstream or for the whole service territory in a given period.

- WorkCapacityUsage

  Represents the capacity usage in a specific service territory for a workstream or for the whole service territory in a given period.

- WorkOrder*

  Represents field service work to be performed for a customer.

- WorkOrderLineItem*

  Represents a subtask on a work order in field service.

- WorkOrderLineItemStatus

  Represents a possible status of a work order line item in field service.

- WorkPlan

  Represents a work plan for a work order or work order line item.

- WorkPlanSelectionRule

  Represents a rule that selects a work plan for a work order or work order line item.

- WorkPlanTemplate

  Represents a template for a work plan.

- WorkPlanTemplateEntry

  Represents an object that associates a work step template with a work plan template.

- WorkOrderStatus

  Represents a possible status of a work order in field service.

- WorkStep

  Represents a work step in a work plan.

- WorkStepStatus

  Represents a picklist for a status category on a work step.

- WorkStepTemplate

  Represents a template for a work step.

- WorkType

  Represents a type of work to be performed in Field Service and Lightning Scheduler. Work types are templates that can be applied to work order or work order line items.

- WorkTypeGroup

  Represents a grouping of work types used to categorize types of appointments available in Lightning Scheduler, or to define scheduling limits in Field Service.

- WorkTypeGroupMember

Represents the relationship between a work type and the work type group it belongs to.

AssetAccountParticipant

Represents a junction between the Asset and Account objects describing the association between a participating account and an asset. This object is available in API version 56.0 and later.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Fields

`AccountId`

- **Type**: reference
- **Properties**: Create, Filter, Group, Sort, Update
- **Description**: The stakeholder account associated with the asset. This field is a relationship field.
- **Relationship Name**: Account
- **Relationship Type**: Lookup
- **Refers To**: Account

`AssetId`

- **Type**: reference
- **Properties**: Create, Filter, Group, Sort
- **Description**: The asset associated with the account. This field is a relationship field.
- **Relationship Name**: Asset
- **Relationship Type**: Lookup
- **Refers To**: Asset

`EffectiveEndDate`

- **Type**: date
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The date when the association between the stakeholder and the vehicle ended.

`EffectiveStartDate`

- **Type**: date
- **Properties**: Create, Filter, Group, Sort, Update
- **Description**: The date when the association between the stakeholder and the vehicle was initiated.

IsActive

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Indicates whether the association between the stakeholder and the vehicle is active (true) or not (false).
- **Default value**: `false`

LastReferencedDate

- **Type**: dateTime
- **Properties**: Filter, Nillable, Sort
- **Description**: The most recent date on which a user referenced this record.

LastViewedDate

- **Type**: dateTime
- **Properties**: Filter, Nillable, Sort
- **Description**: The most recent date on which a user viewed this record.

Name

- **Type**: string
- **Properties**: Create, Filter, Group, idLookup, Sort, Update
- **Description**: The name of the asset account participant.

RecordTypeId

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The record type associated with the asset account participant. This field is a relationship field.
- **Relationship Name**: RecordType
- **Relationship Type**: Lookup
- **Refers To**: RecordType

StakeholderRole

- **Type**: picklist
- **Properties**: Create, Filter, Group, Restricted picklist, Sort, Update
- **Description**: Specifies the role of the associated account. Possible values are:
- `Customer-Preferred Dealer`
- `Nominated Dealer`
- `Closest Dealer`
- `Sales Dealer`
- `Service Dealer`
- `Customer`

- `Driver`
- `Financier`
- `Owner`

`UsageType`

- **Type**: picklist
- **Properties**: Create, Filter, Group, Restricted picklist, Sort, Update
- **Description**: Specifies the usage type of the asset account participant. Possible values are:
- `Automotive`
- `FieldServiceLightning`—Field Service Lightning

`VehicleId`

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The vehicle that's marked as an asset. This field is a relationship field.
- **Relationship Name**: Vehicle
- **Relationship Type**: Lookup
- **Refers To**: Vehicle

Associated Objects

This object has the following associated objects. If the API version isn't specified, they're available in the same API versions as this object. Otherwise, they're available in the specified API version and later.

- AssetAccountParticipantChangeEvent: Change events are available for the object.
- AssetAccountParticipantFeed: Feed tracking is available for the object.
- AssetAccountParticipantHistory: History is available for tracked fields of the object.

AssetContactParticipant

Represents a junction between the Asset and Contact objects describing the association between a participating contact and an asset. This object is available in API version 56.0 and later.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Fields

`AssetId`

- **Type**: reference
- **Properties**: Create, Filter, Group, Sort
- **Description**: The asset associated with the contact. This field is a relationship field.

- **Relationship Name**: Asset
- **Relationship Type**: Lookup
- **Refers To**: Asset

`ContactId`

- **Type**: reference
- **Properties**: Create, Filter, Group, Sort, Update
- **Description**: The contact associated with the asset. This field is a relationship field.
- **Relationship Name**: Contact
- **Relationship Type**: Lookup
- **Refers To**: Contact

`EffectiveEndDate`

- **Type**: date
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The date when the association between the stakeholder and the vehicle ended.

`EffectiveStartDate`

- **Type**: date
- **Properties**: Create, Filter, Group, Sort, Update
- **Description**: The date when the association between the stakeholder and the vehicle was initiated.

`IsActive`

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Indicates whether the association between the stakeholder and the vehicle is active (true) or not (false).
- **Default value**: `false`

`LastReferencedDate`

- **Type**: dateTime
- **Properties**: Filter, Nillable, Sort
- **Description**: The most recent date on which a user referenced this record.

`LastViewedDate`

- **Type**: dateTime
- **Properties**: Filter, Nillable, Sort
- **Description**: The most recent date on which a user viewed this record.

`Name`

- **Type**: string
- **Properties**: Create, Filter, Group, idLookup, Sort, Update
- **Description**: The name of the asset contact participant.

`StakeholderRole`

- **Type**: picklist
- **Properties**: Create, Filter, Group, Sort, Update
- **Description**: Specifies the role of the associated contact.

`UsageType`

- **Type**: picklist
- **Properties**: Create, Filter, Group, Restricted picklist, Sort, Update
- **Description**: Specifies the usage type of the asset contact participant. Possible values are:
- `Automotive`
- `FieldServiceLightning`—Field Service Lightning

`VehicleId`

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The vehicle that's marked as an asset. This field is a relationship field.
- **Relationship Name**: Vehicle
- **Relationship Type**: Lookup
- **Refers To**: Vehicle

Associated Objects

This object has the following associated objects. If the API version isn't specified, they're available in the same API versions as this object. Otherwise, they're available in the specified API version and later.

- AssetContactParticipantChangeEvent: Change events are available for the object.
- AssetContactParticipantFeed: Feed tracking is available for the object.
- AssetContactParticipantHistory: History is available for tracked fields of the object.

SerializedProduct

Records serial numbers for each individual product in an inventory. This object is available in API version 50.0 and later.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Special Access Rules

To use this object, you need the Manage Industries Visit permission and the Industries Visit permission set license. This rule doesn't apply to using serialized products in Field Service.

Fields

AssetId

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: Reference to an asset record.

ExpirationDate

- **Type**: date
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The date the product expires.

LastReferencedDate

- **Type**: dateTime
- **Properties**: Filter, Nillable, Sort
- **Description**: The last date and time this record was referenced.

LastViewedDate

- **Type**: dateTime
- **Properties**: Filter, Nillable, Sort
- **Description**: The last date and time this record was viewed.

Name

- **Type**: string
- **Properties**: Autonumber, Defaulted on create, Filter, idLookup, Sort
- **Description**: The name of the serialized product record. This is auto-generated.

OwnerId

- **Type**: reference
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Reference to the owner of the record.

Product2Id

- **Type**: reference
- **Properties**: Create, Filter, Group, Sort, Update

- **Description**: The product that is serialized.

ProductItemId

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: Reference to the inventory, a product item record, this serialized product belongs to.

SerialNumber

- **Type**: string
- **Properties**: Create, Filter, Group, Sort, Update
- **Description**: The serial number of the product.

Status

- **Type**: picklist
- **Properties**: Create, Defaulted on create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The status of the product. Possible values include:
  - `Available`
  - `Consumed`
  - `Damaged`
  - `Lost`
  - `Sent`
- **Default value**: `Available`

Field Service Custom Fields on Standard Objects

A list of custom fields on standard Salesforce objects installed with the Field Service managed package.

See the Field Service Apex Reference section for more API references related to the managed package.

IN THIS SECTION:

AssignedResource Custom Fields
Custom fields associated with a service resource who is assigned to a service appointment in Field Service. Assigned resources appear in the Assigned Resources related list on service appointments. This object is available in API version 38.0 and later.

ResourceAbsence Custom Fields
Custom fields associated with a time period in which a service resource is unavailable to work in Field Service.

ServiceAppointment Custom Fields
Custom fields associated with an appointment to complete work for a customer in Field Service.

ServiceResource Custom Fields
Custom fields associated with a field service technician or crew in Field Service.

ServiceResourceCapacity Custom Fields

Custom fields associated with the maximum number of scheduled hours or number of service appointments that a capacity-based service resource can complete within a specific time period.

ServiceTerritory Custom Fields

Custom fields associated with a geographic or functional region in which field service work can be performed in Field Service.

TimeSlot Custom Fields

Custom fields associated with a period of time on a specified day of the week during which field service work can be performed in Field Service. Operating hours consist of one or more time slots.

WorkOrder Custom Fields

Custom fields associated with field service work to be performed for a customer.

WorkOrderLineItem Custom Fields

Custom fields associated with a subtask on a work order in field service.

AssignedResource Custom Fields

Custom fields associated with a service resource who is assigned to a service appointment in Field Service. Assigned resources appear in the Assigned Resources related list on service appointments. This object is available in API version 38.0 and later.

The standard fields are documented in the AssignedResource object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `undelete()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

AssignedResource

Fields

FSL__EstimatedTravelDistanceFrom__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The estimated travel distance from the service resource's assigned appointment location to their home base. For this field, the assigned appointment location must have latitude and longitude coordinates and be the last location of the day. If it doesn't have coordinates, isn't the last location of the day, or the travel calculations are disabled, this field is 0.

FSL__EstimatedTravelDistanceTo__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update

- **Description**: The estimated travel distance to the service resource's assigned appointment location from a service appointment, another resource absence location, or their home base. For this field, the assigned appointment location must have latitude and longitude coordinate. If it doesn't have latitude and longitude coordinates or the travel calculations are disabled, this field is 0.

FSL__EstimatedTravelTimeFrom__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The estimated travel time from the service resource's assigned appointment location to their home base. For this field, the assigned appointment location must have latitude and longitude coordinates and be the last location of the day. If it doesn't have coordinates, isn't the last location of the day, or the travel calculations are disabled, this field is 0.

FSL__Estimated_Travel_Time_From_Source__c

- **Type**: picklist
- **Properties**: Create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The method used to calculate the travel time from the service resource's assigned appointment location to their home base. For this field, the assigned appointment location must have latitude and longitude coordinates and be the last location of the day. If it doesn't have coordinates, isn't the last location of the day, or the travel calculations are disabled, this field is `None`.
- **Possible values**:
  - `Aerial`
  - `None`
  - `Predictive`
  - `SLR`

FSL__Estimated_Travel_Time_To_Source__c

- **Type**: picklist
- **Properties**: Create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The method used to calculate the travel time to the service resource's assigned appointment location from a service appointment, another resource absence location, or their home base. For this field, the assigned appointment location must have latitude and longitude coordinates. If it doesn't have latitude and longitude coordinates or the travel calculations are disabled, this field is `None`.
- **Possible values**:
  - `Aerial`
  - `None`
  - `Predictive`
  - `SLR`

Internal Fields

These internal fields are used by the Field Service managed package. Although they're publicly accessible, they must only be updated by the managed package.

- FSL__Last_Updated_Epoch__c—Used to prevent the overlapping of multiple concurrent scheduling requests.
- FSL__UpdatedByOptimization__c—Equals *`true`* if the record was updated by the optimization engine.

- FSL__calculated_duration__c—Indicates the duration (start to end time) of the service appointment assigned to the resource in minutes.

ResourceAbsence Custom Fields

Custom fields associated with a time period in which a service resource is unavailable to work in Field Service.

The standard fields are documented in the ResourceAbsence object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

Salesforce Object Reference: ResourceAbsence

Salesforce Help: View Resource Absences on the Gantt and Map

Fields

FSL__Approved__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: When this field is `true` and the **Activate approval confirmation on resource absences** setting is enabled, the resource absence appears in the dispatcher console's Gantt chart and is considered by the scheduler. When this field is `false` and the setting is enabled, the resource absence doesn't appear in the Gantt chart and is ignored by the scheduler.

  For more info, view the View Resource Absences on the Gantt and Map Help article.
- **Default value**: `false`

FSL__Duration_In_Minutes__c

- **Type**: double
- **Properties**: Filter, Nillable, Sort
- **Description**: The duration of the resource absence in minutes. This field is automatically populated after the resource absence is created. This is a calculated field.
- **Formula**:

```
IF (ISBLANK(Start), 0, (End - Start)*24*60)
```

FSL__EstTravelTimeFrom__c

- **Type**: double

- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The estimated travel time from the service resource's absence location to their home base. For this field, the absence location must have latitude and longitude coordinates and be the last location of the day. If it doesn't have coordinates, isn't the last location of the day, or the travel calculations are disabled, this field is 0.

FSL__EstTravelTime__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The estimated time to the service resource's absence location from a service appointment or another resource absence location. For this field, the absence location must have latitude and longitude coordinates. If it doesn't have latitude and longitude coordinates or the travel calculations are disabled, this field is 0.

FSL__EstimatedTravelDistanceFrom__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The estimated travel distance from the service resource's absence location to their home base. For this field, the absence location must have latitude and longitude coordinates and be the last location of the day. If it doesn't have coordinates, isn't the last location of the day, or the travel calculations are disabled, this field is 0.

FSL__EstimatedTravelDistanceTo__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The estimated travel distance to service resource's absence location from a service appointment or another resource absence location. For this field, the absence location must have latitude and longitude coordinate. If it doesn't have latitude and longitude coordinates or the travel calculations are disabled, this field is 0.

FSL__Estimated_Travel_Time_From_Source__c

- **Type**: picklist
- **Properties**: Create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The method used to calculate the travel time from the service resource's absence location to a service appointment or another resource absence location. For this field, the absence location must have latitude and longitude coordinates and be the last location of the day. If it doesn't have coordinates, isn't the last location of the day, or the travel calculations are disabled, this field is `None`.
- **Possible values**:
  - `Aerial`
  - `None`
  - `Predictive`
  - `SLR`

FSL__Estimated_Travel_Time_To_Source__c

- **Type**: picklist

- **Properties**: Create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The method used to calculate the travel time to this service resource's absence location from a service appointment or another resource absence location. For this field, the absence location must have latitude and longitude coordinates. If it doesn't have latitude and longitude coordinates or the travel calculations are disabled, this field is None.
- **Possible values**:
  - `Aerial`
  - `None`
  - `Predictive`
  - `SLR`

FSL__GanttLabel__c

- **Type**: string
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The label of the resource absence in the Field Service dispatcher console's Gantt chart and resource calendar. This replaces the resource absence's number in the chart.

FSL__Gantt_Color__c

- **Type**: string
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The Hex color of the resource absence in Field Service dispatcher console's Gantt chart and resource calendar.

FSL__Scheduling_Policy_Used__c

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: A scheduling policy used by the scheduler for the service appointment. This policy overrides the default one in the Field Service Settings page or the one that the scheduler would otherwise select. This is used for travel calculations.

  This is a relationship field.
- **Relationship Name**: FSL__Scheduling_Policy_Used__r
- **Relationship Type**: Lookup
- **Refers To**: FSL__Scheduling_Policy__c

Internal Fields

These internal fields are used by the Field Service managed package for Street Level Routing calculations. Although they're publicly accessible, they must only be updated by the managed package.

- FSL__InternalSLRGeolocation__Latitude__s
- FSL__InternalSLRGeolocation__Longitude__s
- FSL__InternalSLRGeolocation__c

ServiceAppointment Custom Fields

Custom fields associated with an appointment to complete work for a customer in Field Service.

The standard fields are documented in the ServiceAppointment object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

- Salesforce Object Reference: ServiceAppointment
- Salesforce Help: Schedule an Appointment Automatically
- Salesforce Help: Create Custom Appointment Icons
- Salesforce Help: Control Which Appointments Appear in the Dispatcher Console
- Salesforce Help: Fill Schedule Gaps
- Salesforce Help: Enable Multiday Service Appointments
- Salesforce Help: Schedule Appointments Using Priorities

Fields

FSL__Appointment_Grade__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The appointment grade of the scheduled appointment using the Appointment Booking feature.

FSL__Auto_Schedule__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Indicates if the appointment is created and scheduled in the same action.

  For more info, view the Schedule an Appointment Automatically Help article.
- **Default value**: `false`

FSL__Duration_In_Minutes__c

- **Type**: double
- **Properties**: Filter, Nillable, Sort
- **Description**: The duration in minutes of the scheduled appointment. It calculates the time between the scheduled start and end times in minutes. This is a calculated field.

- **Formula**:

```
IF (ISBLANK(SchedStartTime), 0,
(SchedEndTime - SchedStartTime)*24*60)
```

FSL__Emergency__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Indicates if the Emergency Wizard global action schedules the appointment. If true, the service appointment has an emergency icon in the Field Service dispatcher console's Gantt chart.
- **Default value**: `false`

FSL__GanttColor__c

- **Type**: string
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The Hex color of the service appointment in the Field Service dispatcher console's Gantt chart and the resource calendar.

FSL__GanttIcon__c

- **Type**: textarea
- **Properties**: Create, Nillable, Update
- **Description**: A custom icon for the service appointment that appears in the Gantt chart, map, and appointment list. This helps dispatchers quickly identify appointment characteristics. For example, use a custom icon to indicate that an appointment is for a VIP or first-time customer. The format is a URL ending in an image suffix, such as .png or .gif. The image is scaled to 16 x 16 pixels.

  For more info, view the Create Custom Appointment Icons Help article.

FSL__GanttLabel__c

- **Type**: string
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The label of the scheduled service appointment in the Field Service dispatcher console's Gantt chart. This replaces the service appointment number in the chart.

FSL__Gantt_Display_Date__c

- **Type**: dateTime
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The Gantt Display Date filter in the date field dropdown menu in the Field Service dispatcher console to control which appointments are visible in the appointment list. When a service appointment's Gantt Display Date falls within the Gantt time frame, the appointment is visible on the Gantt. For example, if a maintenance appointment must be completed within the next six months, you can set the date so that you see it on the Gantt every day. You can set up this field to update an important appointment's Gantt Display Date to today's date on a daily basis.

  For more info, view the Control Which Appointments Appear in the Dispatcher Console Help article.

FSL__InJeopardyReason__c

- **Type**: picklist
- **Properties**: Create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The reason for when the service appointment is in jeopardy. Use this field only when the `FSL__InJeopardy__c` status is `true`. You can add custom picklist values.
- **Possible values**:
  - `Delayed Finish`
  - `Delayed Start`
  - `Due Date Approaching`
  - `No Response`
  - `Rejected by Contractor`

FSL__InJeopardy__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if a service appointment is in jeopardy. This helps dispatchers gain visibility to service appointments at risk. A user can manually set the service appointment status to **In Jeopardy** or this can be done automatically using, for example, process builders or triggers.
- **Default value**: `false`

FSL__IsFillInCandidate__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if this service appointment is considered as a candidate when filling in a schedule with the Fill-In Schedule feature. If a service appointment's parent record is a work order or work order line item, the parent record's `FSL__IsFillInCandidate__c` field must also be set to `true` for the appointment to be a candidate. Alternatively, instead of using this field, you can create a custom checkbox field, including formula fields, to evaluate whether this appointment is considered as a candidate. This can be done through the Field Service Settings page

  For more info, view the Fill Schedule Gaps Help article.

- **Default value**: `true`

FSL__IsMultiDay__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if this service appointment spans over multiple days. Alternatively, instead of using this field, you can create a checkbox formula field through the Field Service Settings page to evaluate whether it spans over multiple days or not.

  For more info, view the Enable Multiday Service Appointments Help article.

- **Default value**: `false`

FSL__Pinned__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if this service appointment is pinned to the Field Service dispatcher console's Gantt chart. Pinned service appointments can't be manually dragged or automatically scheduled by any scheduling operation. Pinned service appointments have a lock icon in the Field Service dispatcher console's Gantt chart.
- **Default value**: `false`

FSL__Prevent_Geocoding_For_Chatter_Actions__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if the service appointment's BeforeUpdate Platform Apex trigger disables the Chatter Actions's geolocation cleanup on address change. When this field is set to `true`, it prevents Chatter Actions to geocode the address and waits until Field Service does it after the address changes. This field is set to `false` after the cleanup completes.
- **Default value**: `false`

FSL__Schedule_Mode__c

- **Type**: picklist
- **Properties**: Create, Defaulted on create, Filter, Group, Nillable, Unrestricted picklist, Sort, Update
- **Description**: The type of the scheduling operation. For example, when not using Enhanced Scheduling and Optimization, if the service appointment is scheduled using drag and drop, the value is `Manual`. If the service appointment is scheduled using the Appointment Booking feature, the value is `Automatic`. When using Enhanced Scheduling and Optimization, if the service appointment is scheduled using drag and drop, the value is `Drag and Drop`. If the service appointment is scheduled using the Appointment Booking feature, the value is `Schedule`. This field is populated by the system. Don't edit this field.

  Additional values for each scheduling operation, available only with Enhanced Scheduling and Optimization:

  - `Drag and Drop`
  - `Schedule`
  - `Global Optimization`
  - `In-Day Optimization`
  - `Resource Optimization`
- **Default value**: `None`
- **Possible values**:
  - `Automatic`
  - `Manual`
  - `None`
  - `Optimization`

FSL__Schedule_over_lower_priority_appointment__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update

- **Description**: Determines whether you can schedule critical service appointments over lower priority appointments.

  For more info, view the Schedule Appointments Using Priorities Help article.

- **Default value**: `false`

FSL__Scheduling_Policy_Used__c

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: A scheduling policy used by the scheduler for the service appointment. If you edit this field, the policy overrides the default one on the Field Service Settings page. If this field is empty, the field populates with the policy used by the scheduler after the service appointment gets scheduled.

  This is a relationship field.

- **Relationship Name**: FSL__Scheduling_Policy_Used__r
- **Relationship Type**: Lookup
- **Refers To**: FSL__Scheduling_Policy__c

FSL__UpdatedByOptimization__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if the service appointment is updated by the optimizer. This field is populated by the system. Don't edit this field.
- **Default value**: `false`

FSL__Use_Async_Logic__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if the scheduling features related to the service appointment run asynchronously. If you use UI features, such as the Appointment Booking global action, the managed package takes care of this async response for you.
- **Default value**: `false`

FSL__Virtual_Service_For_Chatter_Action__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if the service appointment is a candidate or dummy appointment. If the value is `true`, the appointment is ignored by your custom triggers. This field is populated by the system. Don't edit this field.
- **Default value**: `false`

Internal Fields

These internal fields are used by the Field Service managed package for Street Level Routing calculations. Although they're publicly accessible, they must only be updated by the managed package.

- FSL__InternalSLRGeolocation__Latitude__s

- FSL__InternalSLRGeolocation__Longitude__s
- FSL__InternalSLRGeolocation__c

ServiceResource Custom Fields

Custom fields associated with a field service technician or crew in Field Service.

The standard fields are documented in the ServiceResource object reference.

Supported Calls

`create()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

Salesforce Object Reference: ServiceResource

Salesforce Help: Estimate a Service Resource's Efficiency

Fields

FSL__Efficiency__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The efficiency score or the work pace of the service resource. Enter a value from 0.1 to 10. An efficiency of 1 (default) means that the mobile worker works at a typical or average speed. An efficiency greater than 1 means that the mobile worker works faster than average. Less than 1 means that the mobile worker works slower than average.

  For more info, view the Estimate a Service Resource's Efficiency Help article.

FSL__GanttLabel__c

- **Type**: string
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The description of the service resource in the Field Service dispatcher console's Gantt chart. This is shown under the service resource's name in the chart.

FSL__Online_Offset__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The offset of how long the service resource is considered online since they last used or logged into the mobile app. This overrides the default value in the Field Service Settings page.

FSL__Picture_Link__c

- **Type**: url
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The URL link to the customer's picture used as the avatar in the Field Service dispatcher console's Gantt chart. If no URL is provided here, the Gantt chart uses the user avatar.

FSL__Priority__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The priority of the service resource used to rank their appointments. The lower the number the higher the priority.

FSL__Travel_Speed__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The average aerial travel speed of the service resource used to calculate the aerial travel time. This field overrides the default value in the Field Service Settings page. The units, selected in the Field Service Settings page, are KPH or MPH. Street level routing and predictive travel calculations don't use this field. They have their own settings.

ServiceResourceCapacity Custom Fields

Custom fields associated with the maximum number of scheduled hours or number of service appointments that a capacity-based service resource can complete within a specific time period.

The standard fields are documented in the ServiceResourceCapacity object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

Salesforce Object Reference: ServiceResourceCapacity

Fields

FSL__HoursInUse__c

- **Type**: double
- **Properties**: Filter, Nillable, Sort
- **Description**: The total number of hours of scheduled services occupied by the service resource. This is used by the Field Service dispatcher console to show how much resource capacity is used up and by the scheduler to prevent offering resources whose

capacity is full. This field is updated when the scheduler runs and updates the `FSL__MinutesUsed__c` field. This is a calculated field.

- **Formula**:

```
FSL__MinutesUsed__c / 60
```

FSL__Last_Updated_Epoch__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The date and time, in Epoch format, that the Capacity object was last updated. The Capacity object is updated when a service is scheduled on a capacity-based resource. This field is used by the Field Service dispatcher console.

FSL__MinutesUsed__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The total number of minutes of scheduled services occupies by the service resource. This is used by the Field Service dispatcher console to show how much resource capacity is used up and by the scheduler to prevent offering resources whose capacity is full.

FSL__Work_Items_Allocated__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The number of scheduled service appointments that fill the capacity. This is used by the Field Service dispatcher console to show how much resource capacity is used up.

ServiceTerritory Custom Fields

Custom fields associated with a geographic or functional region in which field service work can be performed in Field Service.

The standard fields are documented in the ServiceTerritory object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

Salesforce Object Reference: ServiceTerritory

Salesforce Help: Drip Feed Service Appointments

Fields

FSL__Hide_Emergency_Map__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Controls if the Emergency Wizard map used by the Emergency Wizard global action is hidden for a service territory. This is for privacy purposes. If this is `true`, it shows a list of appointments with estimated time of arrivals instead.
- **Default value**: `false`

FSL__NumberOfServicesToDripFeed__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The drip feed rate to dispatch appointments. This is part of the drip feed dispatching feature. This value overrides the default value in the Field Service Settings page.

  For more info, view the Drip Feed Service Appointments Help article.

FSL__System_Jobs__c

- **Type**: multipicklist
- **Properties**: Create, Filter, Nillable, Update
- **Description**: The list of automators for scheduling jobs associated with a single territory. Possible values are the default or custom automator names configured in the Field Service Settings page. For example, if you create an optimization automator for Los Angeles called "LA_Optimize_1", this field is populated with`LA_Optimize_1` for the LA service territory. This field is populated by the system. Don't edit this field.for the LA service territory. This field is populated by the system. Don't edit this field.

FSL__TerritoryLevel__c

- **Type**: double
- **Properties**: Create, Filter, Nillable, Sort, Update
- **Description**: The territory hierarchy level of the polygon defining the service territory. A polygon is a custom shape drawn on the map to define the area of the territory. It can be nested inside another polygon creating a hierarchy. This field is populated by the system. Don't edit this field.

Internal Fields

These internal fields are used by the Field Service managed package for Street Level Routing calculations. Although they're publicly accessible, they must only be updated by the managed package.

- FSL__Internal_SLRGeolocation__Latitude__s
- FSL__Internal_SLRGeolocation__Longitude__s
- FSL__Internal_SLRGeolocation__c

TimeSlot Custom Fields

Custom fields associated with a period of time on a specified day of the week during which field service work can be performed in Field Service. Operating hours consist of one or more time slots.

The standard fields are documented in the TimeSlot object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

Salesforce Object Reference: TimeSlot

Fields

FSL__Designated_Work_Boolean_Fields__c

- **Type**: multipicklist
- **Properties**: Create, Filter, Nillable, Update
- **Description**: The type of designated work time slot. To convert a regular time slot to a designated one, use the calendar editor in the Visualforce page instead of editing this field manually in the record page.

  Additional values are added by the managed package when a time slot is converted to a designated work type. The values are the API names of the designated work boolean fields on the service appointment.
- **Possible values**:
  - `None`—Default

FSL__Slot_Color__c

- **Type**: picklist
- **Properties**: Create, Filter, Group, Nillable, Restricted picklist, Sort, Update
- **Description**: The time slot color as it appears on the calendar editor in the Visualforce page. This field is populated by the system.
- **Possible values**:
  - `Amber`
  - `Asphalt`
  - `Black`
  - `Blue`
  - `Brown`
  - `Cyan`
  - `Green`
  - `Grey`
  - `Indigo`
  - `Lime`

- `Orange`
- `Pink`
- `Purple`
- `Red`
- `Teal`
- `Yellow`

WorkOrder Custom Fields

Custom fields associated with field service work to be performed for a customer.

The standard fields are documented in the WorkOrder object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

Salesforce Object Reference: WorkOrder

Salesforce Help: Fill Schedule Gaps

Salesforce Help: Schedule Appointments Using Priorities

Salesforce Help: Work Rule Type: Service Appointment Visiting Hours

Fields

FSL__IsFillInCandidate__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if this work order is considered as a candidate when filling in a schedule with the Fill-In Schedule feature. If a service appointment's parent record is a work order, this field must also be set to true for the appointment to be a candidate. Alternatively, you can create a custom checkbox field through the Field Service Settings page, instead of using this field, to evaluate whether this appointment is considered as a candidate. The custom checkbox field includes formula fields.

  For more info, view the Fill Schedule Gaps Help article.

- **Default value**: `true`

FSL__Prevent_Geocoding_For_Chatter_Actions__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update

- **Description**: Determines if the work order's BeforeUpdate trigger disables the Chatter Action's geolocation cleanup on address change. This field is set to `false` after the cleanup completes.
- **Default value**: `false`

FSL__Scheduling_Priority__c

- **Type**: double
- **Properties**: Filter, Nillable, Sort
- **Description**: The work order priority. The lower the value, the higher the priority.

  For more info, view the Schedule Appointments Using Priorities Help article.

  This is a calculated field.

- **Formula**:

```
CASE(TEXT(Priority), 'Critical' , 1 ,
'High' ,2,'Medium' ,3 ,
'Low', 4, null)
```

FSL__VisitingHours__c

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The visiting hours that define when service appointments associated with the work order can be scheduled. The visiting hours are enforced as long as the Visiting Hours work rule complies with the scheduling policy. Visiting hours are enforced only if the Visiting Hours work rule is part of the scheduling policy.

  For more info, view the Work Rule Type: Service Appointment Visiting Hours Help article.

  This is a relationship field.

- **Relationship Name**: FSL__VisitingHours__r
- **Relationship Type**: Lookup
- **Refers To**: OperatingHours

WorkOrderLineItem Custom Fields

Custom fields associated with a subtask on a work order in field service.

The standard fields are documented in the WorkOrderLineItem object reference.

Supported Calls

`create()`, `delete()`, `describeLayout()`, `describeSObjects()`, `getDeleted()`, `getUpdated()`, `query()`, `retrieve()`, `search()`, `undelete()`, `update()`, `upsert()`

Special Access Rules

Field Service managed package must be installed.

SEE ALSO:

- Salesforce Object Reference: WorkOrderLineItem
- Salesforce Help: Fill Schedule Gaps
- Salesforce Help: Work Rule Type: Service Appointment Visiting Hours

Fields

FSL__IsFillInCandidate__c

- **Type**: boolean
- **Properties**: Create, Defaulted on create, Filter, Group, Sort, Update
- **Description**: Determines if this work order line item is considered as a candidate when filling in a schedule with the Fill-In Schedule feature. If a service appointment's parent record is a work order line item, this field must also be set to true for the appointment to be a candidate. Alternatively, you can create a custom checkbox field through the Field Service Settings page, instead of using this field, to evaluate whether this appointment is considered as a candidate. The custom checkbox field includes formula fields.

  For more info, view the Fill Schedule Gaps Help article.
- **Default value**: `true`

FSL__VisitingHours__c

- **Type**: reference
- **Properties**: Create, Filter, Group, Nillable, Sort, Update
- **Description**: The visiting hours that define when service appointments associated with the work order line item can be scheduled. Visiting hours are enforced only if the Visiting Hours work rule is part of the scheduling policy.

  For more info, view the Work Rule Type: Service Appointment Visiting Hours Help article.

  This is a relationship field.
- **Relationship Name**: FSL__VisitingHours__r
- **Relationship Type**: Lookup
- **Refers To**: OperatingHours

Supplementary Field Service Objects

A list of Field Service objects that support history tracking or sharing.

The following list links to reference information for these supplementary objects.

Note: Most objects are available only if Field Service is enabled. Objects not tied to Field Service enablement are shown with an asterisk (*).

- AssetOwnerSharingRule*
- AssetShare*
- LinkedArticleHistory

- MaintenanceWorkRuleFeed
- MaintenanceWorkRuleHistory
- MaintenanceWorkRuleOwnerSharingRule
- MaintenanceWorkRuleShare
- OperatingHoursHistory
- ProductRequestHistory
- ProductRequestOwnerSharingRule
- ProductRequestShare
- ProductServiceCampaignFeed
- ProductServiceCampaignHistory
- ProductServiceCampaignOwnerSharingRule
- ProductServiceCampaignShare
- ProductServiceCampaignItemFeed
- ProductServiceCampaignItemHistory
- ProductServiceCampaignItemOwnerSharingRule
- ProductServiceCampaignItemShare
- ProductTransferHistory
- ProductTransferOwnerSharingRule
- ProductTransferShare
- ResourceAbsenceHistory
- ResourcePreferenceHistory
- ReturnOrderHistory
- ReturnOrderLineItemHistory
- ReturnOrderOwnerSharingRule
- ReturnOrderShare
- ServiceAppointmentHistory
- ServiceAppointmentOwnerSharingRule
- ServiceAppointmentShare
- ServiceCrewHistory
- ServiceCrewMemberHistory
- ServiceCrewOwnerSharingRule
- ServiceCrewShare
- ServiceResourceCapacityHistory
- ServiceResourceHistory
- ServiceResourceOwnerSharingRule
- ServiceResourceShare
- ServiceResourceSkillHistory
- ServiceTerritoryHistory
- ServiceTerritoryMemberHistory
- SkillRequirementHistory

- TimeSheetEntryHistory
- TimeSheetHistory
- TimeSheetOwnerSharingRule
- TimeSheetShare
- TimeSlotHistory
- WorkOrderHistory*
- WorkOrderLineItemHistory*
- WorkOrderShare*
- WorkTypeGroupHistory
- WorkTypeGroupMemberHistory
- WorkTypeGroupShare
- WorkTypeHistory
- WorkTypeOwnerSharingRule
- WorkTypeShare

Field Service API

The REST and Connect REST endpoints mentioned here can be used to manage your field service implementation.

IN THIS SECTION:

Field Service Flow

Returns information corresponding to a field service flow. This resource is available in REST API version 42.0 and later.

Field Service Mobile Settings

Returns information about the Field Service mobile app settings assigned to the profile of the context user. This resource is available in REST API version 42.0 and later.

Service Report Template

Returns information corresponding to one or more service report templates in field service. This resource is available in REST API version 40.0 and later.

Appointment Bundling REST APIs

Use the Appointment Bundling REST APIs to create bundles, remove bundle members, start the automated process for creating bundles, unbundle one or more bundles, and update a bundle.

REST Reference

You can access Field Service APIs using REST endpoints. These REST APIs follow similar conventions as Connect REST APIs.

Field Service Flow

Returns information corresponding to a field service flow. This resource is available in REST API version 42.0 and later.

This resource accepts the API name of a flow definition and returns information about the active flow version. If no flow version is activated, the most recent version is returned. Flow elements can be translated to the language of the user invoking this API by using Translation Workbench. If a particular flow version's elements have been translated, the data returned uses the language of the query. To learn more, see Flow Components in Translation Workbench.

This resource is available to API users with Field Service mobile app user permissions and the Run Flows user permission.

Syntax

URI: `/services/data/vXX.X/support/fieldservice/Flow?developerNames=FLOW_UNIQUE_NAME`

Where `FLOW_UNIQUE_NAME` is a variable.

Formats: JSON, XML

HTTP Method: GET

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Parameters:

Parameter	Description
`developerNames`	The unique name for the flow definition. Currently, you can specify only one unique name using this parameter.

Response Body

The following table describes the main elements of the response body.

Field	Type	Description
`Flows`	Array	List of flows.
`CreatedDate`	String	Date the flow was created.
`Description`	String	Description of the flow.
`FullName`	String	Full name for the flow.
`Id`	String	Unique ID for the flow.
`LastModifiedDate`	String	Date the flow was last modified.
`Metadata`		Flow metadata.
`Choices`	Array	List of choices used by fields.
`ChoiceText`	String	Text of choice.
`DataType`	String	Data type of choice.
`Name`	String	Name for the choice.
`Screens`	Array of objects	List of screens for the flow.
`AllowBack`	Boolean	Whether you can go back from the screen.
`AllowFinish`	Boolean	Whether you can finish the flow from the screen.
`AllowPause`	Boolean	Whether you can pause from the screen.
`Fields`	Array of objects	List of fields used by the screen.
`HelpText`	String	Help text for the screen.

Field	Type	Description
`Label`	String	Label for the screen.
`LocationX`	Number	X coordinate for the screen.
`LocationY`	Number	Y coordinate for the screen.
`Name`	String	Name of the screen.
`ProcessType`	String	Type of process for the flow; in this case, it should be "FieldServiceMobile".
`Status`	String	Status of the flow.
`VersionNumber`	Number	Version number for the flow.

Example

This sample request uses the unique name "FS_Flow" for the flow definition.

The response contains two screens.

Screen 1 contains two fields:

- Field 1: a text field
- RadioButton: a field with two choices

Screen 2 contains one field:

- Field 2: a text field

Request:

```
GET
/services/data/v42.0/support/fieldservice/Flow?developerNames=FS_Flow
```

Response:


```
{
  "flows" : [ {
    "Id" : "301R000000008grIAA",
    "DefinitionId" : "300R00000004OLFIA2",
    "MasterLabel" : "FS_Flow",
    "ManageableState" : "unmanaged",
    "VersionNumber" : 1,
    "Status" : "Draft",
    "Description" : "FS Flow",
    "ProcessType" : "FieldServiceMobile",
    "CreatedDate" : "2017-12-05T19:22:16.000+0000",
    "CreatedById" : "005R0000000J2glIAC",
    "LastModifiedDate" : "2017-12-05T19:22:16.000+0000",
    "LastModifiedById" : "005R0000000J2glIAC",
    "Metadata" : {
      "actionCalls" : null,
      "apexPluginCalls" : null,
      "assignments" : null,
      "choices" : [ {
        "choiceText" : "Choice A",
```

```
    "dataType" : "String",
    "description" : null,
    "name" : "Choice_A",
    "processMetadataValues" : null,
    "userInput" : null,
    "value" : null
  }, {
    "choiceText" : "Choice B",
    "dataType" : "String",
    "description" : null,
    "name" : "Choice_B",
    "processMetadataValues" : null,
    "userInput" : null,
    "value" : null
  } ],
  "constants" : null,
  "decisions" : null,
  "description" : "FS Flow",
  "dynamicChoiceSets" : null,
  "formulas" : null,
  "fullName" : "FS_Flow-1",
  "interviewLabel" : "FS_Flow {!$Flow.CurrentDateTime}",
  "label" : "FS Flow",
  "loops" : null,
  "module" : null,
  "processMetadataValues" : null,
  "processType" : "FieldServiceMobile",
  "recordCreates" : null,
  "recordDeletes" : null,
  "recordLookups" : null,
  "recordUpdates" : null,
  "screens" : [ {
    "allowBack" : true,
    "allowFinish" : true,
    "allowPause" : true,
    "connector" : {
      "processMetadataValues" : null,
      "targetReference" : "Screen_2"
    },
    "description" : null,
    "fields" : [ {
      "choiceReferences" : null,
      "dataType" : "String",
      "defaultSelectedChoiceReference" : null,
      "defaultValue" : null,
      "description" : null,
      "extensionName" : null,
      "fieldText" : "Field 1",
      "fieldType" : "InputField",
      "helpText" : "Help Text 1",
      "inputParameters" : [ ],
      "isRequired" : false,
      "isVisible" : null,
      "name" : "Field_1",
```

```
      "outputParameters" : [ ],
      "processMetadataValues" : null,
      "scale" : null,
      "validationRule" : null
    }, {
      "choiceReferences" : [ "Choice_A", "Choice_B" ],
      "dataType" : "String",
      "defaultSelectedChoiceReference" : null,
      "defaultValue" : null,
      "description" : null,
      "extensionName" : null,
      "fieldText" : "Pick a choice",
      "fieldType" : "RadioButtons",
      "helpText" : "Pick a Choice help text",
      "inputParameters" : [ ],
      "isRequired" : false,
      "isVisible" : null,
      "name" : "Pick_a_choice",
      "outputParameters" : [ ],
      "processMetadataValues" : null,
      "scale" : null,
      "validationRule" : null
    } ],
    "helpText" : null,
    "label" : "Screen 1",
    "locationX" : 189,
    "locationY" : 178,
    "name" : "Screen_1",
    "pausedText" : null,
    "processMetadataValues" : null,
    "rules" : null,
    "showFooter" : true,
    "showHeader" : true
  }, {
    "allowBack" : true,
    "allowFinish" : true,
    "allowPause" : true,
    "connector" : null,
    "description" : null,
    "fields" : [ {
      "choiceReferences" : null,
      "dataType" : "String",
      "defaultSelectedChoiceReference" : null,
      "defaultValue" : null,
      "description" : null,
      "extensionName" : null,
      "fieldText" : "Field 1",
      "fieldType" : "InputField",
      "helpText" : null,
      "inputParameters" : [ ],
      "isRequired" : false,
      "isVisible" : null,
      "name" : "Field_1",
      "outputParameters" : [ ],
```

```
          "processMetadataValues" : null,
          "scale" : null,
          "validationRule" : null
        } ],
        "helpText" : null,
        "label" : "Screen 2",
        "locationX" : 437,
        "locationY" : 289,
        "name" : "Screen_2",
        "pausedText" : null,
        "processMetadataValues" : null,
        "rules" : null,
        "showFooter" : true,
        "showHeader" : true
      } ],
      "startElementReference" : "Screen_1",
      "steps" : null,
      "subflows" : null,
      "textTemplates" : null,
      "urls" : null,
      "variables" : null,
      "waits" : null
    },
    "FullName" : "FS_Flow-1"
  } ]
}
```


Field Service Mobile Settings

Returns information about the Field Service mobile app settings assigned to the profile of the context user. This resource is available in REST API version 42.0 and later.

The Field Service mobile app for Android and iOS comes with a variety of customizable settings, including branding colors, geolocation accuracy, app extensions, and date picker display. You can assign unique configurations of those settings to user profiles for a customized Field Service mobile experience. For example, configure separate settings to address the mobile needs of contractors, technicians of varying levels, and service crew leaders.

The FieldServiceMobileSettings object represents a configuration of the mobile app settings. The assignment of a settings configuration to a user profile is stored in a child object, MobileSettingsAssignment. A FieldServiceMobileSettings record can also have zero or more child AppExtension records. The userSettings extension enables the return of user profile settings information.

Each user profile can be associated with only one FieldServiceMobileSettings record. If a user's profile isn't explicitly assigned a settings record, the profile uses the default settings.

This resource returns information about the FieldServiceMobileApp settings assigned to the profile of the user in the context of which the call is made. It is available in orgs where Field Service is enabled, and can be used for users who have one or more of the following:

- Field Service Mobile permission set license
- View Setup user permission
- Customize Application user permission
- System Administrator profile

This resource can also be executed in the context of an Experience Builder site user with the Field Service Mobile permission set license.

Syntax

URI: `/services/data/vXX.X/sobjects/FieldServiceMobileSettings/userSettings`

Formats: JSON, XML

HTTP Method: GET

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Parameters: None.

Response Body

A FieldServiceMobileSettings record contains the following properties. All string properties have a maximum length of 7 characters unless otherwise noted. For descriptions, see FieldServiceMobileSettings in the *Salesforce Object Reference*.

Field	Type	Description
`BgGeoLocationAccuracy`	Enum	Values: Medium, Coarse, VeryCoarse.
`BgGeoLocationMinUpdateFreqMins`	Integer	
`BrandInvertedColor`	String	
`ContrastInvertedColor`	String	
`ContrastPrimaryColor`	String	
`ContrastQuaternaryColor`	String	
`ContrastQuinaryColor`	String	
`ContrastSecondaryColor`	String	
`ContrastTertiaryColor`	String	
`DefaultListViewDeveloperName`	String	Maximum length: 255.
`FeedbackPrimaryColor`	String	
`FeedbackSecondaryColor`	String	
`FeedbackSelectedColor`	String	
`FutureDaysInDatePicker`	Integer	
`GeoLocationAccuracy`	Enum	Values: Fine, Medium, Coarse.
`GeoLocationMinUpdateFreqMins`	Integer	
`Id`	String	
`IsAssignmentNotification`	Boolean	
`IsDefault`	Boolean	
`IsDispatchNotification`	Boolean	

Field	Type	Description
IsScheduleViewResourceAbsences	Boolean	Available in API version 55.0 and later.
IsSendLocationHistory	Boolean	
IsShowEditFullRecord	Boolean	
IsUseSalesforceMobileActions	Boolean	
MetadataCacheTimeDays	Integer	
NavbarBackgroundColor	String	
NavbarInvertedColor	String	
PastDaysInDatePicker	Integer	
PrimaryBrandColor	String	
RecordDataCacheTimeMins	Integer	
SecondaryBrandColor	String	
TimeIntervalSetupMins	Enum	Values: 1, 5, 15, 20, 30, 60.
UpdateScheduleTimeMins	Integer	

If any of the following is true, an error message is returned:

- The user does not have access to FieldServiceMobileSettings
- Field Service is not enabled in the org
- An internal server error occurs

Example

Request:

```
GET
/services/data/v56.0/sobjects/FieldServiceMobileSettings/userSettings
```

Response:

```
{
  "attributes" : {
    "type" : "FieldServiceMobileSettings",
    "url" : "/services/data/v46.0/sobjects/FieldServiceMobileSettings/0MfRM0000000rpR0AQ"

  },
  "Id" : "0MfRM0000000rpR0AQ",
  "IsDeleted" : false,
  "DeveloperName" : "Field_Service_Mobile_Settings",
  "Language" : "en_US",
  "MasterLabel" : "Field Service Mobile Settings",
  "CreatedDate" : "2019-02-17T05:20:47.000+0000",
  "CreatedById" : "005RM000001hTsLYAU",
```

```
  "LastModifiedDate" : "2019-03-22T22:48:17.000+0000",
  "LastModifiedById" : "005RM000001k2kpYAA",
  "SystemModstamp" : "2019-03-22T22:48:17.000+0000",
  "NavbarBackgroundColor" : "#803ABE",
  "BrandInvertedColor" : "#FFFFFF",
  "FeedbackPrimaryColor" : "#C23934",
  "FeedbackSecondaryColor" : "#13C4A3",
  "PrimaryBrandColor" : "#803ABE",
  "SecondaryBrandColor" : "#2A7AB0",
  "ContrastPrimaryColor" : "#000000",
  "ContrastSecondaryColor" : "#444444",
  "ContrastTertiaryColor" : "#9FAAB5",
  "ContrastQuaternaryColor" : "#E6E6EB",
  "ContrastQuinaryColor" : "#EEEEEE",
  "ContrastInvertedColor" : "#FFFFFF",
  "IsScheduleViewResourceAbsences" : true,
  "IsSendLocationHistory" : false,
  "GeoLocationMinUpdateFreqMins" : 10,
  "GeoLocationAccuracy" : "Medium",
  "RecordDataCacheTimeMins" : 240,
  "MetadataCacheTimeDays" : 7,
  "UpdateScheduleTimeMins" : 30,
  "IsShowEditFullRecord" : false,
  "TimeIntervalSetupMins" : "15",
  "DefaultListViewDeveloperName" : null,
  "NavbarInvertedColor" : "#FFFFFF",
  "FeedbackSelectedColor" : "#FFFFFF",
  "FutureDaysInDatePicker" : 45,
  "PastDaysInDatePicker" : 45,
  "IsDefault" : true,
  "BgGeoLocationMinUpdateFreqMins" : 60,
  "BgGeoLocationAccuracy" : "Coarse",
  "IsUseSalesforceMobileActions" : false,
  "IsAssignmentNotification" : false,
  "IsDispatchNotification" : true
}
```


Service Report Template

Returns information corresponding to one or more service report templates in field service. This resource is available in REST API version 40.0 and later.

This resource allows users to input one or more service report template IDs, and responds with information about the templates. Service reports can be generated for work orders, work order line items, and service appointments.

ServiceReportTemplate is available to API users in orgs where Field Service is enabled. It is also enabled for Experience Builder sites.

Syntax

URI: `/services/data/vXX.X/support/fieldservice/ServiceReportTemplate`

Formats: JSON

HTTP Method: GET

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Parameters:

Parameter	Description
`templateIds`	The list of service report template IDs, provided as comma-separated string values. The API response shows a failure if any template ID is invalid.
`showDefault`	A boolean parameter indicating whether the default service report template is returned in the response. If `true`, the default template is included; if `false` (default), the default template is not included. For example, if a request lists the IDs of two non-default templates and `showDefault=true`, the response will include information for three templates.
`templateTypes`	The type of record that the report template can be used for: ServiceAppointment, WorkOrder, or WorkOrderLineItem. Service report templates include four customizable sub-templates: • `WO`—Work Order • `WOLI`—Work Order Line Item • `WO_SA`—Service Appointment for Work Order • `WOLI_SA`—Service Appointment for Work Order Line Item

Response Body

A service report has three main areas which cannot be deleted: **Header**, **Body** and **Footer**. Each area must contain at least one component—a **section**, **related list**, or **signature**—though the component can be empty. The Header and Footer come with a predefined section, and additional components cannot be added to them. The Body can contain one or more of each component.

Section components have these properties.

Field	Type
`title`	String
`hideTitle`	Boolean
`hideFieldLabels`	Boolean
`columns`	Enum {1, 2}
`rightAlignment`	Enum {true,false}
`fields`	Array of {Field}

A **field** can be added to a section, related list, or signature. Columns in related lists are transformed into fields in the API. Field components have these properties.

Field	Type
`fieldType`	Enum {rta, entityField, blank} If the field type is rta, the field also contains {"richTextValue" : "<html value>"}.

Field	Type
	If the field type is entityField, the field also contains {“entityName” : “<WorkOrder>”, “fieldName” : “Account”}. If the field type is blank, no other information is required.
`row`	Position
`column`	Position

Related list components have these properties.

Field	Type
`title`	String
`hideTitle`	Boolean
`entityName`	String
`relatedListName`	String
`relatedEntityName`	String
`fields`	Array of {Field Name (String), Column Position (Integer)}

Signature components have these properties.

Field	Type
`title`	String
`hideTitle`	Boolean
`hideFieldLabels`	Boolean
`columns`	Enum {1, 2}
`rightAlignment`	Enum {true,false}
`signatureType`	Dynamic Enum {<technician 1>, <dispatcher 1>}
`fields`	Array of {Field Name (String), Position (Row,Column)}

Example

This example request uses two service report template IDs.

Request:

```
GET
/services/data/v42.0/support/fieldservice/ServiceReportTemplate?templateIds=0SLxx0000000ABC,
0SLR000000001QtOAI&showDefault=false&templateTypes=ServiceAppointment
```

Response:


```
{
  "serviceReportTemplates": [
    {
      "defaultTemplate": false,
      "error": {
        "errorCode": "INVALID_TEMPLATE_ID",
        "errorMessage": "The Service Report Template ID is invalid."
      },
      "subTemplates": [],
      "templateId": "0SLxx0000000ABC"
    },
    {
      "defaultTemplate": false,
      "error": null,
      "subTemplates": [
        {
          "regions": [
            {
              "sections": [
                {
                  "columns": 2,
                  "hideFieldLabels": false,
                  "hideTitle": false,
                  "items": [
                    {
                      "position": {
                        "column": 0,
                        "row": 0
                      },
                      "richText": "<img alt=\"User-added image\"
src=\"https://mobile1.file.force.com/servlet/rtaImage?eid=0QRR000000008oZ&amp;feoid=Data&amp;refid=0EMR00000000DGK\"></img>",

                      "type": "rta"
                    },
                    {
                      "position": {
                        "column": 1,
                        "row": 0
                      },
                      "richText": "<u><i>Salesforce.com</i></u>",
                      "type": "rta"
                    }
                  ],
                  "position": 0,
                  "rightAlign": false,
                  "title": "Service Report",
                  "type": "section"
                }
              ],
              "type": "HEADER"
            },
            {
              "sections": [
```

```
      {
        "columns": 2,
        "hideFieldLabels": false,
        "hideTitle": true,
        "items": [],
        "position": 0,
        "rightAlign": false,
        "showPageNumber": false,
        "title": "Footer Section",
        "type": "section"
      }
    ],
    "type": "FOOTER"
  },
  {
    "sections": [
      {
        "columns": 2,
        "hideFieldLabels": false,
        "hideTitle": false,
        "items": [
          {
            "entityName": "ServiceAppointment",
            "label": "Account",
            "name": "AccountId",
            "position": {
              "column": 0,
              "row": 0
            },
            "type": "entityField"
          },
          ...
          {
            "position": {
              "column": 1,
              "row": 0
            },
            "richText": "Prepared By:",
            "type": "rta"
          }
        ],
        "position": 0,
        "rightAlign": false,
        "title": "Account & Contact Information",
        "type": "section"
      },
      {
        "entityName": "WorkOrder",
        "filterCriteria": {
          "conditions": [
            {
              "field": "Status",
              "operation": "includes",
              "position": 0,
```

```
            "values": [
              "In Progress",
              "Completed"
            ]
          }
        ]
      },
      "hideTitle": false,
      "items": [
        {
          "column": 0,
          "label": "Work Order Line Item Number",
          "name": "LineItemNumber"
        },
        ...
        {
          "column": 6,
          "label": "Unit Price",
          "name": "UnitPrice"
        }
      ],
      "position": 5,
      "relatedEntityName": "WorkOrderLineItem",
      "relatedListName": "WorkOrderLineItems",
      "title": "Work Order Line Items",
      "type": "relatedList"
    },
    {
      "columns": 1,
      "hideFieldLabels": false,
      "hideTitle": false,
      "items": [
        {
          "entityName": "DigitalSignature",
          "label": "Signature",
          "name": "Document",
          "position": {
            "column": 0,
            "row": 0
          },
          "type": "entityField"
        },
        {
          "entityName": "DigitalSignature",
          "label": "Signed By",
          "name": "SignedBy",
          "position": {
            "column": 0,
            "row": 1
          },
          "type": "entityField"
        },
        {
          "entityName": "DigitalSignature",
```

```
                    "label": "Date",
                    "name": "SignedDate",
                    "position": {
                      "column": 0,
                      "row": 2
                    },
                    "type": "entityField"
                  }
                ],
                "position": 9,
                "rightAlign": false,
                "signatureType": "Default",
                "signatureTypeLabel": "Default",
                "title": "Customer Signature",
                "type": "signature"
              }
            ],
            "type": "BODY"
          }
        ],
        "subTemplateType": "WO_SA"
      },
      {
        ...
        "subTemplateType": "WOLI_SA"
      }
    ],
    "templateId": "0SLR000000001QtOAI"
  }
 ]
}
```

Appointment Bundling REST APIs

Use the Appointment Bundling REST APIs to create bundles, remove bundle members, start the automated process for creating bundles, unbundle one or more bundles, and update a bundle.

Limitations

Create Bundle, Remove Bundle Members, Unbundle, Unbundle Multiple, Update Bundle

- 1000 API calls within 24 hours.
- 50 API calls per engineer license within 24 hours (in addition to the 1000 API calls).
- 10 concurrent API calls.

Start Batch

- One API call per hour, per territory.

IN THIS SECTION:

Create Bundle

Creates a bundle of service appointments manually. This resource accepts service appointment IDs and a manual bundling policy ID. The manual bundling policy specifies the bundling rules and has to be marked for manual bundling. This resource returns the ID of the bundle service appointment. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Remove Bundle Members

Removes one or more service appointments from an existing bundle. This resource accepts service appointment IDs. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Start Batch

Creates service appointment bundles automatically. The resource uses appointment bundle policies that are marked for automatic bundling. Service appointments that are already bundled manually stay untouched by this API. The resource returns the success or failure message of starting the batch process of automatic bundling. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Unbundle

Unbundle a bundle of service appointments. This resource accepts a bundle ID. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Unbundle Multiple

Unbundle one or more bundles of service appointments. This resource accepts one or more bundle IDs. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Update Bundle

Adds service appointments to an existing bundle. This resource accepts a bundle ID and service appointment IDs. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Create Bundle

Creates a bundle of service appointments manually. This resource accepts service appointment IDs and a manual bundling policy ID. The manual bundling policy specifies the bundling rules and has to be marked for manual bundling. This resource returns the ID of the bundle service appointment. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Add a Remote Site

1. From Setup, in the Quick Find box, enter *Security*, and then select **Remote Site Settings**.
2. Click **New Remote Site**.
3. Enter a name.
4. For Remote Site URL, enter *https://api.salesforce.com/*
5. Save your changes.

Syntax

URI: `HOST/bundleflow/api/v1.0/bundle`

Where `HOST` is a variable.

Format: JSON

HTTP Method: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

If the authentication token isn't valid, a 401 HTTP status is returned.

Headers:

- `request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');`—Field Service functional domain
- `sf_api_version`—Salesforce API version. Minimum version is 53.0.
- `Authorization`—Authorization token.
- `Content-Type`—File format. Valid value is `JSON`.
- `Referer`—Org domain URL.

Request Body

The request body contains these properties.

Field	Type	Description
`user`	String	Optional. User name of user sending the request.
`initiate`	String	Type of bundle request. Valid value: `manual`.
`bundlePolicyId`	String	ID of the related bundle policy containing the bundling rules.
`saRequestPayloads`	Array	Details of service appointments that are bundled.
`serviceAppointmentId`	String	ID of service appointments to be bundled.
`action`	String	Bundling action performed on this service appointment. Valid value: `add`.

Example

This example shows a manual bundle request of two service appointments.

```
{
    "user": "Misha1",
    "initiate": "manual",
    "bundlePolicyId": "7sTx00000000006EAA",
    "saRequestPayloads": [
        {
            "serviceAppointmentId": "08px000000NzbmsAAB",
            "action": "add"
        },
        {
            "serviceAppointmentId": "08pT300000006LLIAY",
            "action": "add"
        }
    ]
}
```

Response Body

The response body contains these properties.

Field	Type	Description
`bundleId`	String	ID of the new bundle. If the bundle isn't created, this field is `null`.
`responsePayloads`	Array	Details of the bundled objects.
`objectName`	String	Type of object. Output: `ServiceAppointment`.
`objectId`	String	ID of the service appointment.
`action`	String	Action performed on the object.
`status`	String	Specifies if this object is bundled correctly. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of bundling the service appointment. 0 means success. The error codes are listed in Error Responses on page 65.
`message`	String	Success or error message of bundling the object.
`status`	String	Status of the request. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of the request. 0 means success. The error codes are listed in Error Responses on page 65.
`message`	String	Success or error message of the request.
`messageAdditionalInfo`	String	More information provided by Salesforce.

Example: Successful Response

This example shows the output of two service appointments bundled successfully.

```
{
  "bundleId": "08px000000NzdH8AAJ",
  "responsePayloads": [
    {
      "objectName": "ServiceAppointment",
      "objectId": "08px000000NzbmsAAB",
      "action": "add",
      "status": "SUCCESS",
      "messageCode": "NONE",
      "message": "Success",
      "messageParams": [

      ]
    },
    {
      "objectName": "ServiceAppointment",
      "objectId": "08pT300000006LLIAY",
      "action": "add",
      "status": "SUCCESS",
      "messageCode": "NONE",
      "message": "Success",
```

```
      "messageParams": [

      ]
    }
  ],
  "status": "SUCCESS",
  "messageCode": "NONE",
  "message": "Success"
}
```

Example: Fail Response

This example shows the output of a request that's trying to bundle a service appointment that's already a bundle service appointment. A service appointment can't be part of two bundles, so the response status is `FAIL`.

```
{
   "bundleId":null,
   "responsePayloads":[
      {
         "objectName":"ServiceAppointment",
         "objectId":"08px000000NzdH8AAJ",
         "action":"add",
         "status":"FAIL",
         "messageCode":"SA_ALREADY_BUNDLED",
         "message":"Is already a bundle service appointment.",
         "messageParams":[

         ]
      }
   ],
   "status":"FAIL",
   "messageCode":"ERROR_CREATING_BUNDLE",
   "message":"We couldn't bundle the service appointment.",
   "messageAdditionalInfo": ""
}
```

Error Responses

Code	Name	Message
0	NONE	<Service Appointment ID#>: Success.
1	UNSUPPORTED_VERSION	Ask your admin to check the API version, then try again.
2	SA_PAYLOAD_SHOULD_CONTAIN_ADD_ACTIONS_ONLY	Create bundles using add actions only.
3	EMPTY_BUNDLE_POLICY	Enter an ID for the bundlePolicyId.
4	EMPTY_BUNDLE_INITIATE	Specify manual or auto for the initiate entry.
5	EMPTY_BUNDLE_SERVICE_APPOINTMENT	Add a bundle service appointment.

Code	Name	Message
6	SA_PAYLOAD_DUPLICATED_SAS	Remove duplicate service appointment IDs.
7	SA_IS_BUNDLE	<Service Appointment ID#>: Is already a bundle service appointment.
8	SA_ALREADY_BUNDLED	<Service Appointment ID#>: Is a bundle member service appointment in another bundle.
9	STATUS_FORBIDDEN	<Service Appointment ID#>: Can't be bundled in the current status.
10	SA_PAYLOAD_SHOULD_NOT_CONTAIN_MIX_ACTIONS	Send separate requests for add and remove actions.
11	SA_IS_NOT_BUNDLE	Select a bundle service appointment and try again.
12	EXCEEDED_BUNDLE_MEMBERS_LIMIT	You reached the maximum number of service appointments for this bundle. Remove some of them and try again.
13	SA_PAYLOAD_SHOULD_CONTAIN_ONLY_BUNDLE_MEMBERS	To remove bundle members from a bundle, include only service appointments that are bundle members.
14	ERROR_IN_BASIC_VALIDATIONS	We couldn't bundle the service appointments.
15	ERROR_IN_LIMIT_VALIDATIONS	N/A
16	ERROR_IN_CRITERIA_VALIDATIONS	N/A
17	ERROR_IN_SERVICE_TERRITORY_VALIDATIONS	Confirm or revise the service territories of the selected service appointments. Or ask your admin for help.
18	ERROR_IN_RESTRICTION_VALIDATIONS	Confirm or revise the selected service appointments. Or ask your admin to check the restriction policies.
19	EXCEEDED_BUNDLE_DURATION_LIMIT	N/A
20	EMPTY_BODY	Fill in the request.
21	SA_IS_NOT_BUNDLED	<Service Appointment ID#>: Can't remove a service appointment that isn't in the bundle.
22	ERROR_IN_AGGREGATION	Confirm or revise the selected service appointments. Or ask your admin to check the aggregation policies.

Code	Name	Message
23	ERROR_IN_PROPAGATION	Confirm or revise the selected service appointments. Or ask your admin to check the propagation policies.
24	GET_POLICY_FAILURE	Ask your admin to check the bundle policies, then try again.
25	GET_CONFIG_FAILURE	Ask your admin to check the bundle config, then try again.
26	GET_BUNDLE_INFO_FAILURE	Try again later.
27	ERROR_IN_LOGIC	We couldn't bundle the service appointments.
28	EMPTY_BUNDLE_MEMBERS	Add bundle member service appointments.
29	EMPTY_BUNDLE_PARENT_WORK_ORDER_ID	Add the work order ID for the bundle service appointment.
30	UNBUNDLE_REQUEST_ASSIGNED_RESOURCES_NOT_COMPATiBLE	To unbundle a scheduled bundle, send assigned resources for each service appointment.
31	EMPTY_OLD_VALUES	To update a bundle, send the current values.
32	EMPTY_NEW_VALUES	To update a bundle, send the new values.
33	SA_PAYLOAD_SHOULD_CONTAIN_AT_LEAST_ONE_ADD_OR_REMOVE_ACTION	To update a bundle, send at least one add or remove action.
34	FORBIDDEN_FIELDS_IN_POLICY_OBJECT	Confirm or revise the selected service appointments. Or ask your admin to check the field names selected in the related policies.
35	EMPTY_SERVICE_TERRITORY	Add a Service Territory ID.
36	EMPTY_START_TIME	Add the interval's start time.
37	EMPTY_END_TIME	Add the interval's end time.
38	ERROR_IN_BUNDLE_POLICY_RFC	Confirm or revise the selected service appointments. Or ask your admin to check the recordset filter criteria in the bundle policy.
39	ERROR_IN_TIME_ZONE_VALIDATION	Select service appointments that are in the same time zone. Or ask your admin for help.

Code	Name	Message
40	ERROR_IN_LOGIC_UNBUNDLE	We couldn't unbundle the service appointment.
41	ERROR_CREATING_BUNDLE	We couldn't bundle the service appointments.
42	ERROR_ADDING_TO_BUNDLE	We couldn't add the service appointment to the bundle.
43	ERROR_REMOVING_FROM_BUNDLE	We couldn't remove the service appointment from the bundle.
44	ERROR_UNBUNDLING	We couldn't unbundle the service appointment.
58	ERROR_IN_MSW_VALIDATIONS	We can't bundle service appointments with dependencies. Remove the dependencies and try again.
-500	http error	Hmm… Something went wrong. Try again.

Code Sample from Apex

```
public static Map<String, String> createSABundle() {
    String apiVersion = '54.0'; // Spring '22
    String bundleApi = '/bundleflow/api/v1.0/bundle';
    String host = {Namespace}.BundleLogic.getBundlerFalconEnvironment();
    String ref = URL.getOrgDomainUrl().toExternalForm();

    String bundleService = host + bundleApi;

    // Create HTTP request
    HttpRequest request = new HttpRequest();
    request.setEndpoint(bundleService);
    request.setMethod('POST');
    request.setHeader('sf_api_version', apiVersion);
    request.setHeader('Content-Type', 'application/json' );
    // NOTE: This user must have 'Field Service Integration' permissions.
    request.setHeader('Authorization', 'Bearer ' + UserInfo.getSessionId());
    request.setHeader('Referer', ref);
    request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');
    request.setTimeout(120000);

    // Create the body
    Map<String, Object> body = new Map<String, Object>();
    body.put('initiate','manual');
   body.put('bundlePolicyId','7sT9A0000004DX6UAM'); // NOTE: Use a real bundle policy ID.


    Map<String, Object> saList = new Map<String, Object>();
    saList.put('serviceAppointmentId', '08p9A0000005LEGQA2'); // NOTE: Use a real Service
```

```
 Appointment ID.
    saList.put('action', 'add');

    List<Object> objectsList =  new List<Object>();
    objectsList.add(saList);

    body.put('saRequestPayloads', objectsList);

    String reqBody = JSON.serialize(body);
    System.debug(body);
    request.setBody(reqBody);

    HttpResponse response = new Http().send(request);

    // Parse the JSON response

    // Handle a redirect message
    while (response.getStatusCode() == 302) {
        request.setEndpoint(response.getHeader('Location'));
        response = new Http().send(request);
    }

    Map<String, String> returnValue = new Map<String, String>();
    returnValue.put('statusCode', String.valueOf(response.getStatusCode()));

    // Return value when we don't get a success response
    if (response.getStatusCode() != 200) {
        returnValue.put('message', 'The status code returned was not expected: ' +
response.getStatusCode() + ' ' + response.getStatus());
        System.debug(returnValue.get('message'));
        return (returnValue);

    // Return value when we do get a success response
    } else {
        returnValue.put('message', response.getBody());
        System.debug(response.getBody());
        return (returnValue);
    }
}

Map<String, String> response = createSABundle();
System.debug(response);
```

Remove Bundle Members

Removes one or more service appointments from an existing bundle. This resource accepts service appointment IDs. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Add a Remote Site

1. From Setup, in the Quick Find box, enter *Security*, and then select **Remote Site Settings**.
2. Click **New Remote Site**.

3. Enter a name.

4. For Remote Site URL, enter `https://api.salesforce.com/`

5. Save your changes.

Syntax

URI: `HOST/bundleflow/api/v1.0/bundle/remove`

Where `HOST` is a variable.

Format: JSON

HTTP Method: PATCH

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

If the authentication token isn't valid, a 401 HTTP status is returned.

Headers:

- `request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');`—Field Service functional domain
- `sf_api_version`—Salesforce API version. Minimum version is 53.0.
- `Authorization`—Authorization token.
- `Content-Type`—File format. Valid value is `JSON`.
- `Referer`—Org domain URL.

Request Body

The request body contains these properties.

Field	Type	Description
`user`	String	Optional. User name of user sending the request.
`initiate`	String	Type of bundle request. Valid value: `manual`.
`saRequestPayloads`	Array	Details of the service appointments to be removed from a bundle.
`serviceAppointmentId`	String	ID of a service appointment.
`action`	String	Bundling action performed on this service appointment. Valid values: `remove`.

Example

This example shows a manual request to remove two service appointments from a bundle.

```
{
   "user":"Misha",
   "initiate":"manual",
   "saRequestPayloads":[
      {
         "serviceAppointmentId":"08px000000NzbmkAAB",
```

```
            "action":"remove"
        },
        {
            "serviceAppointmentId":"08pT300000006LLIAY",
            "action":"remove"
        }
    ]
}
```

Response Body

The response body contains these properties.

Field	Type	Description
`bundleId`	String	ID of the bundle to update. If the update isn't successful, it's `null`.
`responsePayloads`	Array	Details of the service appointment objects that are part of the bundle request.
`objectName`	String	Type of object. Output: `ServiceAppointment`.
`objectId`	String	ID of the service appointment.
`action`	String	Bundling action performed on this service appointment. Valid values: `remove`.
`status`	String	Status of updating the service appointment. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of updating the service appointment. 0 means success. The error codes are listed in Error Responses on page 72.
`message`	String	Success or error message of updating the service appointment.
`status`	String	Status of the request. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of the request. 0 means success. The error codes are listed in Error Responses on page 72.
`message`	String	Success or error message of the request.
`messageAddionalInfo`	String	More information provided by Salesforce.

Example: Successful Response

This example shows a successful response for removing a service appointment from a bundle.

```
{
    "bundleId": "08px000000OAkUXAA1",
    "responsePayloads": [
        {
            "objectName": "ServiceAppointment",
            "objectId": "08px000000OAjPRAA1",
            "action": "remove",
            "status": "SUCCESS",
            "messageCode": "NONE",
            "message": "Success",
```

```
            "messageParams": []
        }
    ],
    "status": "SUCCESS",
    "messageCode": "NONE",
    "message": "Success"
}
```

Example: Fail Response

This example shows the output of a request where the user entered an invalid service appointment ID.

```
{
    "bundleId": null,
    "responsePayloads": [
        {
            "status": "FAIL",
            "messageCode": "GET_BUNDLE_INFO_FAILURE",
            "message": "Try again later.",
            "messageParams": []
        }
    ],
    "status": "FAIL",
    "messageCode": "ERROR_REMOVING_FROM_BUNDLE",
    "message": "We couldn't remove the service appointment from the bundle."
   "messageAdditionalInfo": ""
}
```

Error Responses

Code	Name	Message
0	NONE	<Service Appointment ID#>: Success.
1	UNSUPPORTED_VERSION	Ask your admin to check the API version, then try again.
2	SA_PAYLOAD_SHOULD_CONTAIN_ADD_ACTIONS_ONLY	Create bundles using add actions only.
3	EMPTY_BUNDLE_POLICY	Enter an ID for the bundlePolicyId.
4	EMPTY_BUNDLE_INITIATE	Specify manual or auto for the initiate entry.
5	EMPTY_BUNDLE_SERVICE_APPOINTMENT	Add a bundle service appointment.
6	SA_PAYLOAD_DUPLICATED_SAS	Remove duplicate service appointment IDs.
7	SA_IS_BUNDLE	<Service Appointment ID#>: Is already a bundle service appointment.

Code	Name	Message
8	SA_ALREADY_BUNDLED	<Service Appointment ID#>: Is a bundle member service appointment in another bundle.
9	STATUS_FORBIDDEN	<Service Appointment ID#>: Can't be bundled in the current status.
10	SA_PAYLOAD_SHOULD_NOT_CONTAIN_MIX_ACTIONS	Send separate requests for add and remove actions.
11	SA_IS_NOT_BUNDLE	Select a bundle service appointment and try again.
12	EXCEEDED_BUNDLE_MEMBERS_LIMIT	You reached the maximum number of service appointments for this bundle. Remove some of them and try again.
13	SA_PAYLOAD_SHOULD_CONTAIN_ONLY_BUNDLE_MEMBERS	To remove bundle members from a bundle, include only service appointments that are bundle members.
14	ERROR_IN_BASIC_VALIDATIONS	We couldn't bundle the service appointments.
15	ERROR_IN_LIMIT_VALIDATIONS	N/A
16	ERROR_IN_CRITERIA_VALIDATIONS	N/A
17	ERROR_IN_SERVICE_TERRITORY_VALIDATIONS	Confirm or revise the service territories of the selected service appointments. Or ask your admin for help.
18	ERROR_IN_RESTRICTION_VALIDATIONS	Confirm or revise the selected service appointments. Or ask your admin to check the restriction policies.
19	EXCEEDED_BUNDLE_DURATION_LIMIT	N/A
20	EMPTY_BODY	Fill in the request.
21	SA_IS_NOT_BUNDLED	<Service Appointment ID#>: Can't remove a service appointment that isn't in the bundle.
22	ERROR_IN_AGGREGATION	Confirm or revise the selected service appointments. Or ask your admin to check the aggregation policies.
23	ERROR_IN_PROPAGATION	Confirm or revise the selected service appointments. Or ask your admin to check the propagation policies.
24	GET_POLICY_FAILURE	Ask your admin to check the bundle policies, then try again.

Code	Name	Message
25	GET_CONFIG_FAILURE	Ask your admin to check the bundle config, then try again.
26	GET_BUNDLE_INFO_FAILURE	Try again later.
27	ERROR_IN_LOGIC	We couldn't bundle the service appointments.
28	EMPTY_BUNDLE_MEMBERS	Add bundle member service appointments.
29	EMPTY_BUNDLE_PARENT_WORK_ORDER_ID	Add the work order ID for the bundle service appointment.
30	UNBUNDLE_REQUEST_ASSIGNED_RESOURCES_NOT_COMPATiBLE	To unbundle a scheduled bundle, send assigned resources for each service appointment.
31	EMPTY_OLD_VALUES	To update a bundle, send the current values.
32	EMPTY_NEW_VALUES	To update a bundle, send the new values.
33	SA_PAYLOAD_SHOULD_CONTAIN_AT_LEAST_ONE_ADD_OR_REMOVE_ACTION	To update a bundle, send at least one add or remove action.
34	FORBIDDEN_FIELDS_IN_POLICY_OBJECT	Confirm or revise the selected service appointments. Or ask your admin to check the field names selected in the related policies.
35	EMPTY_SERVICE_TERRITORY	Add a Service Territory ID.
36	EMPTY_START_TIME	Add the interval's start time.
37	EMPTY_END_TIME	Add the interval's end time.
38	ERROR_IN_BUNDLE_POLICY_RFC	Confirm or revise the selected service appointments. Or ask your admin to check the recordset filter criteria in the bundle policy.
39	ERROR_IN_TIME_ZONE_VALIDATION	Select service appointments that are in the same time zone. Or ask your admin for help.
40	ERROR_IN_LOGIC_UNBUNDLE	We couldn't unbundle the service appointment.
41	ERROR_CREATING_BUNDLE	We couldn't bundle the service appointments.
42	ERROR_ADDING_TO_BUNDLE	We couldn't add the service appointment to the bundle.

Code	Name	Message
43	ERROR_REMOVING_FROM_BUNDLE	We couldn't remove the service appointment from the bundle.
44	ERROR_UNBUNDLING	We couldn't unbundle the service appointment.
58	ERROR_IN_MSW_VALIDATIONS	We can't bundle service appointments with dependencies. Remove the dependencies and try again.
-500	http error	Hmm… Something went wrong. Try again.

Start Batch

Creates service appointment bundles automatically. The resource uses appointment bundle policies that are marked for automatic bundling. Service appointments that are already bundled manually stay untouched by this API. The resource returns the success or failure message of starting the batch process of automatic bundling. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Add a Remote Site

1. From Setup, in the Quick Find box, enter *Security*, and then select **Remote Site Settings**.
2. Click **New Remote Site**.
3. Enter a name.
4. For Remote Site URL, enter *https://api.salesforce.com/*
5. Save your changes.

Syntax

URI: `HOST/bundleflow/api/v1.0/startbatch`

Where `HOST` is a variable.

Format: JSON

HTTP Method: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

If the authentication token isn't valid, a 401 HTTP status is returned.

Headers:

- `request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');`—Field Service functional domain
- `sf_api_version`—Salesforce API version. Minimum version is 53.0.
- `Authorization`—Authorization token.
- `Content-Type`—File format. Valid value is `JSON`.

- `Referer`—Org domain URL.

Request Body

The request body contains these properties.

Field	Type	Description
`operation`	String	The bundling operation. Valid value: `start-batch-processing`.

Example

This example shows an automated bundling request of service appointments.

```
{
    "operation": "start-batch-processing"
}
```

Response Body

The response body contains these properties.

Field	Type	Description
`httpStatus`	Number	Response HTTP status.
`statusDescription`	String	Response status description.
`responsePayload`	String	Message about the batch status. If it's successful, the payload is null. If the batch bundling fails, the payload shows an error message, such as, "Wrong API version" or "Failed to start batch agent".
`messageCode`	Number	Message code is always 0.
`messageAdditionalInfo`	String	More information provided by Salesforce.

Example: Successful Response

This example shows the output of a batch process of automatic bundling that started successfully.

```
{
   "httpStatus":200,
   "statusDescription":null,
   "responsePayload":null,
   "messageCode":"NONE"}
}
```

Example: Fail Response

This example shows the output of a batch process that failed to start because the operation parameter of the request is `null` or empty.

```
{
   "httpStatus":400,
   "statusDescription":null,
   "responsePayload":"Failed to start batch agent",
   "messageCode":"NONE"
   "messageAdditionalInfo": ""
}
```

Unbundle

Unbundle a bundle of service appointments. This resource accepts a bundle ID. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Add a Remote Site

1. From Setup, in the Quick Find box, enter *Security*, and then select **Remote Site Settings**.
2. Click **New Remote Site**.
3. Enter a name.
4. For Remote Site URL, enter *https://api.salesforce.com/*
5. Save your changes.

Syntax

URI: `HOST/bundleflow/api/v1.0/bundle/<ID>?initiate=manual`

Where `HOST` is a variable.

Format: JSON

HTTP Method: DELETE

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

If the authentication token isn't valid, a 401 HTTP status is returned.

Headers:

- `request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');`—Field Service functional domain
- `sf_api_version`—Salesforce API version. Minimum version is 53.0.
- `Authorization`—Authorization token.
- `Content-Type`—File format. Valid value is `JSON`.
- `Referer`—Org domain URL.

Parameters:

Parameter	Description
`ID`	Unique ID of the bundle.

Response Body

The response body contains these properties.

Field	Type	Description
`bundleId`	String	ID of the bundle. If the unbundling fails, it's `null`.
`responsePayloads`	Array	Response details of unbundling the service appointments. If it's successful, the payload is `null`. If the unbundling fails, the payload shows error details.
`status`	String	Status of unbundling the service appointments. Output: `FAIL`.
`messageCode`	Number	Error code. The error codes are listed in Error Responses on page 79.
`message`	String	Error message.
`status`	String	Status of the request. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of the request. 0 means success. The error codes are listed in Error Responses on page 79.
`message`	String	Success or error message of the request.
`messageAdditionalInfo`	String	More information provided by Salesforce.

Example: Successful Response

This example shows the output of service appointments that are unbundled successfully.

```
{
   "bundleId":"08px000000NzdMXAAZ",
   "responsePayloads":null,
   "status":"SUCCESS",
   "messageCode":"NONE",
   "message":"Success"
}
```

Example: Fail Response

This example shows the output of a request where the user entered a service appointment ID instead of a bundle ID. The request only works with bundle IDs, so the response status is `FAIL`.

```
{
   "bundleId":null,
   "responsePayloads":[
      {
         "status":"FAIL",
         "messageCode":"GET_BUNDLE_INFO_FAILURE",
         "message":"Replace the service appointment with a bundle service appointment.",
```

```
            "messageParams":[

            ]
        }
    ],
    "status":"FAIL",
    "messageCode":"ERROR_UNBUNDLING",
    "message":"We couldn’t unbundle the service appointment.",
    "messageAdditionalInfo": ""
}
```

Error Responses

Code	Name	Message
0	NONE	<Service Appointment ID#>: Success.
1	UNSUPPORTED_VERSION	Ask your admin to check the API version, then try again.
2	SA_PAYLOAD_SHOULD_CONTAIN_ADD_ACTIONS_ONLY	Create bundles using add actions only.
3	EMPTY_BUNDLE_POLICY	Enter an ID for the bundlePolicyId.
4	EMPTY_BUNDLE_INITIATE	Specify manual or auto for the initiate entry.
5	EMPTY_BUNDLE_SERVICE_APPOINTMENT	Add a bundle service appointment.
6	SA_PAYLOAD_DUPLICATED_SAS	Remove duplicate service appointment IDs.
7	SA_IS_BUNDLE	<Service Appointment ID#>: Is already a bundle service appointment.
8	SA_ALREADY_BUNDLED	<Service Appointment ID#>: Is a bundle member service appointment in another bundle.
9	STATUS_FORBIDDEN	<Service Appointment ID#>: Can’t be bundled in the current status.
10	SA_PAYLOAD_SHOULD_NOT_CONTAIN_MIX_ACTIONS	Send separate requests for add and remove actions.
11	SA_IS_NOT_BUNDLE	Select a bundle service appointment and try again.
12	EXCEEDED_BUNDLE_MEMBERS_LIMIT	You reached the maximum number of service appointments for this bundle. Remove some of them and try again.
13	SA_PAYLOAD_SHOULD_CONTAIN_ONLY_BUNDLE_MEMBERS	To remove bundle members from a bundle, include only service appointments that are bundle members.

Code	Name	Message
14	ERROR_IN_BASIC_VALIDATIONS	We couldn't bundle the service appointments.
15	ERROR_IN_LIMIT_VALIDATIONS	N/A
16	ERROR_IN_CRITERIA_VALIDATIONS	N/A
17	ERROR_IN_SERVICE_TERRITORY_VALIDATIONS	Confirm or revise the service territories of the selected service appointments. Or ask your admin for help.
18	ERROR_IN_RESTRICTION_VALIDATIONS	Confirm or revise the selected service appointments. Or ask your admin to check the restriction policies.
19	EXCEEDED_BUNDLE_DURATION_LIMIT	N/A
20	EMPTY_BODY	Fill in the request.
21	SA_IS_NOT_BUNDLED	<Service Appointment ID#>: Can't remove a service appointment that isn't in the bundle.
22	ERROR_IN_AGGREGATION	Confirm or revise the selected service appointments. Or ask your admin to check the aggregation policies.
23	ERROR_IN_PROPAGATION	Confirm or revise the selected service appointments. Or ask your admin to check the propagation policies.
24	GET_POLICY_FAILURE	Ask your admin to check the bundle policies, then try again.
25	GET_CONFIG_FAILURE	Ask your admin to check the bundle config, then try again.
26	GET_BUNDLE_INFO_FAILURE	Try again later.
27	ERROR_IN_LOGIC	We couldn't bundle the service appointments.
28	EMPTY_BUNDLE_MEMBERS	Add bundle member service appointments.
29	EMPTY_BUNDLE_PARENT_WORK_ORDER_ID	Add the work order ID for the bundle service appointment.
30	UNBUNDLE_REQUEST_ASSIGNED_RESOURCES_NOT_COMPATiBLE	To unbundle a scheduled bundle, send assigned resources for each service appointment.
31	EMPTY_OLD_VALUES	To update a bundle, send the current values.

Code	Name	Message
32	EMPTY_NEW_VALUES	To update a bundle, send the new values.
33	SA_PAYLOAD_SHOULD_CONTAIN_AT_LEAST_ONE_ADD_OR_REMOVE_ACTION	To update a bundle, send at least one add or remove action.
34	FORBIDDEN_FIELDS_IN_POLICY_OBJECT	Confirm or revise the selected service appointments. Or ask your admin to check the field names selected in the related policies.
35	EMPTY_SERVICE_TERRITORY	Add a Service Territory ID.
36	EMPTY_START_TIME	Add the interval's start time.
37	EMPTY_END_TIME	Add the interval's end time.
38	ERROR_IN_BUNDLE_POLICY_RFC	Confirm or revise the selected service appointments. Or ask your admin to check the recordset filter criteria in the bundle policy.
39	ERROR_IN_TIME_ZONE_VALIDATION	Select service appointments that are in the same time zone. Or ask your admin for help.
40	ERROR_IN_LOGIC_UNBUNDLE	We couldn't unbundle the service appointment.
41	ERROR_CREATING_BUNDLE	We couldn't bundle the service appointments.
42	ERROR_ADDING_TO_BUNDLE	We couldn't add the service appointment to the bundle.
43	ERROR_REMOVING_FROM_BUNDLE	We couldn't remove the service appointment from the bundle.
44	ERROR_UNBUNDLING	We couldn't unbundle the service appointment.
58	ERROR_IN_MSW_VALIDATIONS	We can't bundle service appointments with dependencies. Remove the dependencies and try again.
-500	http error	Hmm… Something went wrong. Try again.

Unbundle Multiple

Unbundle one or more bundles of service appointments. This resource accepts one or more bundle IDs. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Add a Remote Site

1. From Setup, in the Quick Find box, enter *Security*, and then select **Remote Site Settings**.
2. Click **New Remote Site**.
3. Enter a name.
4. For Remote Site URL, enter *https://api.salesforce.com/*
5. Save your changes.

Syntax

URI: `HOST/bundleflow/api/v1.0/unbundleMultiple`

Where `HOST` is a variable.

Format: JSON

HTTP Method: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

If the authentication token isn't valid, a 401 HTTP status is returned.

Headers:

- `request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');`—Field Service functional domain
- `sf_api_version`—Salesforce API version. Minimum version is 53.0.
- `Authorization`—Authorization token.
- `Content-Type`—File format. Valid value is `JSON`.
- `Referer`—Org domain URL.

Request Body

The request body contains these properties.

Field	Type	Description
`user`	String	Optional. User name of user sending the request.
`initiate`	String	Type of bundle request. Valid value: `manual`.
`saRequestPayloads`	Array	IDs of bundle service appointment to unbundle.
`serviceAppointmentId`	String	ID of a bundle service appointment.

Example

This example shows a request to unbundle two bundles.

```
{
   "user":"Misha",
   "initiate":"manual",
```

```
   "saRequestPayloads":[
      {
         "serviceAppointmentId":"08px000000NzdLFAAZ"
      },
      {
         "serviceAppointmentId":"08px000000NzdH8AAJ"
      }
   ]
}
```

Response Body

The response body is an array of objects with these fields. Each object contains response details for a bundle service appointment you requested to unbundle.

Field	Type	Description
`headers`	Object	For future use.
`body`	Object	Details of unbundling the service appointment bundle.
`bundle id`	String	ID of the bundle that is unbundled. If the unbundling fails, it's `null`.
`responsePayloads`	Array	Response details of unbundling the service appointments. If it's successful, the payload is `null`. If the unbundling fails, the payload shows error details.
`status`	String	Failure status of unbundling the service appointments. Output: `FAIL`.
`messageCode`	Number	Error code. The error codes are listed in Error Responses on page 84.
`message`	String	Error message.
`status`	String	Status of unbundling the service appointments. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of unbundling the bundle. 0 means success. The error codes are listed in Error Responses on page 84.
`message`	String	Success or error message of unbundling the bundle.
`statusCode`	String	Success or error code of the request.
`statusCodeValue`	Number	Success or error code number of the request. 0 means success. The error codes are listed in Error Responses on page 84.

Example: Successful Response

This example shows a successful response of unbundling two service appointment bundles.

```
{
    "httpStatus": 200,
    "statusDescription": "OK",
    "responsePayload": "[ {\n  \"headers\" : { },\n  \"body\" : {\n    \"bundleId\" :
\"08px000000OAph4AAD\",\n    \"responsePayloads\" : null,\n    \"status\" : \"SUCCESS\",\n
    \"messageCode\" : \"NONE\",\n    \"message\" : \"Success\"\n  },\n  \"statusCode\" :
\"OK\",\n  \"statusCodeValue\" : 200\n} ]",
```

```
    "messageCode": "NONE"
}
```

Example: Fail Response

This example shows the output of a request where the user entered a service appointment ID instead of a bundle ID. The request only works with bundle IDs, so the response status is `FAIL`.

```
[
  {
    "httpStatus": 200,
    "statusDescription": "OK",
    "responsePayload": "[ {\n  \"headers\" : { },\n  \"body\" : {\n    \"bundleId\" :
null,\n    \"responsePayloads\" : [ {\n      \"status\" : \"FAIL\",\n      \"messageCode\"
 : \"GET_BUNDLE_INFO_FAILURE\",\n      \"message\" : \"Try again later.\",\n
\"messageParams\" : [ ]\n    } ],\n    \"status\" : \"FAIL\",\n    \"messageCode\" :
\"ERROR_UNBUNDLING\",\n    \"message\" : \"We couldn't unbundle the service appointment.\",\n
    \"messageAdditionalInfo\" : \"(bundleSaRef)\\n  errorCode: NOT_FOUND\\n  message:
Provided external ID field does not exist or is not accessible: 123\"\n  },\n  \"statusCode\"
 : \"OK\",\n  \"statusCodeValue\" : 200\n} ]",
    "messageCode": "NONE"
   "messageAdditionalInfo": "(bundleSaRef)\n  errorCode: NOT_FOUND\n  message:   Provided
 external ID field does not exist or is not accessible: 123"
}
```

Error Responses

Code	Name	Message
0	NONE	<Service Appointment ID#>: Success.
1	UNSUPPORTED_VERSION	Ask your admin to check the API version, then try again.
2	SA_PAYLOAD_SHOULD_CONTAIN_ADD_ACTIONS_ONLY	Create bundles using add actions only.
3	EMPTY_BUNDLE_POLICY	Enter an ID for the bundlePolicyId.
4	EMPTY_BUNDLE_INITIATE	Specify manual or auto for the initiate entry.
5	EMPTY_BUNDLE_SERVICE_APPOINTMENT	Add a bundle service appointment.
6	SA_PAYLOAD_DUPLICATED_SAS	Remove duplicate service appointment IDs.
7	SA_IS_BUNDLE	<Service Appointment ID#>: Is already a bundle service appointment.
8	SA_ALREADY_BUNDLED	<Service Appointment ID#>: Is a bundle member service appointment in another bundle.

Code	Name	Message
9	STATUS_FORBIDDEN	<Service Appointment ID#>: Can't be bundled in the current status.
10	SA_PAYLOAD_SHOULD_NOT_CONTAIN_MIX_ACTIONS	Send separate requests for add and remove actions.
11	SA_IS_NOT_BUNDLE	Select a bundle service appointment and try again.
12	EXCEEDED_BUNDLE_MEMBERS_LIMIT	You reached the maximum number of service appointments for this bundle. Remove some of them and try again.
13	SA_PAYLOAD_SHOULD_CONTAIN_ONLY_BUNDLE_MEMBERS	To remove bundle members from a bundle, include only service appointments that are bundle members.
14	ERROR_IN_BASIC_VALIDATIONS	We couldn't bundle the service appointments.
15	ERROR_IN_LIMIT_VALIDATIONS	N/A
16	ERROR_IN_CRITERIA_VALIDATIONS	N/A
17	ERROR_IN_SERVICE_TERRITORY_VALIDATIONS	Confirm or revise the service territories of the selected service appointments. Or ask your admin for help.
18	ERROR_IN_RESTRICTION_VALIDATIONS	Confirm or revise the selected service appointments. Or ask your admin to check the restriction policies.
19	EXCEEDED_BUNDLE_DURATION_LIMIT	N/A
20	EMPTY_BODY	Fill in the request.
21	SA_IS_NOT_BUNDLED	<Service Appointment ID#>: Can't remove a service appointment that isn't in the bundle.
22	ERROR_IN_AGGREGATION	Confirm or revise the selected service appointments. Or ask your admin to check the aggregation policies.
23	ERROR_IN_PROPAGATION	Confirm or revise the selected service appointments. Or ask your admin to check the propagation policies.
24	GET_POLICY_FAILURE	Ask your admin to check the bundle policies, then try again.
25	GET_CONFIG_FAILURE	Ask your admin to check the bundle config, then try again.

Code	Name	Message
26	GET_BUNDLE_INFO_FAILURE	Try again later.
27	ERROR_IN_LOGIC	We couldn't bundle the service appointments.
28	EMPTY_BUNDLE_MEMBERS	Add bundle member service appointments.
29	EMPTY_BUNDLE_PARENT_WORK_ORDER_ID	Add the work order ID for the bundle service appointment.
30	UNBUNDLE_REQUEST_ASSIGNED_RESOURCES_NOT_COMPATiBLE	To unbundle a scheduled bundle, send assigned resources for each service appointment.
31	EMPTY_OLD_VALUES	To update a bundle, send the current values.
32	EMPTY_NEW_VALUES	To update a bundle, send the new values.
33	SA_PAYLOAD_SHOULD_CONTAIN_AT_LEAST_ONE_ADD_OR_REMOVE_ACTION	To update a bundle, send at least one add or remove action.
34	FORBIDDEN_FIELDS_IN_POLICY_OBJECT	Confirm or revise the selected service appointments. Or ask your admin to check the field names selected in the related policies.
35	EMPTY_SERVICE_TERRITORY	Add a Service Territory ID.
36	EMPTY_START_TIME	Add the interval's start time.
37	EMPTY_END_TIME	Add the interval's end time.
38	ERROR_IN_BUNDLE_POLICY_RFC	Confirm or revise the selected service appointments. Or ask your admin to check the recordset filter criteria in the bundle policy.
39	ERROR_IN_TIME_ZONE_VALIDATION	Select service appointments that are in the same time zone. Or ask your admin for help.
40	ERROR_IN_LOGIC_UNBUNDLE	We couldn't unbundle the service appointment.
41	ERROR_CREATING_BUNDLE	We couldn't bundle the service appointments.
42	ERROR_ADDING_TO_BUNDLE	We couldn't add the service appointment to the bundle.
43	ERROR_REMOVING_FROM_BUNDLE	We couldn't remove the service appointment from the bundle.

Code	Name	Message
44	ERROR_UNBUNDLING	We couldn't unbundle the service appointment.
58	ERROR_IN_MSW_VALIDATIONS	We can't bundle service appointments with dependencies. Remove the dependencies and try again.
-500	http error	Hmm… Something went wrong. Try again.

Update Bundle

Adds service appointments to an existing bundle. This resource accepts a bundle ID and service appointment IDs. This API is not supported in Gov Cloud. This endpoint is available in version 54.0 and later of the Salesforce API.

Add a Remote Site

1. From Setup, in the Quick Find box, enter *Security*, and then select **Remote Site Settings**.
2. Click **New Remote Site**.
3. Enter a name.
4. For Remote Site URL, enter *https://api.salesforce.com/*
5. Save your changes.

Syntax

URI: `HOST/bundleflow/api/v1.0/bundle/{ID}`

Where `HOST` is a variable.

Format: JSON

HTTP Method: PATCH

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

If the authentication token isn't valid, a 401 HTTP status is returned.

Headers:

- `request.setHeader('x-sfdc-tenant-id', 'core/prod/ORG-ID_18_Characters');`—Field Service functional domain
- `sf_api_version`—Salesforce API version. Minimum version is 53.0.
- `Authorization`—Authorization token.
- `Content-Type`—File format. Valid value is `JSON`.
- `Referer`—Org domain URL.

Parameters:

Parameter	Description
`ID`	Unique ID of the bundle.

Request Body

The request body contains these properties.

Field	Type	Description
`user`	String	Optional. User name of user sending the request.
`initiate`	String	Type of bundle request. Valid value: `manual`.
`saRequestPayloads`	Array	Details of the service appointments to be added to this bundle.
`serviceAppointmentId`	String	ID of a service appointment.
`action`	String	Bundling action performed on this service appointment. Valid values: `add`.

Example

This example shows a manual request to add two service appointments to a bundle.

```
{
   "user":"Misha",
   "initiate":"manual",
   "saRequestPayloads":[
      {
         "serviceAppointmentId":"08px000000NzbmkAAB",
         "action":"add"
      },
      {
         "serviceAppointmentId":"08pT300000006LLIAY",
         "action":"add"
      }
   ]
}
```

Response Body

The response body contains these properties.

Field	Type	Description
`bundleId`	String	ID of the bundle to update. If the update isn't successful, it's `null`.
`responsePayloads`	Array	Details of the service appointment objects that are part of the bundle request.
`objectName`	String	Type of object. Output: `ServiceAppointment`.
`objectId`	String	ID of the service appointment.
`action`	String	Bundling action performed on this service appointment. Valid values: `add`.

Field	Type	Description
`status`	String	Status of updating the service appointment. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of updating the service appointment. 0 means success. The error codes are listed in Error Responses on page 90.
`message`	String	Success or error message of updating the service appointment.
`status`	String	Status of the request. Output: `SUCCESS` or `FAIL`.
`messageCode`	Number	Success or error code of the request. 0 means success. The error codes are listed in Error Responses on page 90.
`message`	String	Success or error message of the request.
`messageAdditionalInfo`	String	More information provided by Salesforce.

Example: Successful Response

This example shows a successful response of adding a service appointment to a bundle.

```
{
    "bundleId": "08px000000OAkUXAA1",
    "responsePayloads": [
        {
            "objectName": "ServiceAppointment",
            "objectId": "08px000000OAjPRAA1",
            "action": "add",
            "status": "SUCCESS",
            "messageCode": "NONE",
            "message": "Success",
            "messageParams": []
        }
    ],
    "status": "SUCCESS",
    "messageCode": "NONE",
    "message": "Success"
}
```

Example: Fail Response

This example shows the output of a request where the user entered an invalid service appointment ID.

```
{
    "bundleId": *null*,
    "responsePayloads": [
        {
            "status": "FAIL",
            "messageCode": "GET_BUNDLE_INFO_FAILURE",
            "message": "Try again later.",
            "messageParams": []
        }
    ],
    "status": "FAIL",
```

```
    "messageCode": "ERROR_ADDING_TO_BUNDLE",
    "message": "We couldn't add the service appointment to the bundle.",
   "messageAdditionalInfo": ""
}
```

Error Responses

Code	Name	Message
0	NONE	<Service Appointment ID#>: Success.
1	UNSUPPORTED_VERSION	Ask your admin to check the API version, then try again.
2	SA_PAYLOAD_SHOULD_CONTAIN_ADD_ACTIONS_ONLY	Create bundles using add actions only.
3	EMPTY_BUNDLE_POLICY	Enter an ID for the bundlePolicyId.
4	EMPTY_BUNDLE_INITIATE	Specify manual or auto for the initiate entry.
5	EMPTY_BUNDLE_SERVICE_APPOINTMENT	Add a bundle service appointment.
6	SA_PAYLOAD_DUPLICATED_SAS	Remove duplicate service appointment IDs.
7	SA_IS_BUNDLE	<Service Appointment ID#>: Is already a bundle service appointment.
8	SA_ALREADY_BUNDLED	<Service Appointment ID#>: Is a bundle member service appointment in another bundle.
9	STATUS_FORBIDDEN	<Service Appointment ID#>: Can't be bundled in the current status.
10	SA_PAYLOAD_SHOULD_NOT_CONTAIN_MIX_ACTIONS	Send separate requests for add and remove actions.
11	SA_IS_NOT_BUNDLE	Select a bundle service appointment and try again.
12	EXCEEDED_BUNDLE_MEMBERS_LIMIT	You reached the maximum number of service appointments for this bundle. Remove some of them and try again.
13	SA_PAYLOAD_SHOULD_CONTAIN_ONLY_BUNDLE_MEMBERS	To remove bundle members from a bundle, include only service appointments that are bundle members.
14	ERROR_IN_BASIC_VALIDATIONS	We couldn't bundle the service appointments.
15	ERROR_IN_LIMIT_VALIDATIONS	N/A
16	ERROR_IN_CRITERIA_VALIDATIONS	N/A

Code	Name	Message
17	ERROR_IN_SERVICE_TERRITORY_VALIDATIONS	Confirm or revise the service territories of the selected service appointments. Or ask your admin for help.
18	ERROR_IN_RESTRICTION_VALIDATIONS	Confirm or revise the selected service appointments. Or ask your admin to check the restriction policies.
19	EXCEEDED_BUNDLE_DURATION_LIMIT	N/A
20	EMPTY_BODY	Fill in the request.
21	SA_IS_NOT_BUNDLED	<Service Appointment ID#>: Can't remove a service appointment that isn't in the bundle.
22	ERROR_IN_AGGREGATION	Confirm or revise the selected service appointments. Or ask your admin to check the aggregation policies.
23	ERROR_IN_PROPAGATION	Confirm or revise the selected service appointments. Or ask your admin to check the propagation policies.
24	GET_POLICY_FAILURE	Ask your admin to check the bundle policies, then try again.
25	GET_CONFIG_FAILURE	Ask your admin to check the bundle config, then try again.
26	GET_BUNDLE_INFO_FAILURE	Try again later.
27	ERROR_IN_LOGIC	We couldn't bundle the service appointments.
28	EMPTY_BUNDLE_MEMBERS	Add bundle member service appointments.
29	EMPTY_BUNDLE_PARENT_WORK_ORDER_ID	Add the work order ID for the bundle service appointment.
30	UNBUNDLE_REQUEST_ASSIGNED_RESOURCES_NOT_COMPATiBLE	To unbundle a scheduled bundle, send assigned resources for each service appointment.
31	EMPTY_OLD_VALUES	To update a bundle, send the current values.
32	EMPTY_NEW_VALUES	To update a bundle, send the new values.
33	SA_PAYLOAD_SHOULD_CONTAIN_AT_LEAST_ONE_ADD_OR_REMOVE_ACTION	To update a bundle, send at least one add or remove action.

Code	Name	Message
34	FORBIDDEN_FIELDS_IN_POLICY_OBJECT	Confirm or revise the selected service appointments. Or ask your admin to check the field names selected in the related policies.
35	EMPTY_SERVICE_TERRITORY	Add a Service Territory ID.
36	EMPTY_START_TIME	Add the interval's start time.
37	EMPTY_END_TIME	Add the interval's end time.
38	ERROR_IN_BUNDLE_POLICY_RFC	Confirm or revise the selected service appointments. Or ask your admin to check the recordset filter criteria in the bundle policy.
39	ERROR_IN_TIME_ZONE_VALIDATION	Select service appointments that are in the same time zone. Or ask your admin for help.
40	ERROR_IN_LOGIC_UNBUNDLE	We couldn't unbundle the service appointment.
41	ERROR_CREATING_BUNDLE	We couldn't bundle the service appointments.
42	ERROR_ADDING_TO_BUNDLE	We couldn't add the service appointment to the bundle.
43	ERROR_REMOVING_FROM_BUNDLE	We couldn't remove the service appointment from the bundle.
44	ERROR_UNBUNDLING	We couldn't unbundle the service appointment.
58	ERROR_IN_MSW_VALIDATIONS	We can't bundle service appointments with dependencies. Remove the dependencies and try again.
-500	http error	Hmm… Something went wrong. Try again.

REST Reference

You can access Field Service APIs using REST endpoints. These REST APIs follow similar conventions as Connect REST APIs.

To understand the architecture, authentication, rate limits, and how the requests and responses work, see Connect REST API Developer Guide.

IN THIS SECTION:

Resources
Field Service Connect APIs are RESTful APIs. You can access Field Service Connect REST resources, including Scheduling Operations, Recordset Filter Criteria, and Shifts from Pattern using REST endpoints. These REST APIs follow similar conventions as Connect REST APIs. Here's a list of Field Service API resources.

Request Bodies
Here's a list of Scheduling Operations API request bodies.

Response Bodies
Here's a list of Scheduling Operations response bodies.

Resources

Field Service Connect APIs are RESTful APIs. You can access Field Service Connect REST resources, including Scheduling Operations, Recordset Filter Criteria, and Shifts from Pattern using REST endpoints. These REST APIs follow similar conventions as Connect REST APIs. Here's a list of Field Service API resources.

To understand the architecture, authentication, rate limits, and how the requests and responses work, see Connect REST API Developer Guide.

IN THIS SECTION:

Rebook and Notify Resources
Field Service Rebook and Notify Connect APIs are RESTful APIs.

Scheduling Operations Resources
Field Service Scheduling Operations Connect APIs are RESTful APIs.

Rebook and Notify Resources

Field Service Rebook and Notify Connect APIs are RESTful APIs.

IN THIS SECTION:

Rebook and Notify Customer
Rebook service appointments and notify the end customer for Agentforce Scheduling for Field Service.

Rebook and Notify Customer

Rebook service appointments and notify the end customer for Agentforce Scheduling for Field Service.

Resource

```
/connect/field-service/rebookNotifyAgentforceSchedulingFS
```

Resource Example

```
https://yourInstance.salesforce.com/services/data/v68.0/connect/field-service/rebookNotifyAgentforceSchedulingFS
```

Available Version

64.0

HTTP Methods

POST

Request Body for POST

JSON Example

```
{
  "flowDefinitionId": "3015g0000008XYZAA2",
  "operationType": "UpdateObj",
  "serviceAppointmentId": "08p5g0000004AbCEAU"
}
```

Properties

Name	Type	Description	Required or Optional	Available Version
`flowDefinitionID`	String	The ID of the FlowDefinition object.	Required	64.0
`operationType`	String	Defines the type of operation. Possible values are: • `CreateObj` • `DeleteObj` • `Invalid` • `UpdateObj`	Required	64.0
`serviceAppointmentId`	String	The ID of the service appointment that needs rescheduling.	Required	64.0

Response Body for POST

Rebook And Notify

Scheduling Operations Resources

Field Service Scheduling Operations Connect APIs are RESTful APIs.

IN THIS SECTION:

Fetch Optimization Files
Fetch the files generated for an optimization request in Enhanced Scheduling and Optimization.

Fetch Optimization Files

Fetch the files generated for an optimization request in Enhanced Scheduling and Optimization.

This resource requires Field Service to be enabled for the org and requires the context user to have Field Service enabled.

Resource

```
/connect/field-service/scheduling-operations/fetchOptimizationFiles
```

Resource Example

```
https://yourInstance.salesforce.com/services/data/v68.0/connect/field-service/scheduling-operations/fetchOptimizationFiles
```

Available Version

66.0

HTTP Methods

POST

Request Body for POST

JSON Example

```
{
  "optimizationRequestId": "0XxXX0000000001AAA"
}
```

Properties

Name	Type	Description	Required or Optional	Available Version
optimizationRequestId	String	ID of the optimization request for which to retrieve the generated files.	Required	66.0

Response Body for POST

Fetch Files

Request Bodies

Here's a list of Scheduling Operations API request bodies.

IN THIS SECTION:

Fetch Files Input
Input representation for retrieving files generated by an optimization request in Enhanced Scheduling and Optimization.

Rebook And Notify Input
Input representation for the rebooking and notifying.

Fetch Files Input

Input representation for retrieving files generated by an optimization request in Enhanced Scheduling and Optimization.

JSON Example

```
{
  "optimizationRequestId": "0XxXX0000000001AAA"
}
```

Properties

Name	Type	Description	Required or Optional	Available Version
optimizationRequestId	String	ID of the optimization request for which to retrieve the generated files.	Required	66.0

Rebook And Notify Input

Input representation for the rebooking and notifying.

JSON Example

```
{
  "flowDefinitionId": "3015g0000008XYZAA2",
  "operationType": "UpdateObj",
  "serviceAppointmentId": "08p5g0000004AbCEAU"
}
```

Properties

Name	Type	Description	Required or Optional	Available Version
flowDefinitionID	String	The ID of the FlowDefinition object.	Required	64.0
operationType	String	Defines the type of operation. Possible values are: • CreateObj • DeleteObj	Required	64.0

Name	Type	Description	Required or Optional	Available Version
		• `Invalid` • `UpdateObj`		
`serviceAppointmentId`	String	The ID of the service appointment that needs rescheduling.	Required	64.0

Response Bodies

Here's a list of Scheduling Operations response bodies.

IN THIS SECTION:

Created File
Represents a single file created during a fetch optimization files operation.

Fetch Files
Output representation for a fetch optimization files operation. Contains a map of the files created in the org for the specified optimization request.

Rebook And Notify
Output representation of the rebook and notify request.

Created File

Represents a single file created during a fetch optimization files operation.

JSON Example

```
{
  "contentVersionId": "string",
  "success": true
}
```

Properties

Property Name	Type	Description	Filter Group and Version	Available Version
`contentVersionId`	String	ID of the Content Version record created in the org for this file.	Small, 66.0	66.0
`success`	Boolean	Indicates whether the Content Version record was created successfully (`true`) or not (`false`).	Small, 66.0	66.0

Fetch Files

Output representation for a fetch optimization files operation. Contains a map of the files created in the org for the specified optimization request.

JSON Example

```
{
  "createdFiles": {
    "property1": {
      "contentVersionId": "string",
      "success": true
    },
    "property2": {
      "contentVersionId": "string",
      "success": true
    }
  }
}
```

Properties

Property Name	Type	Description	Filter Group and Version	Available Version
`createdFiles`	Map<String, Created File>	Map of file names to their corresponding file creation details. Each key is the name of a file associated with the optimization request. Each value contains the Content Version ID and a success indicator.	Small, 66.0	66.0

Rebook And Notify

Output representation of the rebook and notify request.

JSON Example

```
{
  "errorCode": 0,
  "errorMessage": "",
  "requestId": "0Xx5g0000008AbCEAU",
  "status": "Completed"
}
```

Properties

Property Name	Type	Description	Filter Group and Version	Available Version
`errorCode`	Integer	A unique code representing the reason for the failure.	Small, 64.0	64.0
`errorMessage`	String	The reason for the failure.	Small, 64.0	64.0
`requestId`	String	The ID of the ConvMessageSendActionRequest record created by the request that's used to track the message's progress.	Small, 64.0	64.0
`status`	String	Defines whether the process completed successfully. Possible values are: • `Completed`The operation successfully completed. • `Failed`The operation failed.	Small, 64.0	64.0

Field Service Metadata API

The following Metadata types are used in Field Service.

The following list links to Metadata API types used in Field Service.

- FieldServiceSettings

  Represents an organization's Field Service settings.

- Flow

  Represents the metadata associated with a flow, including its structure, logic, and run-time behavior.

  – Flow for Field Service on page 99

    Field Service exposes additional actionType values for the FlowActionCall Metadata type.

- Skill

  Represents the settings for a skill used for field service or to route chats to agents in Chat, such as the name of the skill and which agents the skills are assigned to.

- TimeSheetTemplate

  Represents a template for creating time sheets in Field Service. This type extends the Metadata metadata type and inherits its `fullName` field.

Flow for Field Service

Represents the metadata associated with a flow. Use Flow to create an application that takes users through a series of pages to query and update records in the database. You can also execute logic and provide branching capability based on user input to build dynamic applications.

FlowActionCall

Field Service exposes additional actionType values for the FlowActionCall Metadata type.

actionType

- **Field Type**: InvocableActionType (enumeration of type string)
- **Description**: Required. The action type. Valid values include:
  - `convertTimeZone`Converts the time from the current time zone to the requested time zone. This value is available in API version 62.0 and later.
  - `createEngagementTopic`Creates an engagement topic record that connects a given topic ID. This value is available in API version 63.0 and later.
  - `getAgentContext`Retrieves the content of the agent context. This value is available in API version 67.0 and later.
  - `getAppointmentBookingSlots`Retrieves available service appointment time slots. This value is available in API version 62.0 and later.
  - `getCurrentTime`Retrieves the current time for a given time zone. This value is available in API version 65.0 and later.
  - `getEngagementTopics`Retrieves a list of pairs from the engagement topics. This value is available in API version 68.0 and later.
  - `getFieldServiceObjectProperty`Retrieves the value of an object's field according to the value of another field. This value is available in API version 65.0 and later.
  - `scheduleServiceAppointment`Schedules a service appointment. This value is available in API version 62.0 and later.

Field Service Tooling API

The following Tooling objects are used in Field Service.

The following list links to Tooling API objects used in Field Service.

- CleanRule

  Represents a data integration rule that controls how a data service adds and updates data for existing records in an org.

- TimeSheetTemplate

  Represents a template for creating time sheets in Field Service.

Field Service Invocable Actions

Here's a list of Field Service invocable actions.

To understand more about invoking actions, see Invoking Actions.

IN THIS SECTION:

Convert Time Zone

Converts the time from the current time zone to the requested time zone. If the time doesn't contain a time zone, the value defined in Input Time Zone is used instead.

Create Engagement Topic

Creates an engagement topic record that connects a given topic ID. For example, a service appointment ID, to a messaging session ID. If an engagement topic for this connection already exists, no action is taken.

Get Agent Context

Retrieves the content of the agent context.

Get Service Appointment Time Slots

Retrieves available service appointment time slots based on the time zone, operating hours, scheduling policy and time zone name parameters.

Get Current Time

Gets the current time for a given time zone.

Get Engagement Topics

Retrieves a list of pairs from the engagement topics of the object ID (from the TopicId field) and its object name for the current messaging session.

Get Field Service Object Field

Retrieves the value of an object's field according to the value of another field. Relevant for Field Service objects only. For example, returns the ID of a Scheduling Policy according to a specific policy's name.

Schedule Service Appointment

Schedules a service appointment based on the scheduling policy ID and the service appointment ID.

Convert Time Zone

Converts the time from the current time zone to the requested time zone. If the time doesn't contain a time zone, the value defined in Input Time Zone is used instead.

This action is available in API version 62.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/convertTimeZone`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`time`	string	Required. The time that requires conversion. It can be with or without a time zone. The format for times that include a time zone is `yyyy-MM-dd'T'HH:mm:ss.SSSXXX`. For example, `2025-02-24T14:30:15.123-05:00`. The format for times that don't

Input	Type	Description
		include a time zone is `yyyy-MM-dd'T'HH:mm:ss.SSS`. For example, `2025-02-24T14:30:15.123`.
`outputTimeZone`	string	Required. The requested time zone. For example `UTC`.
`inputTimeZone`	string	Optional. The time zone to use if the time doesn't include a time zone. For example, `America/Los_Angeles`.

Outputs

Output	Type	Description
`dateTime`	string	The time after it's converted to the required time zone, in the format `yyyy-MM-dd'T'HH:mm:ss.SSSXXX`. For example, `2025-02-24T14:30:15.123Z`.

Example

Sample request body:

```
{
  "inputs" : [{
    "time" : "2025-02-24T14:30:15.123",
    "outputTimeZone" : "UTC",
    "inputTimeZone" : "America/Los_Angeles"
  }]
}
```

Sample response body:

```
{
  "actionName" : "convertTimeZone",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "dateTime" : "2025-02-24T22:30:15.123Z"
  }
}
```

Create Engagement Topic

Creates an engagement topic record that connects a given topic ID. For example, a service appointment ID, to a messaging session ID. If an engagement topic for this connection already exists, no action is taken.

This action is available in API version 63.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/createEngagementTopic`

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`topicId`	id	Required. The topic's ID.

Outputs

Output	Type	Description
`completionStatus`	boolean	Defines whether the engagement topic was created successfully.

Example

Sample request body:

```
{
  "inputs" : [{
    "topicId" : "08p000000000001AAA"
  }]
}
```

Sample response body:

```
{
  "actionName" : "createEngagementTopic",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "completionStatus" : true
  }
}
```

Get Agent Context

Retrieves the content of the agent context.

This action is available in API version 67.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/getAgentContext`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`contextDescription`	string	Optional. A description of the agent context.

Outputs

Output	Type	Description
`agentContextApex`	string	Specifies the agent context as an Apex object. This output is an Apex output type that contains nested arrays in JSON format.

Example

Sample request body:

```
{
  "inputs" : [{
    "contextDescription" : "Sample agent context description."
  }]
}
```

Sample response body:

```
{
  "actionName" : "getAgentContext",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "agentContextApex" : {
      "contextDescription" : "Sample agent context description.",
      "contextId" : "0XxRZ0000008iPeYAI",
      "content" : "Sample agent context content."
    }
  }
}
```

Get Service Appointment Time Slots

Retrieves available service appointment time slots based on the time zone, operating hours, scheduling policy and time zone name parameters.

This action is available in API version 62.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/getAppointmentBookingSlots`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`serviceAppointmentId`	string	Required. The ID of the service appointment to get available time slots for.
`operatingHoursId`	string	Required. The operating hours ID of the appointment to get time slots for.
`schedulingPolicyId`	string	Required. The scheduling policy ID of the appointment to get time slots for.
`timeZoneName`	string	Required. The name of the time zone to use when searching for and showing time slots.

Outputs

Output	Type	Description
`availableSlots`	string	The available time slots for the service appointment.

Example

Sample request body:

```
{
  "inputs" : [{
    "serviceAppointmentId" : "08p000000000001AAA",
    "operatingHoursId" : "0OH000000000001AAA",
    "schedulingPolicyId" : "0Vr000000000001AAA",
    "timeZoneName" : "America/Los_Angeles"
  }]
}
```

Sample response body:

```
{
  "actionName" : "getAppointmentBookingSlots",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "availableSlots" :
```

```
"[{\"startTime\":\"2026-08-28T17:00:00.000Z\",\"endTime\":\"2026-08-28T18:00:00.000Z\"}]"

  }
}
```

Get Current Time

Gets the current time for a given time zone.

This action is available in API version 65.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/getCurrentTime`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`timeZone`	string	Optional. The text representation of the time zone.

Outputs

Output	Type	Description
`currentTime`	string	The current time in the given time zone.

Example

Sample request body:

```
{
  "inputs" : [{
    "timeZone" : "America/Los_Angeles"
  }]
}
```

Sample response body:

```
{
  "actionName" : "getCurrentTime",
  "errors" : null,
```

```
  "isSuccess" : true,
  "outputValues" : {
    "currentTime" : "2025-02-24 14:30:15"
  }
}
```

Get Engagement Topics

Retrieves a list of pairs from the engagement topics of the object ID (from the TopicId field) and its object name for the current messaging session.

This action is available in API version 68.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/getEngagementTopics`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`salesforceInternal`	string	Optional. For internal use only.

Outputs

Output	Type	Description
`topicIdsAndObjectNames`	string	A list of pairs from the engagement topics of the object ID (from the TopicId field) and its object name for the current messaging session.

Example

Sample request body:

```
{
  "inputs" : [{
    "salesforceInternal" : "internal"
  }]
}
```

Sample response body:


```
{
  "actionName" : "getEngagementTopics",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "topicIdsAndObjectNames" :
"[{\"TopicId\":\"08qRZ0000008iReYAI\",\"ObjectName\":\"ServiceAppointment\"},{\"TopicId\":\"02iRZ0000006AeDYAU\",\"ObjectName\":\"Asset\"}]"

  }
}
```


Get Field Service Object Field

Retrieves the value of an object's field according to the value of another field. Relevant for Field Service objects only. For example, returns the ID of a Scheduling Policy according to a specific policy's name.

This action is available in API version 65.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/getFieldServiceObjectProperty`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`objectName`	string	Required. The developer name of the Field Service object. For example, SchedulingPolicy.
`fieldName`	string	Required. The name of the object's field used to find the relevant object. For example Name.
`fieldValue`	string	Required. The value of the specific field used to find the relevant object. For example, Customer First.
`fieldToReturn`	string	Required. The name of the field whose value you want to return. For example, Id.
`isCustomObject`	boolean	Required. Indicates whether the object is a custom object (`true`) or a standard object (`false`).

Outputs

Output	Type	Description
`fieldServiceStandardVariable`	string	Returns the value of an object's field as an Apex object, which contains a `textValue`, `integerValue`, or `booleanValue`.

Example

Sample request body:

```
{
  "inputs" : [{
    "objectName" : "SchedulingPolicy",
    "fieldName" : "Name",
    "fieldValue" : "Customer First",
    "fieldToReturn" : "Id",
    "isCustomObject" : false
  }]
}
```

Sample response body:

```
{
  "actionName" : "getFieldServiceObjectProperty",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "fieldServiceStandardVariable" : {
      "textValue" : "0Vr000000000001AAA",
      "integerValue" : null,
      "booleanValue" : null
    }
  }
}
```

Schedule Service Appointment

Schedules a service appointment based on the scheduling policy ID and the service appointment ID.

This action is available in API version 62.0 and later.

Supported REST HTTP Methods

URI: `/services/data/vXX.X/actions/standard/scheduleServiceAppointment`

Where `XX.X` is the latest API version.

Formats: JSON, XML

HTTP Methods: POST

Authentication: `Authorization: Bearer TOKEN`

Where `TOKEN` is a variable.

Inputs

Input	Type	Description
`serviceAppointmentId`	id	Required. The ID of the service appointment to schedule.
`schedulingPolicyId`	id	Required. The scheduling policy ID to use to schedule the service appointment.

Outputs

Output	Type	Description
`serviceAppointmentId`	string	If the scheduling succeeds, it returns the ID of the scheduled service appointment.

Example

Sample request body:

```
{
  "inputs" : [{
    "serviceAppointmentId" : "08p000000000001AAA",
    "schedulingPolicyId" : "0Vr000000000001AAA"
  }]
}
```

Sample response body:

```
{
  "actionName" : "scheduleServiceAppointment",
  "errors" : null,
  "isSuccess" : true,
  "outputValues" : {
    "serviceAppointmentId" : "08p000000000001AAA"
  }
}
```

Field Service Apex Reference

The Field Service Apex Reference provides the Field Service (FSL) namespace for programmatic access to Field Service functionality and the ConnectApi namespace for accessing Connect REST API data and building custom experiences in Salesforce.

IN THIS SECTION:

ConnectApi Namespace

The `ConnectApi` namespace (also called Connect in Apex) provides classes for accessing the same data available in Connect REST API. Use Connect in Apex to create custom experiences in Salesforce.

FSL Apex Namespace

The Field Service (FSL) namespace contains all classes, methods, Visualforce pages, and custom objects within the Field Service managed package. To allow access to namespace elements, assign the FSL custom permission set that's appropriate for the user's persona. For example, to allow a user to book appointments, assign the FSL Agent custom permission set.

SEE ALSO:

Install the Field Service Managed Package

`ConnectApi` Namespace

The `ConnectApi` namespace (also called Connect in Apex) provides classes for accessing the same data available in Connect REST API. Use Connect in Apex to create custom experiences in Salesforce.

For information about working with the `ConnectApi` classes, see Connect in Apex.

IN THIS SECTION:

OptimizationFiles Class

Fetch files associated with optimization requests for Enhanced Scheduling and Optimization. Requires Field Service to be enabled for the org and the running user to have Field Service enabled.

FieldService Class

Preview and create shifts from a pattern.

OptimizationFiles Class

Fetch files associated with optimization requests for Enhanced Scheduling and Optimization. Requires Field Service to be enabled for the org and the running user to have Field Service enabled.

Namespace

ConnectApi

OptimizationFiles Methods

These methods are for `OptimizationFiles`. All methods are static.

IN THIS SECTION:

FetchOptimizationFiles(fetchFilesInput)

Fetch the files generated for an optimization request in Enhanced Scheduling and Optimization.

`FetchOptimizationFiles(fetchFilesInput)`

Fetch the files generated for an optimization request in Enhanced Scheduling and Optimization.

API Version

66.0

Requires Chatter

No

Signature

```
public static ConnectApi.FetchFilesOutputRepresentation
FetchOptimizationFiles(ConnectApi.fetchFilesInput fetchFilesInput)
```

Parameters

fetchFilesInput
Type: `ConnectApi.fetchFilesInput`

The input representation containing the ID of the optimization request for which to retrieve the associated files.

Return Value

Type: `ConnectApi.FetchFilesOutputRepresentation`

Usage

Use FetchOptimizationFiles to retrieve the Content Version files produced for a completed optimization request. Each file in the response corresponds to a file created in the org and is keyed by filename in the returned map.

Example

```
ConnectApi.fetchFilesInput input = new ConnectApi.fetchFilesInput();
      input.optimizationRequestId = '0XxXXXXXXXXXXXX';

      ConnectApi.FetchFilesOutputRepresentation output =
      ConnectApi.OptimizationFiles.FetchOptimizationFiles(input);

      Map<String, ConnectApi.CreatedFile> files = output.createdFiles;

      for (String fileName : files.keySet()) {
      ConnectApi.CreatedFile file = files.get(fileName);
     System.debug(fileName + ': id=' + file.contentVersionId + ', success=' + file.success);

      }
```

FieldService Class

Preview and create shifts from a pattern.

Namespace

ConnectApi

FieldService Methods

These methods are for `FieldService`. All methods are static.

IN THIS SECTION:

createShiftsFromPattern(shiftsFromPatternInput, shiftPatternId)
Create up to 2,000 shifts from a pattern.

previewShiftsFromPattern(shiftsFromPatternInput, shiftPatternId)
Preview up to 2,000 shifts from a pattern.

createShiftsFromPattern(shiftsFromPatternInput, shiftPatternId)

Create up to 2,000 shifts from a pattern.

API Version

51.0

Requires Chatter

Yes

Signature

```
public static ConnectApi.ShiftsFromPattern
createShiftsFromPattern(ConnectApi.ShiftsFromPatternInput shiftsFromPatternInput, String
shiftPatternId)
```

Parameters

shiftsFromPatternInput
Type: `ConnectApi.ShiftsFromPatternInput`

A `ConnectApi.ShiftsFromPatternInput` object providing the pattern.

shiftPatternId
Type: String

ID of the shift pattern.

Return Value

Type: `ConnectApi.ShiftsFromPattern`

previewShiftsFromPattern(shiftsFromPatternInput, shiftPatternId)

Preview up to 2,000 shifts from a pattern.

API Version

51.0

Requires Chatter

Yes

Signature

```
public static ConnectApi.ShiftsFromPattern
previewShiftsFromPattern(ConnectApi.ShiftsFromPatternInput shiftsFromPatternInput,
String shiftPatternId)
```

Parameters

shiftsFromPatternInput
Type: `ConnectApi.ShiftsFromPatternInput`

A `ConnectApi.ShiftsFromPatternInput` object providing the pattern.

shiftPatternId
Type: String

ID of the shift pattern.

Return Value

Type: `ConnectApi.ShiftsFromPattern`

FSL Apex Namespace

The Field Service (FSL) namespace contains all classes, methods, Visualforce pages, and custom objects within the Field Service managed package. To allow access to namespace elements, assign the FSL custom permission set that's appropriate for the user's persona. For example, to allow a user to book appointments, assign the FSL Agent custom permission set.

IN THIS SECTION:

AdvancedGapMatrix Class
Represents a matrix of service resource IDs and graded time slots. An instance of this class is used only within the context of the results returned by the `GradeSlotsService` class.

AppointmentBookingService Class
Represents the appointment booking scheduling process in field service. Appointment booking returns the available slots for a service appointment, while considering scheduling policies, work rules, and service objectives. For example, we can schedule an appointment tomorrow between 9 and 11 AM, or next Monday between 4 and 6 PM.

AppointmentBookingSlot Class
Represents a booking slot available for a given service appointment. An instance of this class is part of the results returned by the `AppointmentBookingService` and `GradeSlotsService` classes.

GradeSlotsService Class
Represents the results shown in the Candidates quick action. Use the `GradeSlotsService` class to evaluate all possible slots where a given service appointment can be scheduled.

AppointmentInsightsResult Class
Represents the response returned by the `getAppointmentInsights` on page 167 method. AppointmentInsightsResult includes details about why a service appointment can't be added to the Gantt, including resource availability, blocked slots, and applicable blocking rules.

BlockingRule Class

Represents a global class that holds an array of blocking rules. . `BlockingRule` is a required parameter for `AppointmentInsightsResult` on page 131, which is the response class for the `getAppointmentInsights` on page 167 method of `ScheduleService` on page 164.

GroupData Class

Represents a global class that combines a group policy ID and a list of service territory IDs. `GroupData` is a required input parameter for `FSL.ScheduleJobsApi.getJob.setTerritory` on page 160.

OAAS Class

Contains all the methods used in the available optimization services, such as global or in-day optimizations, reshuffle operations, and resource schedule optimizations.

OAASRequest Class

Represents the request sent in the `OAAS.Optimize()` method. This class contains all the details of a global optimization call.

PolygonUtils Class

Represents the result of using the `getTerritoryIdByPolygons` method to return a service territory record ID using latitude and longitude geolocation values.

RecurringAppointmentsManager Class

Represents the result of using the `getRecurringAppointmentSlots` on page 153 method to return a pattern of appointments that recur weekly.

RecurringPattern Class

Represents a global sharing class that is a required parameter for `RecurringAppointmentsManager` on page 152 to return a pattern of appointments that recur weekly.

RecurringSequence Class

Represents the result of using the `getRecurringAppointmentsSlots` on page 153 method to return a pattern of appointments that recur daily, weekly, or monthly.

ResourceScheduleData Class

Contains all the details of a service resource's available slots for a given service appointment. An instance of this class is used only within the context of the results returned by the `AdvancedGapMatrix` class.

ScheduleJobsApi Class

Represents configurations to the Territory tab that is available for all scheduled jobs that are categorized under Enhanced Optimization.

ScheduleResult Class

Represents the result of scheduling a service appointment that is returned by the `ScheduleService.schedule` method.

ScheduleService Class

Represents the scheduling of a given service appointment in the best available slot based on the applied scheduling policy.

SchedulingOption Class

Provides the details of an individual slot available for scheduling. Represents a run-time object used within the context of the results returned in the `AdvancedGapMatrix` class.

TimeInterval Class

Holds an interval's start and end times. An instance of this class is used when capturing the start and end times of a scheduling horizon, during which a scheduling process runs. It considers scheduled jobs and is also used to represent the start and end times of an appointment slot or window.

AdvancedGapMatrix Class

Represents a matrix of service resource IDs and graded time slots. An instance of this class is used only within the context of the results returned by the `GradeSlotsService` class.

Namespace

FSL

Example

For an example of how to parse the class results, see GradeSlotsService Class.

IN THIS SECTION:

AdvancedGapMatrix Properties

AdvancedGapMatrix Properties

`AdvancedGapMatrix` contains the following properties.

IN THIS SECTION:

FSLOperationId

This field only applies to scheduling service appointments that don't use Enhanced Scheduling and Optimization (ESO). Populated when scheduling a chain of two service appointments with scheduling dependencies, also known as complex work. An asynchronous call is made to schedule both service appointments in the chain. This property contains the `FSL__FSL_Operation__c` ID.

partialResults

Relevant only when Limit Apex Operations is enabled in the Field Service Admin app. If the processing time of the `ScheduleService` is about to exceed the maximum CPU limits, it returns the results that were already calculated and halts the process. This list contains the details of the partial, processed results. All other parameters are the same.

resourceIDToScheduleData

A map that returns service resource IDs and their corresponding available slots stored in the `ResourceScheduleData` class.

Service

The service appointment for which the `AdvancedGapMatrix` was run.

FSLOperationId

This field only applies to scheduling service appointments that don't use Enhanced Scheduling and Optimization (ESO). Populated when scheduling a chain of two service appointments with scheduling dependencies, also known as complex work. An asynchronous call is made to schedule both service appointments in the chain. This property contains the `FSL__FSL_Operation__c` ID.

Signature

```
public Id FSLOperationId {get; set;}
```

Property Value

Type: Id

Usage

To learn more about the complex work feature, see Create Scheduling Dependencies Between Service Appointments.

partialResults

Relevant only when Limit Apex Operations is enabled in the Field Service Admin app. If the processing time of the `ScheduleService` is about to exceed the maximum CPU limits, it returns the results that were already calculated and halts the process. This list contains the details of the partial, processed results. All other parameters are the same.

Signature

```
public List<FSL.PartialResultsInfo> partialResults {get; set;}
```

Property Value

Type: List<FSL.PartialResultsInfo>

resourceIDToScheduleData

A map that returns service resource IDs and their corresponding available slots stored in the `ResourceScheduleData` class.

Signature

```
public Map<Id, FSL.ResourceScheduleData> resourceIDToScheduleData {get; set;}
```

Property Value

Type: Map<Id, FSL.ResourceScheduleData>

Service

The service appointment for which the `AdvancedGapMatrix` was run.

Signature

```
public Service {get; set;}
```

Property Value

Type: ServiceAppointment

AppointmentBookingService Class

Represents the appointment booking scheduling process in field service. Appointment booking returns the available slots for a service appointment, while considering scheduling policies, work rules, and service objectives. For example, we can schedule an appointment tomorrow between 9 and 11 AM, or next Monday between 4 and 6 PM.

Namespace

FSL

Usage

Before calling the `AppointmentBookingService` class, make sure that the parent work order and service appointment already exist.

Note: When scheduling a service appointment, the user scheduling the appointment must have one of four managed package permission sets: Field Service Admin, Field Service Dispatcher, Field Service Call Center, Field Service Self-Service. If you're using platform events to schedule appointments, you must explicitly configure the user so that the user has the correct permissions. Without the proper configuration, the platform event runs as the Automated Process system user and doesn't have the correct permissions to schedule an appointment. To learn more, see Configure the User and Batch Size for Your Platform Event Trigger.

The time zone is typically based on the location of the customer requesting the service. As a best practice, use the time zone specified on the service territory's operating hours.

If you want to return time slots ordered by grade or by date, use the `getSlots()` method that accepts a sort parameter; either `sortResultsBy` or `sortBy`.

When the `AppointmentBookingService` class is called, the *`ExactAppointment`* value on the associated work type is ignored. If exact appointments are needed, set this value to `true` in the parameters. If you want to respect the *`ExactAppointment`* of the work type, query it directly from the work type.

After receiving the slots, the developer can decide how to display, or manage, the slots. Typically, the slots are displayed to the customer, who selects a slot. After a slot is selected, the service appointment's `ArrivalWindowStartTime` and `ArrivalWindowEndTime` fields are updated with the slot's start and end times; for example:

```
FSL.AppointmentBookingSlot slot = slots[0];
sa.ArrivalWindowStartTime = slot.Interval.Start.addSeconds(tz.getOffset(slot.Interval.Start)
 / -1000);
sa.ArrivalWindowEndTime = slot.Interval.Finish.addSeconds(tz.getOffset(slot.Interval.Finish)
 / -1000);

update sa;
```

The `FSL.ScheduleService` method is then called so that the service appointment is scheduled correctly within the selected slot.

IN THIS SECTION:

AppointmentBookingService Methods

AppointmentBookingService Methods

`AppointmentBookingService` includes the following static methods.

IN THIS SECTION:

getSlots(serviceID, policyId, operatingHoursId, tz, exactAppointment)
Returns a list of `FSL.AppointmentBookingSlot` records.

getSlots(serviceID, policyId, calendar, tz, exactAppointment)
Returns a list of `FSL.AppointmentBookingSlot` records.

getSlots(serviceID, policyId, calendar, tz, sortResults, exactAppointment)
Returns a sorted list of `FSL.AppointmentBookingSlot` records.

getSlots(serviceID, policyId, calendar, tz, sortBy, exactAppointment)

Returns a sorted list of `FSL.AppointmentBookingSlot` records.

getABSlots(serviceID, policyId, calendar, tz, sortBy, exactAppointment)

Returns a list of `FSL.AppointmentBookingSlot` records that respect complex work dependencies for a chain of service appointments.

getSlots(serviceID, policyId, operatingHoursId, tz, exactAppointment)

Returns a list of `FSL.AppointmentBookingSlot` records.

Signature

```
public static List<FSL.AppointmentBookingSlot> getSlots(Id serviceID, Id policyId, Id operatingHoursId, System.TimeZone tz, Boolean exactAppointment)
```

Parameters

`serviceID`
Type: Id

Required. The ID of the service appointment being scheduled.

`policyId`
Type: Id

Required. The ID of the scheduling policy being used.

`operatingHoursId`
Type: Id

Required. The ID of the operating hours record used to determine time slot intervals.

Note: The *`TimeZone`* on the operating hours record is ignored.

`tz`
Type: System.TimeZone

Required. The time zone in which the slots are returned. This time zone must be the time zone of the service territory in which the service appointment is performed. If any other timezone is used, the appointment booking slots are converted to the service territory time zone at run time.

`exactAppointment`
Type: Boolean

Required. Specifies whether the result uses exact appointments or an arrival window. When the `getSlots()` method is called, the *`exactAppointment`* value on the work type is ignored.

Return Value

Type: List<FSL.AppointmentBookingSlot>

Usage

This method returns a list of appointment booking time slots available for a given service appointment. Results are returned in the time zone sent in the original request.

Appointment windows depend on the *operatingHoursId* value provided in the call. The *operatingHoursId* represents various slots offered to the customer during appointment booking (for example, Mondays 9 to 11 AM, 11 AM to 1 PM, 1 to 3 PM; Tuesdays 9 to 11 AM, 12 to 4 PM; and so forth.)

This method only returns available slots and doesn't schedule the service appointment. After the end user selects the desired slot, update the *ArrivalWindowStartTime* and *ArrivalWindowEndTime* properties on the service appointment and call the `FSL.ScheduleService` method.

To convert dateTime values to UTC before they're updated in the database, perform a time zone conversion.

Example

This example calls the `AppointmentBookingService` class using the Customer First scheduling policy and the Gold Appointments Calendar operating hours as the desired appointment slots.

```
// FSL.AppointmentBookingService
// The getSlots method returns a list of AppointmentBookingSlot objects.

ServiceAppointment sa = [SELECT Id, EarliestStartTime, DueDate FROM ServiceAppointment
                         WHERE Id='07r3F0000009dTSGEC'];
Id schedulingPolicyId = [SELECT Id FROM FSL__Scheduling_Policy__c
                         WHERE Name='Customer first' LIMIT 1].Id;
Id operatingHoursId   = [SELECT id FROM OperatingHours
                         WHERE name='Gold Appointments Calendar' LIMIT 1].Id;
Timezone tz = UserInfo.getTimeZone();

List<FSL.AppointmentBookingSlot> slots = FSL.AppointmentBookingService.GetSlots(sa.Id,
schedulingPolicyId, operatingHoursId, tz, false);

System.debug('Returned ' + slots.size() + ' appointment slots');
for(integer i=0; i<slots.size(); i++){
    system.debug('Slot:'+i+' Start: '+slots[i].Interval.Start+' Finish:'+
slots[i].Interval.Finish+' Grade: '+slots[i].Grade);
}
```

Here's an implementation suggestion for returning time slots in a different time zone. In this example, we're returning the time slots in the Europe/Jersey time zone. The code doesn't take daylight saving time into account, which can cause issues when daylight saving time begins or ends.

```
ServiceAppointment sa = [SELECT Id, EarliestStartTime,
DueDate,serviceterritory.operatinghours.timezone FROM ServiceAppointment WHERE
Id='08pTC0000000rCPYAY'];

Id schedulingPolicyId = [SELECT Id FROM FSLQA__Scheduling_Policy__c WHERE
Id='a0cTC000000JDOB' LIMIT 1].Id;

Id operatingHoursId = [SELECT id FROM OperatingHours WHERE Id='0OHTC0000000c0U' LIMIT
1].Id;

Timezone tz = TimeZone.getTimeZone(sa.serviceterritory.operatinghours.timezone);//timezone
 of the territory of the Service Appointment

List<FSLQA.AppointmentBookingSlot> slots = FSLQA.AppointmentBookingService.GetSlots(sa.Id,
 schedulingPolicyId, operatingHoursId, tz, false);
```

```
Timezone JerseyTZ = TimeZone.getTimeZone('Europe/Jersey');//desired output timezone

for(integer i=0; i<slots.size(); i++){
//Remove the time offset of the current timezone.
    slots[i].Interval.Start =
slots[i].Interval.Start.addSeconds(tz.getOffset(slots[i].Interval.Start) / -1000);
    slots[i].Interval.Finish =
slots[i].Interval.Finish.addSeconds(tz.getOffset(slots[i].Interval.Finish) / -1000);
//Add the time offset of the new timezone.
    slots[i].Interval.Start =
slots[i].Interval.Start.addSeconds(JerseyTZ.getOffset(slots[i].Interval.Start) / 1000);
    slots[i].Interval.Finish =
slots[i].Interval.Finish.addSeconds(JerseyTZ.getOffset(slots[i].Interval.Finish) / 1000);
}
```

getSlots(serviceID, policyId, calendar, tz, exactAppointment)

Returns a list of `FSL.AppointmentBookingSlot` records.

Signature

```
public static List<FSL.AppointmentBookingSlot> getSlots(Id serviceID, Id policyId,
OperatingHours calendar, System.TimeZone tz, Boolean exactAppointment)
```

Parameters

`serviceID`
Type: Id

Required. The ID of the service appointment being scheduled.

`policyId`
Type: Id

Required. The ID of the scheduling policy being used.

`calendar`
Type: OperatingHours object

Required. The *`calendar`* is used to determine time slot intervals.

Note: The *`TimeZone`* on the operating hours record is ignored.

`tz`
Type: System.TimeZone

Required. The time zone in which the slots are returned. This time zone must be the time zone of the service territory in which the service appointment is performed. If any other timezone is used, the appointment booking slots are converted to the service territory time zone at run time..

`exactAppointment`
Type: Boolean

Required. Specifies whether the result uses exact appointments or an arrival window. When the `getSlots()` method is called, the *`exactAppointment`* value on the work type is ignored.

Return Value

Type: List<FSL.AppointmentBookingSlot>

Usage

This method returns a list of appointment booking time slots for a given service appointment. Results are returned in the time zone sent in the original request. The appointment windows (9–11, 11–1, 1–3, and so on) depend on slots returned in a subquery of the OperatingHours object.

This method only returns available slots and does not schedule the service appointment. Update the *ArrivalWindowStartTime* and *ArrivalWindowEndTime* properties on the service appointment and call the `FSL.ScheduleService` method.

Perform a time zone conversion to convert dateTime values to UTC before they are updated in the database.

Example

This example shows the use of a subquery to retrieve time slots from *calendar*, an OperatingHours object.

```
// FSL.AppointmentBookingService
// The getSlots method returns a list of AppointmentBookingSlot objects.

ServiceAppointment sa = [SELECT Id, EarliestStartTime, DueDate FROM ServiceAppointment
                         WHERE Id='08p4F0000008sRMQAY'];
Id schedulingPolicyId = [SELECT Id FROM FSL__Scheduling_Policy__c
                         WHERE Name='Customer First' LIMIT 1].Id;
operatinghours oh = new operatinghours();
oh = [SELECT id, (SELECT EndTime, StartTime, Type, DayOfWeek FROM TimeSlots) FROM
OperatingHours
        WHERE name='AB Slot: 2 Hour Slots' LIMIT 1];
Timezone tz = UserInfo.getTimeZone();

List<FSL.AppointmentBookingSlot> slots = FSL.AppointmentBookingService.GetSlots(sa.Id,
schedulingPolicyId, oh, tz, false);

System.debug('Returned ' + slots.size() + ' appointment slots');
for(integer i=0; i<slots.size(); i++){
    system.debug('Slot:'+i+' Start: '+slots[i].Interval.Start+' Finish:'+
slots[i].Interval.Finish+' Grade: '+slots[i].Grade);
}
```

getSlots(serviceID, policyId, calendar, tz, sortResults, exactAppointment)

Returns a sorted list of `FSL.AppointmentBookingSlot` records.

Signature

```
public static List<FSL.AppointmentBookingSlot> getSlots(Id serviceID, Id policyId,
OperatingHours calendar, System.TimeZone tz, FSL.AppointmentBookingService.SortResultsBy
sortResults, Boolean exactAppointment)
```

Parameters

serviceID
Type: Id

Required. The ID of the service appointment being scheduled.

policyId
Type: Id

Required. The ID of the scheduling policy being used.

calendar
Type: OperatingHours object

Required. The *calendar* is used to determine time slot intervals.

Note: The *TimeZone* on the operating hours record is ignored.

tz
Type: System.TimeZone

Required. The time zone in which the slots are returned. This time zone must be the time zone of the service territory in which the service appointment is performed. If any other timezone is used, the appointment booking slots are converted to the service territory time zone at run time.

sortResults
Type: *FSL.AppointmentBookingService.SortResultsBy* enumeration

Required. Used to sort AppointmentBookingSlot results that are returned. Possible sort values are Grade, NoSort, and SlotDateTime.

exactAppointment
Type: Boolean

Required. Specifies whether the result uses exact appointments or an arrival window. When the `getSlots()` method is called, the *exactAppointment* value on the work type is ignored.

Return Value

Type: List<FSL.AppointmentBookingSlot>

Usage

This method returns a sorted list of appointment booking time slots for a given service appointment. Results are returned in the time zone sent in the original request. The appointment windows (9–11, 11–1, 1–3, and so on) depend on slots returned in a subquery of *calendar*, an OperatingHours object.

This method only returns available slots and doesn't schedule the service appointment. Update the *ArrivalWindowStartTime* and *ArrivalWindowEndTime* properties on the service appointment and call the `FSL.ScheduleService` method.

Perform a time zone conversion to convert dateTime values to UTC before they are updated in the database.

Example

This example sorts the returned slots by grade. When using a signature that contains a *calendar* parameter, use a subquery for time slots in the OperatingHours object.

```
// FSL.AppointmentBookingService
// The getSlots method returns a list of AppointmentBookingSlot objects.
```

```
ServiceAppointment sa = [SELECT Id, EarliestStartTime, DueDate FROM ServiceAppointment
                            WHERE Id='08p4F0000008sRMQAY'];
Id schedulingPolicyId = [SELECT Id FROM FSL__Scheduling_Policy__c
                            WHERE Name='Customer First' LIMIT 1].Id;
operatinghours oh = new operatinghours();
oh = [SELECT id, (SELECT EndTime, StartTime, Type, DayOfWeek FROM TimeSlots) FROM
OperatingHours
         WHERE name='AB Slot: 2 Hour Slots' LIMIT 1];
Timezone tz = UserInfo.getTimeZone();

FSL.AppointmentBookingService.SortResultsBy sortResults =
FSL.AppointmentBookingService.SortResultsBy.Grade;
List<FSL.AppointmentBookingSlot> slots = FSL.AppointmentBookingService.GetSlots(sa.Id,
schedulingPolicyId, oh, tz, sortResults, false);

System.debug('Returned ' + slots.size() + ' appointment slots');
for(integer i=0; i<slots.size(); i++){
    system.debug('Slot:'+i+' Start: '+slots[i].Interval.Start+' Finish:'+
slots[i].Interval.Finish+' Grade: '+slots[i].Grade);
}
```

getSlots(serviceID, policyId, calendar, tz, sortBy, exactAppointment)

Returns a sorted list of `FSL.AppointmentBookingSlot` records.

Signature

```
public static List<FSL.AppointmentBookingSlot> getSlots(Id serviceID, Id policyId,
OperatingHours calendar, System.TimeZone tz, String sortBy, Boolean exactAppointment)
```

Parameters

serviceID
Type: Id

Required. The ID of the service appointment being scheduled.

policyId
Type: Id

Required. The ID of the scheduling policy being used.

calendar
Type: OperatingHours object

Required. The *calendar* is used to determine time slot intervals.

Note: The *TimeZone* on the operating hours record is ignored.

tz
Type: System.TimeZone

Required. The time zone in which the slots are returned. This time zone must be the time zone of the service territory in which the service appointment is performed. If any other timezone is used, the appointment booking slots are converted to the service territory time zone at run time.

sortBy
Type: String

Required. Used to sort AppointmentBookingSlot results. Acceptable values are SORT_BY_GRADE, SORT_BY_DATE, and SORT_BY_NO_SORT.

exactAppointment
Type: Boolean

Required. Specifies whether the result uses exact appointments or an arrival window. When the `getSlots()` method is called, the *exactAppointment* value on the work type is ignored.

Return Value

Type: List<FSL.AppointmentBookingSlot>

Usage

This method returns a sorted list of appointment booking time slots for a given service appointment. Results are returned in the time zone sent in the original request. The appointment windows (9–11, 11–1, 1–3, and so on) depend on slots returned in a subquery of *calendar*, an OperatingHours object.

This method only returns available slots and doesn't schedule the service appointment. Update the *ArrivalWindowStartTime* and *ArrivalWindowEndTime* properties on the service appointment and call the `FSL.ScheduleService` method.

Perform a time zone conversion to convert dateTime values to UTC before they are updated in the database.

getABSlots(serviceID, policyId, calendar, tz, sortBy, exactAppointment)

Returns a list of `FSL.AppointmentBookingSlot` records that respect complex work dependencies for a chain of service appointments.

Signature

```
public static FSL.ABSlotsResponse getABSlots(Id serviceID, Id policyId, OperatingHours
calendar, System.TimeZone tz, String sortBy, Boolean exactAppointment)
```

Parameters

serviceID
Type: Id

Required. The ID of the service appointment being scheduled.

policyId
Type: Id

Required. The ID of the scheduling policy being used.

calendar
Type: OperatingHours object and related TimeSlot object

Required. The *calendar* is used to determine time slot intervals.

Note: The *TimeZone* on the operating hours record is ignored.

tz
Type: System.TimeZone

Required. The time zone in which the slots are returned. This time zone must be the time zone of the service territory in which the service appointment is performed. If any other timezone is used, the appointment booking slots are converted to the service territory time zone at run time.

sortBy
Type: String

Required. Used to sort the results. Acceptable values are SORT_BY_GRADE, SORT_BY_DATE, and SORT_BY_NO_SORT.

exactAppointment
Type: Boolean

Required. Specifies whether the result uses exact appointments or an arrival window. When this method is called, the *exactAppointment* value on the work type is ignored.

Return Value

Type: FSL.ABSlotsResponse

Usage

This method is used to retrieve appointment booking slots for a complex work chain of two appointments. This method respects the complex work setting **Use all-or-none scheduling for related appointments**.

The appointment booking slots returned by this method are for the service appointment that was sent to the API.

If Enhanced Scheduling and Optimization (ESO) is enabled:

- The method runs synchronously.
- The method always uses the ESO behavior, which is all-or-none for related appointments scheduling of complex work.
- The method is valid for a chain of up to five to appointments.

If ESO isn't enabled, this method is asynchronous. To examine results that the asynchronous method returns, use the streaming API and subscribe to MstCompletedChannel, the channel for the Field Service managed package.

This method only returns available slots and doesn't schedule the service appointment. Update the *ArrivalWindowStartTime* and *ArrivalWindowEndTime* properties on the service appointment and call the `scheduleExtended` method of the FSLScheduleService class.

Perform a time zone conversion to convert dateTime values to UTC before they're updated in the database.

Example

This code example retrieves data related to a specific service appointment, including details about the appointment, scheduling policy, and operating hours. It then uses this data to fetch available appointment slots based on the user's timezone and logs the details of these slots for debugging purposes.

```
// Retrieve a specific ServiceAppointment record by its ID
ServiceAppointment sa = [
    SELECT Id, EarliestStartTime, DueDate
    FROM ServiceAppointment
    WHERE Id='08p4F0000008sRMQAY'
];

// Retrieve the ID of a Scheduling Policy by its name
Id schedulingPolicyId = [
    SELECT Id
```

```
    FROM FSL__Scheduling_Policy__c
    WHERE Name='Customer First'
    LIMIT 1
].Id;

// Retrieve Operating Hours and its related Time Slots by name
OperatingHours oh = [
    SELECT Id,(SELECT EndTime, StartTime, Type, DayOfWeek,
FSL__Designated_Work_Boolean_Fields__c
    FROM TimeSlots)
    FROM OperatingHours
    WHERE Name='AB Slot: 2 Hour Slots'
    LIMIT 1
];

// Retrieve the current user's timezone
TimeZone tz = UserInfo.getTimeZone();

// Call a method to get available appointment slots
FSL.ABSlotsResponse ABSlotsResponse = FSL.AppointmentBookingService.getABSlots(sa.Id,
schedulingPolicyId, oh, tz, 'SORT_BY_DATE', false);

// Debug statement to log the number of returned slots
List<FSL.AppointmentBookingSlot> slots = ABSlotsResponse.Slots;
    System.debug('Returned ' + slots.size() + ' appointment slots');

// Loop through each slot and log its details
for (integer i = 0; i < slots.size(); i++) {
    System.debug('Slot:' + i + ' Start: ' + slots[i].Interval.Start + ' Finish: ' +
slots[i].Interval.Finish + ' Grade: ' + slots[i].Grade);
}
```

SEE ALSO:

Example: Subscribe to and Replay Events Using a Visualforce Page

AppointmentBookingSlot Class

Represents a booking slot available for a given service appointment. An instance of this class is part of the results returned by the `AppointmentBookingService` and `GradeSlotsService` classes.

Namespace

FSL

Example

The debug log shows the relevant parts of the result: `Grade` and `Interval`. The `BestSlotGrades` parameters are private and inaccessible via code.

```
AppointmentBookingSlot:
[
Grade=85.18518518518518518518518518518518,
```

```
Interval=TimeInterval:
[
2018-01-26 09:00:00,2018-01-26 11:00:00
]
]
```

IN THIS SECTION:

AppointmentBookingSlot Properties

AppointmentBookingSlot Properties

`AppointmentBookingSlot` contains the following properties.

IN THIS SECTION:

grade
The grade of the available appointment booking slot.

interval
The time frame of the returned appointment booking slot.

grade

The grade of the available appointment booking slot.

Signature

```
public Decimal grade {get; set;}
```

Property Value

Type: Decimal

interval

The time frame of the returned appointment booking slot.

Signature

```
public FSL.TimeInterval interval {get; set;}
```

Property Value

Type: FSL.TimeInterval

GradeSlotsService Class

Represents the results shown in the Candidates quick action. Use the `GradeSlotsService` class to evaluate all possible slots where a given service appointment can be scheduled.

Namespace

FSL

Usage

Note: When scheduling a service appointment, the user scheduling the appointment must have one of four managed package permission sets: Field Service Admin, Field Service Dispatcher, Field Service Call Center, Field Service Self-Service. If you're using platform events to schedule appointments, you must explicitly configure the user so that the user has the correct permissions. Without the proper configuration, the platform event runs as the Automated Process system user and doesn't have the correct permissions to schedule an appointment. To learn more, see Configure the User and Batch Size for Your Platform Event Trigger.

IN THIS SECTION:

GradeSlotsService Constructors

GradeSlotsService Methods

GradeSlotsService Constructors

The following are constructors for `GradeSlotsService`.

IN THIS SECTION:

GradeSlotsService(schedulingPolicyId, serviceAppointmentId)
Creates a new instance of the `GradeSlotsService` class using the specified scheduling policy ID and service appointment ID.

GradeSlotsService(schedulingPolicyId, serviceAppointmentId)

Creates a new instance of the `GradeSlotsService` class using the specified scheduling policy ID and service appointment ID.

Signature

```
public GradeSlotsService(Id schedulingPolicyId, Id serviceAppointmentId)
```

Parameters

`schedulingPolicyId`
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

`serviceAppointmentId`
Type: Id

The record ID of the service appointment being scheduled.

GradeSlotsService Methods

`GradeSlotsService` includes the following static method.

IN THIS SECTION:

getGradedMatrix(i_ResultsInUserTimeZone)
Returns a matrix of resource IDs and graded time slots. The information is similar to that shown in the Candidates quick action.

getGradedMatrix(i_ResultsInUserTimeZone)

Returns a matrix of resource IDs and graded time slots. The information is similar to that shown in the Candidates quick action.

Signature

```
public FSL.AdvancedGapMatrix getGradedMatrix(Boolean i_ResultsInUserTimeZone)
```

Parameters

i_ResultsInUserTimeZone
Type: Boolean

If `true`, all DateTimes are returned in the user's time zone. If `false`, all DateTimes are returned in UTC.

Return Value

Type: FSL.AdvancedGapMatrix

Usage

If Enhanced Scheduling and Optimization (ESO) is enabled, this method runs synchronously. If ESO isn't enabled, this method can be called with only one service appointment at a time, and runs asynchronously. To examine results that the asynchronous method returns, use the streaming API and subscribe to MstCompletedChannel, the channel for the Field Service managed package.

Example

This example illustrates how to parse the results in the `FSL.AdvancedGapMatrix` to extract the service resource ID, start and end times, and grade of each slot.

```
// FSL.GradeSlotsService class
// The getGradedMatrix method returns a matrix of resource id's AND graded time slots

Id serviceAppointmentId = '08p1N000000qN4sQAE';
Id schedulingPolicyId=[SELECT Id FROM FSL__Scheduling_Policy__c WHERE Name='Customer First'
 LIMIT 1].Id;

// GENERATE the graded time slots for the service appointment
FSL.GradeSlotsService mySlotService = new
FSL.GradeSlotsService(schedulingPolicyId,serviceAppointmentId);

// STORE the matrix of service resource id's and graded time slots
FSL.AdvancedGapMatrix myResultMatrix = mySlotService.getGradedMatrix(true);

Map<Id,FSL.ResourceScheduleData> mySRGradedTimeSlotMap =
myResultMatrix.ResourceIDToScheduleData;
for (Id thisresourceid : mySRGradedTimeSlotMap.keySet()){
    for (FSL.SchedulingOption thisso :
```

```
mySRGradedTimeSlotMap.get(thisresourceid).SchedulingOptions ) {
        system.debug('***** Resource Id' + thisresourceid);
        system.debug('***** Start - ' + thisso.Interval.Start);
        system.debug('***** Finish - ' + thisso.Interval.Finish);
        system.debug('****** Grade - ' + thisso.Grade);
    }
}
```

AppointmentInsightsResult Class

Represents the response returned by the `getAppointmentInsights` on page 167 method. AppointmentInsightsResult includes details about why a service appointment can't be added to the Gantt, including resource availability, blocked slots, and applicable blocking rules.

Namespace

FSL

Example

This code block represents the output of the `getAppointmentInsights` method from the Appointment Insights API. It provides a detailed result of an appointment insight operation. The result indicates there are no slots blocked.

```
AppointmentInsightsResult:[
blockedSlots=0,
blockingRules=(BlockingRule:[
    ruleName=Due Date,
    slotsBlockedByMultipleRules=0,
    slotsBlockedByRule=0
]),
horizonEndDate=2024-06-06 14:00:00,
horizonStartDate=2024-05-27 14:00:00,
operationTimeStamp=2024-06-16 14:00:53,
policyId=a0c8B00000638CMQAY,
resourcesEvaluated=0,
serviceAppointmentId=08p8B000000jCjBQAU,
serviceTerritoryId=0Hh8B000000HrctSAC
]
```

IN THIS SECTION:

Appointment Insights Result Properties

Appointment Insights Result Properties

`AppointmentInsightsResult` contains the following properties.

IN THIS SECTION:

blockedSlots

The total number of potential blocked slots returned by the `getAppointmentInsights` method.

blockingRules
Array of blocking rules with the rule name and number of slots

horizonStartDate
The horizon start date as calculated and used by the operation.

horizonEndDate
The horizon end date as calculated and used by the operation.

operationTimeStamp
The time the operation was executed in Coordinated Universal Time (UTC).

policyId
The default policy ID or the policy ID provided in the request.

resourcesEvaluated
The total number of service resources loaded by fetch data and evaluated by the `getAppointmentInsights` method.

serviceAppointmentId
The ID of the service appointment used for the request.

serviceTerritoryId
The service territory associated with the service appointment. If there is no territory for the service appointment, keep this field empty. This property is optional.

blockedSlots

The total number of potential blocked slots returned by the `getAppointmentInsights` method.

Signature

```
global Integer blockedSlots {get; set;}
```

Property Value

Type: Integer

blockingRules

Array of blocking rules with the rule name and number of slots

Signature

```
global List<BlockingRule> blockingRules {get; set;}
```

Property Value

Type: List<BlockingRule>

horizonStartDate

The horizon start date as calculated and used by the operation.

Signature

```
global Datetime horizonStartDate {get; set;}
```

Property Value

Type: Datetime

horizonEndDate

The horizon end date as calculated and used by the operation.

Signature

```
global Datetime horizonEndDate {get; set;}
```

Property Value

Type: Datetime

operationTimeStamp

The time the operation was executed in Coordinated Universal Time (UTC).

Signature

```
global Datetime operationTimeStamp {get; set;}
```

Property Value

Type: Datetime

policyId

The default policy ID or the policy ID provided in the request.

Signature

```
global Id policyId {get; set;}
```

Property Value

Type: Id

resourcesEvaluated

The total number of service resources loaded by fetch data and evaluated by the `getAppointmentInsights` method.

Signature

```
global Integer resourcesEvaluated {get; set;}
```

Property Value

Type: Integer

serviceAppointmentId

The ID of the service appointment used for the request.

Signature

```
global Id serviceAppointmentId {get; set;}
```

Property Value

Type: Id

serviceTerritoryId

The service territory associated with the service appointment. If there is no territory for the service appointment, keep this field empty. This property is optional.

Signature

```
global Id serviceTerritoryId {get; set;}
```

Property Value

Type: Id

BlockingRule Class

Represents a global class that holds an array of blocking rules. . `BlockingRule` is a required parameter for `AppointmentInsightsResult` on page 131, which is the response class for the `getAppointmentInsights` on page 167 method of `ScheduleService` on page 164.

Namespace

FSL

Usage

`BlockingRule` is the Apex class type for `blockingRules`, a required parameter for `FSL.AppointmentInsightsResult`.

IN THIS SECTION:

BlockingRule Properties

BlockingRule Properties

`BlockingRule` contains the following properties.

IN THIS SECTION:

ruleName
The name of the rule that is blocking the slot. When there is no rule name, use the rule type.

slotsBlockedByMultipleRules
The number of slots that are blocked by a combination of this rule and others.

slotsBlockedByRule
The number of slots that are blocked only by this rule.

ruleName

The name of the rule that is blocking the slot. When there is no rule name, use the rule type.

Signature

```
global String ruleName {get; set;}
```

Property Value

Type: String

slotsBlockedByMultipleRules

The number of slots that are blocked by a combination of this rule and others.

Signature

```
global Integer slotsBlockedByMultipleRules {get; set;}
```

Property Value

Type: Integer

slotsBlockedByRule

The number of slots that are blocked only by this rule.

Signature

```
global Integer slotsBlockedByRule {get; set;}
```

Property Value

Type: Integer

GroupData Class

Represents a global class that combines a group policy ID and a list of service territory IDs. `GroupData` is a required input parameter for `FSL.ScheduleJobsApi.getJob.setTerritory` on page 160.

Namespace

FSL

Usage

`GroupData` is a required input parameter for `FSL.ScheduleJobsApi.getJob.setTerritory`. `GroupData` has two input parameters: `groupPolicyId` and a list of `territoryIds`.

Example

This sample code shows you how to populate `GroupData`.

```
global class GroupData {
        public String groupPolicyId { get; set; }
        public List<String> territoryIds { get; set; }

        global GroupData(String groupPolicyId, List<String> territoryIds) {
            this.groupPolicyId = groupPolicyId;
            this.territoryIds = territoryIds;
        }
    }
```

See `FSL.ScheduleJobsApi.getJob.setTerritory` to see how to use `GroupData` with `setTerritory`.

IN THIS SECTION:

GroupData Properties

GroupData Properties

`GroupData` contains the following properties.

IN THIS SECTION:

groupPolicyId
The group policy of a job required by `FSL.ScheduleJobsApi.getJob.setTerritory`.

territoryIds
A list of service territory IDs required by `FSL.ScheduleJobsApi.getJob.setTerritory`.

`groupPolicyId`

The group policy of a job required by `FSL.ScheduleJobsApi.getJob.setTerritory`.

Signature

```
public String groupPolicyId { get; set; }
```

Property Value

Type: String

territoryIds

A list of service territory IDs required by `FSL.ScheduleJobsApi.getJob.setTerritory`.

Signature

```
public List<String> territoryIds { get; set; }
```

Property Value

Type: String

OAAS Class

Contains all the methods used in the available optimization services, such as global or in-day optimizations, reshuffle operations, and resource schedule optimizations.

Namespace

FSL

Usage

If you call OAAS APIs, running asynchronously, from the following methods, you receive a `Database.executeBatch` exception error message.

- `start` batch Apex method
- `execute` batch Apex method
- `future` methods

The limitation doesn't apply if you're using Enhanced Scheduling and Optimization. See Using Batch Apex & Future Methods in the *Apex Developer Guide*.

IN THIS SECTION:

OAAS Methods

OAAS Methods

`OAAS` includes the following methods.

IN THIS SECTION:

optimize(request)

Returns the optimization request ID of the global optimization process that was triggered. If the scheduling policy includes an in-day optimization Boolean that's set to true, the request triggers an in-day optimization.

reshuffle(serviceId, policyId)

Returns the optimization request ID of a reshuffle operation that has begun on a given service appointment.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSasFields, unschedulableServicesField, maxOptRuntime)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSasFields, unschedulableServicesField, maxOptRuntime, nowTimeOnSchedule)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSas, unschedulableServices, maxOptRuntime)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSas, unschedulableServices, maxOptRuntime, nowTimeOnSchedule)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

optimize(request)

Returns the optimization request ID of the global optimization process that was triggered. If the scheduling policy includes an in-day optimization Boolean that's set to true, the request triggers an in-day optimization.

Signature

```
public static Id optimize(FSL.OAASRequest request)
```

Parameters

request

Type: FSL.OAASRequest

The optimization request.

Return Value

Type: Id

Record ID of the optimization request.

Example

This example creates an instance of the `OAASRequest` class that holds all the details of the optimization call being initiated. Next, the example calls the optimize method and passes in the request.

```
DateTime start=Datetime.now();
DateTime finish=Datetime.now().addDays(3);

LIST<Id> lstServiceTerritories = new List<Id>();
lstServiceTerritories.add('0Hh0b000000cIwsCAE');

FSL.OAASRequest oaasRequest = new FSL.OAASRequest();
```

```
oaasRequest.allTasksMode = true;
oaasRequest.filterFieldAPIName = null;
oaasRequest.start = start;
oaasRequest.finish = finish;
oaasRequest.includeServicesWithEmptyLocation = false;
oaasRequest.locations = lstServiceTerritories;
oaasRequest.schedulingPolicyID = 'a0N4E0000031HKkUAM';

FSL.OAAS oaas = new FSL.OAAS();
id optRequest = oaas.optimize(oaasRequest);
```

reshuffle(serviceId, policyId)

Returns the optimization request ID of a reshuffle operation that has begun on a given service appointment.

Signature

```
public static Id reshuffle(Id serviceId, Id policyId)
```

Parameters

serviceId
Type: Id

The record ID of the service appointment that must be scheduled.

policyId
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

Return Value

Type: Id

Record ID of the optimization request.

Usage

The Reshuffle action is used when a high-priority service appointment must be scheduled within a full schedule. It runs a "mini-optimization" that attempts to reshuffle the schedule to accommodate the appointment.

To learn more about the Reshuffle action, see Reschedule Service Appointments.

Example

```
FSL.OAAS oaas = new FSL.OAAS();
id optRequest = oaas.reshuffle('08p4E000000M21CQAS', 'a0N4E0000031HKkUAM');
```

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSasFields, unschedulableServicesField, maxOptRuntime)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

Signature

```
public static Id resourceDayOptimization(Id resourceId, Id policyId, FSL.TimeInterval horizon, Boolean includeAllTasks, Boolean includeOnlyResourceFutureSA, Decimal radius, String candidateSasFields, String unschedulableServicesField, Decimal maxOptRuntime)
```

Parameters

resourceId
Type: Id

The record ID of the service resource whose schedule is being optimized.

policyId
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

horizon
Type: FSL.TimeInterval

The time frame used to run resource schedule optimization.

includeAllTasks
Type: Boolean

If `true`, all relevant service appointments within the time frame are considered during optimization. If `false`, only unscheduled service appointments are considered, and all scheduled service appointments are pinned (unmovable).

includeOnlyResourceFutureSA
Type: Boolean

If `true`, only service appointments that are already assigned to the service resource are considered during optimization. If `false`, service appointments assigned to other service resources are also considered.

radius
Type: Decimal

The suggested distance between required service appointments and adjacent appointments. Required appointments are defined by the `unschedulableServicesField` parameter. Resource schedule optimization tries to group service appointments so that their distance from a required appointment is less than this radius. Appointments that are further from a required appointment can still be scheduled, but are deprioritized. If `null`, the data is not filtered based on the radius.

candidateSasFields
Type: String

Boolean service appointment field that indicates which appointments are candidates to be scheduled.

unschedulableServicesField
Type: String

Boolean service appointment field that indicates whether a service appointment is required (pinned), meaning it must remain on the schedule during resource schedule optimization.

maxOptRuntime
Type: Decimal

Total time in seconds during which the optimization results must be returned. This parameter only enforces the optimization time, and does not include optimization queue or Apex job queue times. If `null`, the default value of 30 seconds is used.

Usage

To learn more about resource schedule optimization, see Optimize a Single Resource's Schedule.

Return Value

Type: Id

Record ID of the optimization request.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSasFields, unschedulableServicesField, maxOptRuntime, nowTimeOnSchedule)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

Signature

```
public static Id resourceDayOptimization(Id resourceId, Id policyId, FSL.TimeInterval horizon, Boolean includeAllTasks, Boolean includeOnlyResourceFutureSA, Decimal radius, String candidateSasFields, String unschedulableServicesField, Decimal maxOptRuntime, Datetime nowTimeOnSchedule)
```

Parameters

resourceId
Type: Id

The record ID of the service resource whose schedule is being optimized.

policyId
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

horizon
Type: FSL.TimeInterval

The time frame used to run resource schedule optimization.

includeAllTasks
Type: Boolean

If `true`, all relevant service appointments within the time frame are considered during optimization. If `false`, only unscheduled service appointments are considered, and all scheduled service appointments are pinned (unmovable).

includeOnlyResourceFutureSA
Type: Boolean

If `true`, only service appointments that are already assigned to the service resource are considered during optimization. If `false`, service appointments assigned to other service resources are also considered.

radius
Type: Decimal

The suggested distance between required service appointments and adjacent appointments. Required appointments are defined by the `unschedulableServicesField` and `nowTimeOnSchedule` parameters. Resource schedule optimization tries to group service appointments so that their distance from a required appointment is less than this radius. Appointments that are

further from a required appointment can still be scheduled, but are deprioritized. If `null`, the data is not filtered based on the radius.

candidateSasFields
Type: String

Boolean service appointment field that indicates which appointments are candidates to be scheduled.

unschedulableServicesField
Type: String

Boolean service appointment field that indicates whether a service appointment is required (pinned), meaning it must it must remain on the schedule during resource schedule optimization.

maxOptRuntime
Type: Decimal

Total time in seconds during which the optimization results must be returned. This parameter only enforces the optimization time, and does not include optimization queue or Apex job queue times. If `null`, the default value of 30 seconds is used.

nowTimeOnSchedule
Type: Datetime

An appointment whose scheduled start time is earlier than this time is considered required and isn't updated during resource schedule optimization. Because resource schedule optimization is asynchronous, this parameter indicates when optimization was initiated. For example, if `nowTimeOnSchedule` is set to April 17, 2018, 10:30, appointments with an earlier scheduled start time are considered required and excluded from resource schedule optimization.

Return Value

Type: Id

Record ID of the optimization request.

Usage

To learn more about resource schedule optimization, see Optimize a Single Resource's Schedule.

Example

To use this code sample, replace the ID placeholders—for example, *`Service Appointment ID`*—with record IDs from your org. Surround the IDs with single quotes: `'08p4E00000017Gq'`.

```
FSL.OAAS a = new FSL.OAAS();

//SET the horizon interval
DateTime start = DateTime.newInstanceGmt(DateTime.Now().dateGmt(),
Time.newInstance(0,0,0,0));
DateTime finish = start.addDays(3);
FSL.TimeInterval horizon = new FSL.TimeInterval(start,finish);

//SELECT the candidatesServices
List<ServiceAppointment> services = [SELECT Id FROM ServiceAppointment WHERE Id IN
    (SERVICE_APPOINTMENT_ID,SERVICE_APPOINTMENT_ID) ];
SET<Id> candidatesIds = new SET<Id>();
FOR(ServiceAppointment service : services) {
    candidatesIds.add(service.Id);
```

```
}

//SET the RSO required appointment services
Set<Id> requiredSaIds = new Set<Id>();
List<ServiceAppointment> services2 = [SELECT Id FROM ServiceAppointment WHERE
Id=SERVICE_APPOINTMENT_ID];
FOR(ServiceAppointment service : services2) {
    requiredSaIds.add(service.Id);
}

//START the RSO process
Id requestId = a.resourceDayOptimization(SERVICE_RESOURCE_ID,SCHEDULING_POLICY_ID,

horizon,false,true,50,candidatesIds,requiredSaIds,60,DateTime.newInstance(2018,1,0,0,0,0));
```

Where `SERVICE_APPOINTMENT_ID`, `SERVICE_RESOURCE_ID`, and `SCHEDULING_POLICY_ID` are variables.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSas, unschedulableServices, maxOptRuntime)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

Signature

```
public static Id resourceDayOptimization(Id resourceId, Id policyId, FSL.TimeInterval
horizon, Boolean includeAllTasks, Boolean includeOnlyResourceFutureSA, Decimal radius,
Set<String> candidateSas, Set<String> unschedulableServices, Decimal maxOptRuntime)
```

Parameters

resourceId
Type: Id

The record ID of the service resource whose schedule is being optimized.

policyId
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

horizon
Type: FSL.TimeInterval

The time frame used to run resource schedule optimization.

includeAllTasks
Type: Boolean

If `true`, all relevant service appointments within the time frame are considered during optimization. If `false`, only unscheduled service appointments are considered, and all scheduled service appointments are pinned (unmovable).

includeOnlyResourceFutureSA
Type: Boolean

If `true`, only service appointments that are already assigned to the service resource are considered during optimization. If `false`, service appointments assigned to other service resources are also considered.

radius
Type: Decimal

The suggested distance between required service appointments and adjacent appointments. Required appointments are defined by the *unschedulableServices* parameter. Resource schedule optimization tries to group service appointments so that their distance from a required appointment is less than this radius. Appointments that are further from a required appointment can still be scheduled, but are deprioritized. If `null`, the data is not filtered based on the radius.

candidateSas
Type: Set<Id>

Set of IDs of service appointments that are candidates for scheduling.

unschedulableServices
Type: Set<Id>

Set of IDs of service appointments that are required (pinned), meaning they must remain on the schedule during resource schedule optimization

maxOptRuntime
Type: Decimal

Total time in seconds during which the optimization results must be returned. This parameter only enforces the optimization time, and does not include optimization queue or Apex job queue times. If `null`, the default value of 30 seconds is used.

Usage

To learn more about resource schedule optimization, see Optimize a Single Resource's Schedule.

Return Value

Type: Id

Record ID of the optimization request.

resourceDayOptimization(resourceId, policyId, horizon, includeAllTasks, includeOnlyResourceFutureSA, radius, candidateSas, unschedulableServices, maxOptRuntime, nowTimeOnSchedule)

Returns the optimization request ID with the details of the optimization call. This method runs resource schedule optimization—the optimization of a single service resource's schedule—using the parameters provided.

Signature

```
public static Id resourceDayOptimization(Id resourceId, Id policyId, FSL.TimeInterval
horizon, Boolean includeAllTasks, Boolean includeOnlyResourceFutureSA, Decimal radius,
Set<String> candidateSas, Set<String> unschedulableServices, Decimal maxOptRuntime,
Datetime nowTimeOnSchedule)
```

Parameters

resourceId
Type: Id

The record ID of the service resource whose schedule is being optimized.

policyId
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

horizon
Type: FSL.TimeInterval

The time frame used to run resource schedule optimization.

includeAllTasks
Type: Boolean

If `true`, all relevant service appointments within the time frame are considered during optimization. If `false`, only unscheduled service appointments are considered, and all scheduled service appointments are pinned (unmovable).

includeOnlyResourceFutureSA
Type: Boolean

If `true`, only service appointments that are already assigned to the service resource are considered during optimization. If `false`, service appointments assigned to other service resources are also considered.

radius
Type: Decimal

The suggested distance between required service appointments and adjacent appointments. Required appointments are defined by the *unschedulableServices* and `nowTimeOnSchedule` parameters. Resource schedule optimization tries to group service appointments so that their distance from a required appointment is less than this radius. Appointments that are further from a required appointment can still be scheduled, but are deprioritized. If `null`, the data is not filtered based on the radius.

candidateSas
Type: Set<Id>

Set of IDs of service appointments that are candidates for scheduling.

unschedulableServices
Type: Set<Id>

Set of IDs of service appointments that are required, meaning they must remain on the schedule during resource schedule optimization. These appointments may be moved to a different time slot, but they will continue to comply with their Earliest Start Permitted and Due Date values.

maxOptRuntime
Type: Decimal

Total time in seconds during which the optimization results must be returned. This parameter only enforces the optimization time, and does not include optimization queue or Apex job queue times. If `null`, the default value of 30 seconds is used.

nowTimeOnSchedule
Type: Datetime

Optional. An appointment whose scheduled start time is earlier than this time is considered required and isn't updated during resource schedule optimization. Because resource schedule optimization is asynchronous, this parameter indicates when optimization was initiated. For example, if `nowTimeOnSchedule` is set to April 17, 2018, 10:30, appointments with an earlier scheduled start time are considered required and excluded from resource schedule optimization.

Return Value

Type: Id

Record ID of the optimization request.

Usage

To learn more about resource schedule optimization, see Optimize a Single Resource's Schedule.

Example

```
FSL.OAAS a = new FSL.OAAS();

//SET the horizon interval
DateTime start = DateTime.newInstanceGmt(DateTime.Now().dateGmt(),
Time.newInstance(0,0,0,0));
DateTime finish = start.addDays(3);
FSL.TimeInterval horizon = new FSL.TimeInterval(start,finish);

//SELECT the candidatesServices
List<ServiceAppointment> services = [SELECT Id FROM ServiceAppointment WHERE Id IN
('08p4E00000017Gp','08p4E00000017Go') ];
SET<Id> candidatesIds = new SET<Id>();
FOR(ServiceAppointment service : services) {
    candidatesIds.add(service.Id);
}

//SET the RSO required appointment services
Set<Id> requiredSaIds = new Set<Id>();
List<ServiceAppointment> services2 = [SELECT Id FROM ServiceAppointment WHERE
Id='08p4E00000017Gq'];
FOR(ServiceAppointment service : services2) {
    requiredSaIds.add(service.Id);
}

//START the RSO process
Id requestId =
a.resourceDayOptimization('0Hn4E0000004JRS','a1w4E000000Ac6S',horizon,false,true,50,
    candidatesIds,requiredSaIds,60,DateTime.newInstance(2018,1,0,0,0,0));
```

OAASRequest Class

Represents the request sent in the `OAAS.Optimize()` method. This class contains all the details of a global optimization call.

Namespace

FSL

IN THIS SECTION:

OAASRequest Properties

OAASRequest Properties

`OAASRequest` contains the following properties.

IN THIS SECTION:

allTasksMode
If `true`, all service appointments within the time frame are considered during optimization. If `false`, only unscheduled appointments are considered, and scheduled appointments are pinned (unmovable).

filterFieldAPIName
The API name of a Boolean field on the ServiceAppointment object. Only service appointments on which the Boolean field is set to `true` are optimized.

start
The start of the time frame in which the optimization process considers data for scheduling.

finish
The end of the time frame in which the optimization process considers data for scheduling.

includeServicesWithEmptyLocation
If `true`, service appointments both with and without an associated service territory are considered during optimization. If `false`, only appointments with an associated service territory are considered.

keepApptScheduled (Beta)
The API name of a Boolean field on the ServiceAppointment object. Only service appointments on which the Boolean field is set to `true` are never dropped from the schedule after global or in-day optimization. `keepApptScheduled`is only available for Enhanced Scheduling and Optimization.

locations
A list of IDs of service territories for which optimization will run.

numberOfServicesToSchedule
Not currently in use. Leave as `null`.

schedulingPolicyId
The record ID of the scheduling policy that is used to schedule the service appointment.

allTasksMode

If `true`, all service appointments within the time frame are considered during optimization. If `false`, only unscheduled appointments are considered, and scheduled appointments are pinned (unmovable).

Signature

```
public Boolean allTasksMode {get; set;}
```

Property Value

Type: Boolean

filterFieldAPIName

The API name of a Boolean field on the ServiceAppointment object. Only service appointments on which the Boolean field is set to `true` are optimized.

Signature

```
public String filterFieldAPIName {get; set;}
```

Property Value

Type: String

start

The start of the time frame in which the optimization process considers data for scheduling.

Signature

```
public Datetime start {get; set;}
```

Property Value

Type: Datetime

finish

The end of the time frame in which the optimization process considers data for scheduling.

Signature

```
public Datetime finish {get; set;}
```

Property Value

Type: Datetime

includeServicesWithEmptyLocation

If `true`, service appointments both with and without an associated service territory are considered during optimization. If `false`, only appointments with an associated service territory are considered.

Signature

```
public Boolean includeServicesWithEmptyLocation {get; set;}
```

Property Value

Type: Boolean

`keepApptScheduled` (Beta)

The API name of a Boolean field on the ServiceAppointment object. Only service appointments on which the Boolean field is set to `true` are never dropped from the schedule after global or in-day optimization. `keepApptScheduled`is only available for Enhanced Scheduling and Optimization.

Signature

```
public String keepApptScheduled {get; set;}
```

Property Value

Type: String

Usage

Use this property to ensure that appointments are optimized but not removed from the schedule. They can be moved to another resource or time slot.

There are four possible property value options:

- Valid field name: the field name is a valid Boolean field on the ServiceAppointment object Service appointments that are marked as "Keep Scheduled" won't be dropped from the schedule after global or in-day optimization.
- Invalid field name: the field doesn't exist on the Service Appointment object or isn't a Boolean type. An error message appears.
- Null: Service appointments are marked "Keep Scheduled" based on the org-level defaults defined in the global or in-day optimization settings.+
- NO_KEEP_APPT_SCHEDULED: Service appointments aren't marked as "Keep Scheduled".

locations

A list of IDs of service territories for which optimization will run.

Signature

```
public List<Id> locations {get; set;}
```

Property Value

Type: List<Id>

numberOfServicesToSchedule

Not currently in use. Leave as `null`.

Signature

```
public Integer numberOfServicesToSchedule {get; set;}
```

Property Value

Type: Integer

schedulingPolicyId

The record ID of the scheduling policy that is used to schedule the service appointment.

Signature

```
public Id schedulingPolicyId {get; set;}
```

Property Value

Type: Id

PolygonUtils Class

Represents the result of using the `getTerritoryIdByPolygons` method to return a service territory record ID using latitude and longitude geolocation values.

Namespace

FSL

Usage

Before calling the `PolygonUtils` class, make sure that you have existing map polygon records that are associated with a service territory.

If **Base service appointment territories on polygons** is enabled, global actions consume this class to populate the Service Territory field on service appointments. When standard FSL Global Actions are not being used, you can use the `getTerritoryIdByPolygons` method to populate the Service Territory lookup field on objects.

If `getTerritoryIdByPolygons` doesn't find a territory it returns `null`.

For more information, see Create and Manage Map Polygons.

IN THIS SECTION:

PolygonUtils Methods

PolygonUtils Methods

`PolygonUtils` includes the following static method.

IN THIS SECTION:

getTerritoryIdByPolygons(longitude, latitude)
Returns a service territory record ID from longitude and latitude values.

getAllPolygonsByLatLong(longitude, latitude)
Returns a list of IDs of all matching polygons that include the point with the longitude and latitude values.

getTerritoryIdByPolygons(longitude, latitude)

Returns a service territory record ID from longitude and latitude values.

Signature

```
static Id getTerritoryIdByPolygons(Double longitude, Double latitude)
```

Parameters

longitude
Type: Double

Required. The longitude value of the geolocation to be used to retrieve a service territory record ID via Polygon Classification.

latitude
Type: Double

Required. The latitude value of the geolocation to be used to retrieve a service territory record ID via Polygon Classification.

Return Value

Type: Id

Usage

This method returns a service territory record ID from longitude and latitude values. If a service appointment's geolocation matches more than one polygon, the appointment is assigned to either the highest or lowest-level territory in the hierarchy depending on your settings.

Example

This example retrieves a service territory record ID using latitude and longitude geolocation fields.

```
//fetch Service Appointment record(s)
List<ServiceAppointment> sas = [select id, latitude, longitude from serviceappointment
where appointmentnumber = 'SA-3600' limit 1];
If(!sas.isEmpty()){
    //retrieve the service territory record id from the getTerritoryIdByPolygons method
using geolocation fields from the first appointment in the list
    Id ServiceTerritoryId =
FSL.PolygonUtils.getTerritoryIdByPolygons(double.valueOf(sas[0].longitude),double.valueOf(sas[0].latitude));

    system.debug(ServiceTerritoryId);
}
```

getAllPolygonsByLatLong(longitude, latitude)

Returns a list of IDs of all matching polygons that include the point with the longitude and latitude values.

Signature

```
static List getAllPolygonsByLatLong(Double longitude, Double latitude)
```

Parameters

longitude
Type: Double

Required. The longitude value of the geolocation to be used to retrieve a service territory record ID using polygon classification.

latitude
Type: Double

Required. The latitude value of the geolocation to be used to retrieve a service territory record ID using polygon classification.

Return Value

Type: List <Id>

Usage

This method returns a list of polygon record IDs from longitude and latitude values. The method doesn't include sharing.

Example

This example retrieves a list of relevant polygons that include the point with the longitude and latitude values.

```
//fetch Service Appointment record(s)
Double longitude = 32.077213;
Double latitude = 34.792759;
    List<FSL__Polygon__c> relevantPolygons =
FSL.PolygonUtils.getAllPolygonsByLatLong
(longitude,latitude);
    system.debug(relevantPolygons);
```

RecurringAppointmentsManager Class

Represents the result of using the `getRecurringAppointmentSlots` on page 153 method to return a pattern of appointments that recur weekly.

Namespace

FSL

Usage

`RecurringAppointmentsManager` is an Apex class that takes the `RecurringPattern` on page 155 class as a required parameter, and returns a list of `RecurringSequence` on page 157 appointments that recur weekly.

IN THIS SECTION:

RecurringAppointmentsManager Methods

RecurringAppointmentsManager Methods

`RecurringAppointmentsManager` includes the following static method.

IN THIS SECTION:

getRecurringAppointmentsSlots

Returns a list of `RecurringSequence` on page 157 appointments.

getRecurringAppointmentsSlots

Returns a list of `RecurringSequence` on page 157 appointments.

Signature

```
static List getRecurringAppointmentsSlots
```

Parameters

`ServiceID`
Type: Id

Required. The ID of a service appointment that represents a recurring visit. This record defines the scheduling requirements and constraints for each appointment in the recurring visits sequence. All visits use the location, required resources, skill requirements, and other constraints that are associated with this service appointment.

`PolicyID`
Type: Id

Required. The policy to be used to get the relevant work rules and objectives for the operation.

`CalendarOperatingHoursId`
Type: Id

Required. The ID of the operating hours used to determine the structure of the slots returned by the API.

`RecurringPattern`
Type: RecurringPattern Class

Required. This class instance is sent as a parameter that contains all the required details from the recurring pattern.

`SchedulingOptionsCount`
Type: Integer

Required. Indicates how many sets of scheduling options the API returns. You can have between one and three sets of options for the recurring appointment. If you select three, but fewer than three sets of recurrence scheduling options are available, you receive however many options are found given your constraints.

Note: As of API version 60.0, *`SchedulingOptionsCount`* is hard coded to `3`.

Return Value

Type: List `RecurringSequence`

Usage

This method returns a list of recurring appointments to repeat weekly.

This method only returns a list of potential scheduling options in the `RecurringSequence` on page 157 class and doesn't schedule the recurring appointments. Similar to the `getSlots` method for AppointmentBookingService on page 117 , you use the response from `getRecurringAppointmentsSlots` to create a list of records to insert, and then schedule the appointments.

The time constraint fields on ServiceAppointment that you pass into the API are used for all appointments within the recurrence and determine the availability of scheduling options. Ensure that there's a significant difference between `EarliestStartTime` and `DueDate` fields. For example, if you must schedule six appointments, and your customer only has visiting hours on Monday, you need six weeks to schedule all visits. Therefore, make sure you set `EarliestStartTime` and `DueDate` to at least six weeks apart. Don't

set other time constraint fields on the appointment that are more restrictive, such as **ArrivalWindowStartTime** and `ArrivalWindowEndTime`, which can further limit scheduling options.

If you need service to occur within a given time frame, for example 8 AM to 10 AM on a Monday, set the `DaysOfWeek` to Monday. Then, set the time frame using the visiting hours record that has operating hours time slots during your required time frame.

See the ServiceAppointment object reference for more information on time constraint fields.

Example 1: Using an Execution Script

The following code sample creates an instance of the `RecurringPattern` on page 155 class. Then, it sets the properties for that recurring pattern. In this case, the code creates a recurring pattern that excludes weekends, and considers that the customer isn't available on Tuesdays, needs two visits per week, and requires six total visits. It requests that the API return three scheduling options.

```
//Fill the pattern object
FSL.RecurringPattern pattern = new FSL.RecurringPattern();
pattern.DaysOfWeek = new
Set<FSL.RecurringPattern.DaysOfWeek>{FSL.RecurringPattern.DaysOfWeek.Monday,
FSL.RecurringPattern.DaysOfWeek.Wednesday, FSL.RecurringPattern.DaysOfWeek.Thursday,
FSL.RecurringPattern.DaysOfWeek.Friday};
pattern.FrequencyType = FSL.RecurringPattern.FrequencyType.WEEKLY;
pattern.Frequency = 2;
pattern.NumberOfVisits = 6;

Integer schedulingOptionsCount = 3;
Id policyID = 'a1Lx00000004CUXEA2';
Id serviceId = '08px000000NzmOeAAJ';
Id operatingHoursId = '0OHx0000000D3ETGA0';

//Call the API
FSL.RecurringAppointmentSlots result =
FSL.RecurringAppointmentsManager.getRecurringAppointmentsSlots(serviceId , policyID,
operatingHoursId, schedulingOptionsCount, pattern);

//Handle the response of the API (example only)
for (Integer i=1 ; i<= result.recurringSequences.size() ; i++){
    System.debug('Sequence number: ' + i);
    System.debug('participatingResources details: /n' + result);
    System.debug('visitSchedulingOptions details: /n' + result);
    System.debug('averageObjectivesGrades details: /n' + result);
    System.debug('sequenceScore details: /n' + result);
    System.debug('firstPatternOccurrence details: /n' + result);
}
```

Example 2: Using a Method

This code sample uses a method to create the same recurring pattern.

```
//Using a method (example):
public callRecurringVisitsAPI(Id serviceID, Id policyID, Id calendarOperatingHoursId,
Integer SchedulingOptionsCount) {

//Fill the pattern object
RecurringPattern pattern = new FSL.RecurringPattern();
pattern.DaysOfWeek = new Set<RecurringPattern.DaysOfWeek>{RecurringPattern.DaysOfWeek.Monday,
```

```
 RecurringPattern.DaysOfWeek.Wednesday, RecurringPattern.DaysOfWeek.Thursday,
RecurringPattern.DaysOfWeek.Friday};

pattern.FrequencyType = RecurringPattern.FrequencyType.WEEKLY;
pattern.Frequency = 2;
pattern.NumberOfVisits = 6;

//Call the API
FSL.RecurringAppointmentSlots result =
RecurringAppointmentsManager.getRecurringAppointmentsSlots(serviceID, policyID,
calendarOperatingHoursId, SchedulingOptionsCount, pattern);
}

}
```

RecurringPattern Class

Represents a global sharing class that is a required parameter for `RecurringAppointmentsManager` on page 152 to return a pattern of appointments that recur weekly.

Namespace

FSL

Usage

`RecurringPattern`is a global sharing class that is a required parameter for `RecurringAppointmentsManager` on page 152. All properties must be complete to call the `RecurringAppointmentsManager` API.

Example

This example shows code for `RecurringPattern` and its properties.

```
global class RecurringPattern {
    global enum DaysOfWeek {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}

     global enum FrequencyType {DAILY, WEEKLY, MONTHLY}
     global Set<DaysOfWeek> DaysOfWeek { get; set; }
     global FrequencyType FrequencyType { get; set; }
     global Integer Frequency { get; set; }
     global Integer NumberOfVisits { get; set; }
}
```

IN THIS SECTION:

Recurring Pattern Properties

Recurring Pattern Properties

`RecurringPattern` contains the following properties.

IN THIS SECTION:

DaysOfWeek
The list of the days that can be used to book appointments, which can be derived from the visiting hours record associated with the service appointment..

Frequency
The number of appointments to book inside the `FrequencyType` time frame provided.

FrequencyType
The frequency of recurring appointments. Currently, only weekly frequency is supported.

NumberOfVisits
Total number of appointments to schedule.

DaysOfWeek

The list of the days that can be used to book appointments, which can be derived from the visiting hours record associated with the service appointment..

Signature

```
global Set<DaysOfWeek> DaysOfWeek { get; set; }
```

```
global enum<FSL.RecurringPattern> DaysOfWeek {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}
```

Property Value

Type: Set<enum>

Usage

You can set `DaysofWeek` manually. Alternately, you can derive the days the customer is available from the visiting hours record associated with the work order that is associated with the service appointment passed into the API.

Only one visit should be scheduled per day. For example, if you are scheduling a total of six visits and set `DaysOfWeek` to Monday and Wednesday, the API doesn't provide a response with more than one visit scheduled for Monday and one visit scheduled for Wednesday.

Frequency

The number of appointments to book inside the `FrequencyType` time frame provided.

Signature

```
global Integer Frequency { get; set; }
```

Property Value

Type: Integer

FrequencyType

The frequency of recurring appointments. Currently, only weekly frequency is supported.

Signature

```
global FrequencyType FrequencyType { get; set; }
```

Property Value

Type: Enum

NumberOfVisits

Total number of appointments to schedule.

Signature

```
global Integer NumberOfVisits { get; set; }
```

Property Value

Type: Integer

RecurringSequence Class

Represents the result of using the `getRecurringAppointmentsSlots` on page 153 method to return a pattern of appointments that recur daily, weekly, or monthly.

Namespace

FSL

Example

This example shows the contents of a returned RecurringSequence object.

```
global class RecurringAppointmentSlots {
    global ServiceAppointment rootServiceAppointment;
    global List<RecurringSequence> recurringSequences;
}

global class RecurringSequence {
    global List<EncapsulatedResource> participatingResources;
    global List<SchedulingOption> visitSchedulingOptions;
    global List<GradeSlotResult> averageObjectivesGrades;
    global Double sequenceScore;
    global List<TimeInterval> firstPatternOccurrence;
}
```

ResourceScheduleData Class

Contains all the details of a service resource's available slots for a given service appointment. An instance of this class is used only within the context of the results returned by the `AdvancedGapMatrix` class.

Namespace

FSL

Usage

For more information, see `AdvancedGapMatrix`.

IN THIS SECTION:

ResourceScheduleData Properties

ResourceScheduleData Properties

`ResourceScheduleData` contains the following properties.

IN THIS SECTION:

currentSlotsIndexInAB
Index of the slot.

resource
Not currently in use.

schedulingOptions
A list of all possible scheduling options for the service resource.

currentSlotsIndexInAB

Index of the slot.

Signature

```
public Integer currentSlotsIndexInAB {get; set;}
```

Property Value

Type: Integer

resource

Not currently in use.

Signature

```
public FSL.EncapsulatedResource resource {get; set;}
```

Property Value

Type: FSL.EncapsulatedResource

schedulingOptions

A list of all possible scheduling options for the service resource.

Signature

```
public List<FSL.SchedulingOption> schedulingOptions {get; set;}
```

Property Value

Type: List<FSL.SchedulingOption>

ScheduleJobsApi Class

Represents configurations to the Territory tab that is available for all scheduled jobs that are categorized under Enhanced Optimization.

Namespace

FSL

Usage

This API uses two methods, `getJob` and `setTerritory`. `getJob` passes a job to `setTerritory`, and `setTerritory` applies a `territoryID` and `groupPolicyID` to that job via GroupData on page 135.

IN THIS SECTION:

ScheduleJobsApi Methods

ScheduleJobsApi Methods

`ScheduleJobsApi` includes the following static methods.

IN THIS SECTION:

getJob
Passes a scheduled job to the`setTerritory` method, which modifies the territories.

setTerritory
Sets territories for a job returned by `getJob` and performs various validation checks.

`getJob`

Passes a scheduled job to the`setTerritory` method, which modifies the territories.

Signature

```
global static ScheduleJobsApi getJob(String jobName)
```

Parameters

jobName
Type: String

Required. The name of the job to update for service territory. The job name is case-sensitive.

Return Value

Type: ScheduleJobsApi(validateJobId(jobId))

Usage

This method passes a job to `setTerritory` on page 160, which updates the Territory tab. See setTerritory for sample code.

setTerritory

Sets territories for a job returned by `getJob` and performs various validation checks.

Signature

```
public void setTerritory(List<GroupData> groupList)
```

Parameters

groupList
Type: List<GroupData>

Required. A list of `GroupData` objects representing the territories and group policies to be associated with the job.

Return Value

Type: Void

Usage

This method configures the Territory tab based on optimized territory and scheduling policy for a scheduled job. Here are two examples of code that configures the territory tab for a scheduled job.

Example 1:

```
String jobName = 'Optimization';

List<FSL.ScheduleJobsApi.GroupData> groupList = new List<FSL.ScheduleJobsApi.GroupData>();

FSL.ScheduleJobsApi.GroupData firstGroup = new FSL.ScheduleJobsApi.GroupData
    ('a0cSM0000000fEU', new List<String>{'0HhSM0000000S5x', '0HhSM0000000TbV0AU'});

FSL.ScheduleJobsApi.GroupData secondGroup = new FSL.ScheduleJobsApi.GroupData
```

```
    ('0', new List<String>{'0HhSM0000000RbJ'});

groupList.add(firstGroup);
groupList.add(secondGroup);

FSL.ScheduleJobsApi.GetJob(jobName).setTerritory(groupList);
```

Example 2:

```
String jobName = "Optimization";

// Get all service territory groups.
List<YourGroupType> All_ST_GROUPS = getAllGroupsFromAnotherSource();

List<FSL.ScheduleJobsApi.GroupData> groupList = new List<FSL.ScheduleJobsApi.GroupData>();

for (Integer i = 0; i < All_ST_GROUPS.size(); i++) {
    FSL.ScheduleJobsApi.GroupData groupData = new FSL.ScheduleJobsApi.GroupData(
        All_ST_GROUPS[i].policyId,
        new List<String> {
            All_ST_GROUPS[i].territoryIds
        }
    );
    // Add the created groupData to the list.
    groupList.add(groupData);
}

FSL.ScheduleJobsApi.GetJob(jobName).setTerritory(groupList);
```

ScheduleResult Class

Represents the result of scheduling a service appointment that is returned by the `ScheduleService.schedule` method.

Namespace

FSL

Example

This example shows the contents of a returned ScheduledResult object.

```
[
 BreakObject=null,
 Grade=100.00000000000000000000000000000000,
 LongOperationId=null,
 PartialResults=(),
 Resource=ServiceResource:
 {Id=0Hn4E0000004bucSAA, Name=Crew A, IsActive=true, IsCapacityBased=false, ResourceType=C,
 ServiceCrewId=1cr4E0000004CM7QAM, Contractor_Priority__c=2},
 Service=ServiceAppointment:
 {Id=08p4E000000LeMiQAK, Status=None, FSL__Same_Day__c=false, FSL__Same_Resource__c=false,
 AppointmentNumber=SA-6214, DueDate=2018-09-20 17:07:00, EarliestStartTime=2018-09-17
```

```
17:07:00, Duration=1.5, DurationType=Hours, Latitude=37.793872000000000,
Longitude=-122.394865000000000, FSL__InternalSLRGeolocation__Latitude__s=37.793834,
FSL__InternalSLRGeolocation__Longitude__s=-122.395123, ServiceTerritoryId=0Hh4E0000000OtPSAU,
 FSL__Schedule_over_lower_priority_appointment__c=false, FSL__Use_Async_Logic__c=false,
FSL__IsMultiDay__c=false, ParentRecordId=0WO4E000000Vl6rWAC, FSL__Emergency__c=false,
SchedStartTime=2018-09-17 17:13:00, SchedEndTime=2018-09-17 18:43:00,
FSL__Schedule_Mode__c=Automatic, FSL__Scheduling_Policy_Used__c=a0N4E0000031HPVUA2}
 ]
```

IN THIS SECTION:

ScheduleResult Properties

ScheduleResult Properties

`ScheduleResult` contains the following properties.

IN THIS SECTION:

breakObject
Not currently in use.

grade
The grade of the slot in which the service appointment was scheduled.

longOperationId
This field only applies to scheduling service appointments that don't use Enhanced Scheduling and Optimization (ESO). Populated when scheduling a group of two dependent service appointments, also known as complex work. An asynchronous call is made to schedule both service appointments in the dependency group. This property contains the `FSL__FSL_Operation__c` ID.

partialResults
Relevant only when Limit Apex Operations is enabled in the Field Service Admin app. If the processing time of the `ScheduleService` is about to exceed the maximum CPU limits, partialResultsreturns the results that were already calculated and halts the process. This list contains the details of the partial, processed results.

serviceResource
The service resource assigned to the service appointment.

serviceAppointment
The scheduled service appointment.

breakObject

Not currently in use.

Signature

```
public ResourceAbsence breakObject {get; set;}
```

Property Value

Type: ResourceAbsence

grade

The grade of the slot in which the service appointment was scheduled.

Signature

```
public Decimal grade {get; set;}
```

Property Value

Type: Decimal

longOperationId

This field only applies to scheduling service appointments that don't use Enhanced Scheduling and Optimization (ESO). Populated when scheduling a group of two dependent service appointments, also known as complex work. An asynchronous call is made to schedule both service appointments in the dependency group. This property contains the `FSL__FSL_Operation__c` ID.

Signature

```
public Id longOperationId {get; set;}
```

Property Value

Type: Id

Usage

To learn more about scheduling dependencies and complex work, see Create Scheduling Dependencies Between Service Appointments.

To learn about Enhanced Scheduling and Optimization, see Get Ready for Scheduling and Optimization.

partialResults

Relevant only when Limit Apex Operations is enabled in the Field Service Admin app. If the processing time of the `ScheduleService` is about to exceed the maximum CPU limits, partialResultsreturns the results that were already calculated and halts the process. This list contains the details of the partial, processed results.

Signature

```
public List<FSL.PartialResultsInfo> partialResults {get; set;}
```

Property Value

Type: List<FSL.PartialResultsInfo>

serviceResource

The service resource assigned to the service appointment.

Signature

```
public ServiceResource serviceResource {get; set;}
```

Property Value

Type: ServiceResource

serviceAppointment

The scheduled service appointment.

Signature

```
public ServiceAppointment serviceAppointment {get; set;}
```

Property Value

Type: ServiceAppointment

ScheduleService Class

Represents the scheduling of a given service appointment in the best available slot based on the applied scheduling policy.

Namespace

FSL

Usage

This class calls the scheduling engine and schedules the given service appointment in the highest-scoring available slot.

When using Enhanced Scheduling and Optimization, calling this API triggers a callout to the Enhanced Scheduling and Optimization service to retrieve results. When not using Enhanced Scheduling and Optimization, you can make a callout to obtain required travel times when:

- SLR or point-to-point predictive travel is the selected routing, and
- Results aren't stored in the local cache

We recommend that you avoid performing any DML in the same Apex transaction before calling this API.

Note: When scheduling a service appointment, the user scheduling the appointment must have one of four managed package permission sets: Field Service Admin, Field Service Dispatcher, Field Service Call Center, Field Service Self-Service. If you're using platform events to schedule appointments, you must explicitly configure the user so that the user has the correct permissions. Without the proper configuration, the platform event runs as the Automated Process system user and doesn't have the correct permissions to schedule an appointment. To learn more, see Configure the User and Batch Size for Your Platform Event Trigger.

IN THIS SECTION:

ScheduleService Methods

ScheduleService Methods

`ScheduleService` includes the following static methods.

IN THIS SECTION:

schedule(policy, serviceId)

Returns `FSL.ScheduleResult` with the result of the scheduling process.

scheduleExtended(policy, serviceId)

Returns `List<FSL.ScheduleResult>` with the result of the scheduling process for appointments in a complex work chain.

getAppointmentInsights

Returns an AppointmentInsightsResult on page 131 class with details about why a service appointment can't be scheduled on the Gantt, including blocking rules, blocked slots, and resource availability.

scheduleHere(startDate, usedPolicyId, serviceAppointmentId, newResourceId)

Schedules a service appointment to a specific service resource at a designated time. To use this method, enable Enhanced Scheduling and Optimization. Otherwise, the method fails.

schedule(policy, serviceId)

Returns `FSL.ScheduleResult` with the result of the scheduling process.

Signature

```
public static FSL.ScheduleResult schedule(Id policy, Id serviceId)
```

Parameters

`policy`
Type: Id

The record ID of the scheduling policy being used to schedule the service appointment.

`serviceId`
Type: Id

The record ID of the service appointment being scheduled.

Return Value

Type: `FSL.ScheduleResult`

Usage

This method schedules the service appointment in the best available slot. If there are no available slots, the appointment isn't scheduled. This method can be called with only one service appointment at a time. To schedule multiple service appointments, use an Apex batch class. Call this method in batches of one.

If you are using the `schedule` method with the appointment booking method, perform a time zone conversion. The results of appointment booking are returned in the time zone specified in the method signature. Convert these values back to UTC.

Example

```
// FSL.ScheduleService class
// The Schedule method returns a ScheduleResult result
FSL.ScheduleResult myResult = new FSL.ScheduleResult();

// static FSL.ScheduleResult Schedule(SCHEDULING_POLICY_ID, SERVICE_APPOINTMENT_ID)
myResult = FSL.ScheduleService.schedule(SCHEDULING_POLICY_ID,SERVICE_APPOINTMENT_ID);

System.debug(myResult);
```

Where `SCHEDULING_POLICY_ID` and `SERVICE_APPOINTMENT_ID` are variables.

scheduleExtended(policy, serviceId)

Returns `List<FSL.ScheduleResult>` with the result of the scheduling process for appointments in a complex work chain.

Signature

```
public static List<FSL.ScheduleResult> scheduleExtended(Id policy, Id serviceId)
```

Parameters

`policy`
Type: Id

The record ID of the scheduling policy that's used to schedule the service appointment.

`serviceId`
Type: Id

The record ID of a service appointment in a complex work chain of service appointments being scheduled.

Return Value

Type: `List<FSL.ScheduleResult>`

Usage

Use this method to schedule two service appointments in a complex work chain. This method respects the complex work setting **Use all-or-none scheduling for related appointments**. If there are no available slots, appointments in the complex work chain aren't scheduled.

The `scheduleExtended` method is valid for a chain of two appointments; if the appointment in *`serviceId`* has dependencies with more than one other appointment, scheduling results can be different than expected.

This method can be called with only one service appointment at a time, and runs asynchronously. To examine results that the asynchronous method returns, use the streaming API and subscribe to MstCompletedChannel, the channel for the Field Service managed package.

If Enhanced Scheduling and Optimization (ESO) is enabled:

- The method runs synchronously.
- The method always uses the ESO behavior, which is **all-or-none** for related appointments scheduling of complex work.
- The method is valid for a chain of up to five to appointments.

If you're using the `scheduleExtended` method with the appointment booking method, perform a time zone conversion. The results of appointment booking are returned in the time zone specified in the method signature. Convert these values back to UTC.

SEE ALSO:

Example: Subscribe to and Replay Events Using a Visualforce Page

getAppointmentInsights

Returns an AppointmentInsightsResult on page 131 class with details about why a service appointment can't be scheduled on the Gantt, including blocking rules, blocked slots, and resource availability.

Signature

```
public static FSL.AppointmentInsightsResult<FSL.AppointmentInsightsResult>
getAppointmentInsights(Id policyId, Id serviceAppointmentId))
```

Parameters

serviceAppointmentId
Type: Id

The ID of the appointment for which insights are being requested.

policyId
Type: Id

The ID of the policy under which the appointment is evaluated.

Return Value

Type: `FSL.AppointmentInsightsResult`

Example

The following code sample uses the `getAppointmentInsights` method to return an `AppointmentInsightsResult` class that provides details about a specific service appointment that can't be scheduled on the Gantt.

```
// FSL.ScheduleService class
// The getAppointmentInsights method returns a AppointmentInsightsResult result
FSL.AppointmentInsightsResult myresult = new FSL.AppointmentInsightsResult();

// static FSL.AppointmentInsightsResult
myresult = FSL.ScheduleService.getAppointmentInsights(Service Appointment ID,Scheduling
Policy ID);

System.debug(myresult);
```

scheduleHere(startDate, usedPolicyId, serviceAppointmentId, newResourceId)

Schedules a service appointment to a specific service resource at a designated time. To use this method, enable Enhanced Scheduling and Optimization. Otherwise, the method fails.

Signature

```
global static List<FSL.ScheduleResult> scheduleHere(Datetime startDate, Id usedPolicyId,
Id serviceAppointmentId, Id newResourceId)
```

Parameters

startDate
Type: Datetime

The desired start date and time to schedule the service appointment. Use the getAppointmentCandidates results to select a time slot. The *startDate* value must fall within the getAppointmentCandidates scheduling horizon. The default horizon is 10 days and can be configured up to 30 days.

usedPolicyId
Type: Id

The ID of the scheduling policy to use for scheduling rules and optimization.

serviceAppointmentId
Type: Id

The ID of the service appointment to schedule.

newResourceId
Type: Id

The ID of the service resource to assign to the service appointment.

Return Value

Type: `List<FSL.ScheduleResult>`

Returns a list of `ScheduleResult` objects containing the scheduled service appointment details, including the scheduled service appointment with updated start and end times, the assigned service resource, the grade or score of the scheduling option, and additional scheduling metadata.

Usage

Use this method after calling the getAppointmentCandidates method to schedule a service appointment to a specific candidate slot and resource.After the service appointment's start time is set, the end time is calculated based on the service duration and the service resource efficiency. To prevent scheduling conflicts, the method performs resource locking and collision detection.

The method handles dependent service appointments in a complex work chain. For service appointments that are part of a complex work chain, all dependent appointments in the chain are scheduled synchronously, and the method returns a `ScheduleResult` for each. To schedule a chain explicitly, see scheduleExtended. If sliding and reshuffling is enabled, this method applies reshuffle logic during scheduling. A higher-priority incoming appointment can cause an existing appointment to be dropped. See reshuffle.

Example

```
Id serviceApptId = '08pxx000000001';
Id resourceId = '0Hnxx000000001';
Id policyId = '0Uyxx000000001';

Datetime startTime = Datetime.newInstance(2024, 12, 15, 9, 0, 0);
```

```
List<FSL.ScheduleResult> results = FSL.ScheduleService.scheduleHere(
    startTime,
    policyId,
    serviceApptId,
    resourceId
);

if (results != null && !results.isEmpty()) {
    FSL.ScheduleResult result = results[0];
    System.debug('Scheduled Service: ' + result.Service.Id);
    System.debug('Start: ' + result.Service.SchedStartTime);
    System.debug('End: ' + result.Service.SchedEndTime);
    System.debug('Resource: ' + result.Resource.Name);
    System.debug('Grade: ' + result.Grade);
}
```

SEE ALSO:

- Schedule Appointments from the Record Feed
- Scheduling Policies
- Enhanced Scheduling and Optimization

SchedulingOption Class

Provides the details of an individual slot available for scheduling. Represents a run-time object used within the context of the results returned in the `AdvancedGapMatrix` class.

Namespace

FSL

IN THIS SECTION:

- SchedulingOption Properties

SchedulingOption Properties

`SchedulingOption` contains the following properties.

IN THIS SECTION:

- grade
  The grade of the available slot.
- interval
  The time frame of the returned slot.

grade

The grade of the available slot.

Signature

```
public Decimal grade {get; set;}
```

Property Value

Type: Decimal

interval

The time frame of the returned slot.

Signature

```
public FSL.TimeInterval interval {get; set;}
```

Property Value

Type: FSL.TimeInterval

TimeInterval Class

Holds an interval's start and end times. An instance of this class is used when capturing the start and end times of a scheduling horizon, during which a scheduling process runs. It considers scheduled jobs and is also used to represent the start and end times of an appointment slot or window.

Namespace

FSL

IN THIS SECTION:

- TimeInterval Properties
- TimeInterval Methods

TimeInterval Properties

`TimeInterval` contains the following properties.

IN THIS SECTION:

start
The start time of the TimeInterval class.

finish
The end time of the TimeInterval class.

start

The start time of the TimeInterval class.

Signature

```
public Datetime start {get; set;}
```

Property Value

Type: Datetime

finish

The end time of the TimeInterval class.

Signature

```
public Datetime finish {get; set;}
```

Property Value

Type: Datetime

TimeInterval Methods

`TimeInterval` includes the following method.

IN THIS SECTION:

toString()
Returns the start and end datetimes as a string.

toString()

Returns the start and end datetimes as a string.

Signature

```
public String toString()
```

Return Value

Type: String

Field Service Custom Triggers

Field Service custom triggers ensure that the objects and respective fields are processed before or after scheduling.

The Field Service package provides triggers on Salesforce objects. Each trigger checks for various conditions and then performs tasks based on what it finds. Some of the triggers run on custom objects and some on standard objects.

Table 1: Field Service Quick Reference for Triggers

Trigger Name	Host Object	Object Type
FSL__TR021_AssignedResource.trigger	Assigned Resource	Standard Object
FSL__TR004_Event.trigger	Event	Standard Object
FSL__TR030_GanttPalette.trigger	Gantt Palette	Custom Object
FSL__TR029_GanttFilter.trigger	Gantt Filter	Custom Object
FSL__TR034_OperatingHours.trigger	Operating Hours	Standard Object
FSL__TR013_OptimizationRequest.trigger	Optimization Request	Custom Object
FSL__TR028_Polygon.trigger	Map Polygon	Custom Object
FSL__TR007_ResourceAbsence.trigger	Resource Absence	Standard Object
FSL__TR010_SchedulingPolicy.trigger	Scheduling Policy	Custom Object
FSL__SchedulingPolicyWorkRule.trigger	Scheduling Policy Work Rule	Custom Object
FSL__TR001_Service.trigger	Service Appointment	Standard Object
FSL__TR008_ServiceResource.trigger	Service Resource	Standard Object
FSL__TR012_Capacity.trigger	Service Resource Capacity	Standard Object
FSL__TR066_ServiceObjective.trigger	Service Objective	Custom Object
FSL__TR025_ServiceResourceSkill.trigger	Service Resource Skill	Standard Object
FSL__TR0023_ServiceTerritory.trigger	Service Territory	Standard Object
FSL__TR020_ResourceTerritories.trigger	Service Territory Member	Standard Object
FSL__TR0024_ServiceTerritory.trigger	Skill Requirement	Standard Object
FSL__TR051_TimeDependency.trigger	Time Dependency	Custom Object
FSL__TR027_TimeSlot.trigger	Time Slot	Standard Object
FSL__TR005_UserTerritory.trigger	User Territory	Custom Object
FSL__TR022_WorkOrder.trigger	Work Order	Standard Object
FSL__TR026_WorkRule.trigger	Work Rule	Custom Object
FSL__TR022_WorkOrderLineItem.trigger	Work Order Line Item	Standard Object

Assigned Resource

Before Insert:

- Stops the trigger execution if one of the created resources is crew members. This situation is handled in a different flow.
- Fails the trigger if the inserted assigned resources don't have the scheduled start and schedule end time fields populated.
- Fails the trigger if the created resources don't have a valid service territory member assigned to them.

- Populates the created resource's ServiceCrewId field if the resources are from type Crew.

After Insert:

- If the Enable User Territories sharing setting is enabled, this trigger shares the created service appointment with the service territory public group. This functionality provides sharing to the dispatchers that are part of the service appointment territory. If the service resource is relocated, the appointment is shared with the public group of the new territory, ensuring dispatchers have access to relevant service appointments.
- If the Enable User Territories sharing setting is enabled and the respective service appointment parent sharing setting is enabled, this trigger shares the created service appointment's parent with the service territory public group. This action provides sharing to the dispatchers that are part of the service appointment territory. To learn more, see Limit Access to Field Service Records.
- If the status categories change to Dispatched, this trigger shares the created resource's service appointments with the service resource's users. If the service territories change, this trigger shares with the service territory public groups.
- Mentions the assigned resource user if the relevant setting is enabled.
- Performs this operation in an asynchronous method if possible.
- Stops the trigger execution if one of the created resources is crew members. This behavior is handled in a different flow.
- Updates or creates a calendar event based on the created resource.
- If the assigned resources' service statuses are either None or Canceled, this trigger changes their statuses to Assigned.
- If the created resources' service is a multi-day service appointment, this trigger checks if it's necessary to recalculate the service duration. If so, the trigger recalculates the length.
- If the travel trigger is enabled in the org, this trigger recalculates travel for all services scheduled on the same day as the services' start and end dates.
- Creates assigned resources for crew members. If the resource in the trigger is for the resource type crew, we create a single resource per crew member as well.
- Updates the ResourceCapacity object according to the resources' service appointments. If a service appointment is scheduled to a contractor, this trigger updates their capacity to reflect the actual scheduled working time and working items allocated.

Before Update:

- Fails the trigger if the created Assigned Resources record doesn't have a valid Service Territory Member assigned to them.
- Populates the ServiceCrewId field if the service resources are from type crew.

After Update:

- If the Enable User Territories sharing setting is enabled, this trigger shares the created service appointment with the service territory public group. This functionality provides sharing to the dispatchers that are part of the service appointment territory. If the service resource is relocated, the appointment is shared with the public group of the new territory, ensuring dispatchers have access to relevant service appointments.
- If the service appointment is a multi-day appointment, this trigger checks if there's a need to recalculate the service duration. If so, this trigger recalculates the length.
- Updates or creates a calendar event based on the assigned resources updated.
- Creates and deletes assigned resources based on the resources' updates. If an assigned resource in the trigger is for the service resource of type crew, we create a single assigned resource per crew member as well.
- Updates the Resource Capacity object according to the assigned resources' service appointments. If an assigned resource was scheduled to a contractor, we update their capacity to reflect the actual scheduled working time and working items allocated.
- If the Service Resource field was changed on the assigned resource update, this trigger removes old sharing and creates sharing according to the service resource's and the dispatcher's territories.

- If the Make assigned resources followers of service appointments that are Dispatched or In Progress setting is enabled in the settings page, and the respective service appointment Status Category field was updated, this trigger makes the updated assigned resources followers of the service appointments and parents.
- If the travel trigger is enabled in the org, this trigger recalculates travel for all services scheduled in the same day of the services' start and end dates.
- Updates the schedule mode according to the actual scheduling operation (that is, Optimization, Automatic, Manual).

Before Delete:

- Removes sharing from the delete assigned resources' service appointments.
- If the deleted assigned resource is a resource of the type crew, this trigger deletes the respective crew members appointment resources.

After Delete:

- If the travel trigger is enabled in the org, this trigger recalculates travel for all services scheduled in the same day of the deleted assigned resources' services' start and end dates.
- Updates the Resource Capacity object according to the deleted assigned resources' service appointments. If a service appointment was unscheduled from a contractor, update their capacity to reflect the actual scheduled working time and working items allocated.
- If the Make assigned resources followers of service appointments that are Dispatched or In Progress setting is enabled in the settings page, this trigger unfollows the deleted assigned resources' service appointments and service appointments' parents.
- Nullifies the deleted assigned resources' Service Appointments' scheduled start and end time, and changes the status to None.
- If one of two Follow Immediately chains were unscheduled, that is, the assigned resource got deleted, this trigger unschedules the respective service appointment in the chain. To learn more, see Schedule an Appointment That Immediately Follows Another.

Event

All triggers for this object are part of the Calendar Sync feature, which is described in Calendar Sync's Knowledge Article.

Before Insert:

- Creates Resource Absences, Work Orders, Service Appointments, and Assigned Resources according to the settings described in Calendar Sync's Knowledge Article.
- Calendar Sync's settings are located in the Field Service settings page. From the settings page, select **Sharing** > **Calendar Sync**.

After Update:

This file is called FSL__TR004_Event_BeforeUpdate but fires only after update.

- If one or more of the following fields has changed, this trigger updates the Salesforce records accordingly: OwnerId, Subject, Location, StartDateTime, EndDateTime.
- If the FSL__Event_Type__c field has changed and the new value is valid, this trigger creates the Salesforce records accordingly.

After Delete:

This file is called FSL__TR004_Event_BeforeDelete but fires only after delete.

- If the FSL__Event_Type__c field matches the 'Calendar Event type' setting of Service Appointment, and the service hasn't been updated, then this trigger deletes the Assigned Resource related to that service. This trigger unschedules the Service Appointment.
- If the FSL__Event_Type__c field matches the 'Calendar Event type' setting of Resource Absence, then this trigger deletes the related absence.

Gantt Filter

After Insert:

- If the Make this filter available for all users checkbox is checked, this trigger creates a Share record for the FSL__Gantt_Filter__c and AllInternalUsers group.

After Update:

- If the Make this filter available for all users checkbox is unchecked in the update, then this trigger deletes its share record.
- If the checkbox is checked in the update, then this trigger adds a sharing record as described in After Insert.

Gantt Palette

After Insert:

- Add sharing to the AllInternalUsers group.

After Update:

- Add sharing to the AllInternalUsers group if no such sharing exists already.

Map Polygon

Before Insert:

- Sets the minimum and maximum Latitude and Longitude of the polygon to the designated fields.

Before Update:

- Sets the minimum and maximum Latitude and Longitude of the polygon to the designated fields.

Operating Hours

Before Delete:

- Validates that the record deleted isn't the default operating hours in appointment booking settings. From the Field Service settings page, select **Global Actions** > **Appointment Booking**.

After Update:

- If the Appointment Bundling feature is enabled, and the Use Bundle Apex Mode custom setting equals 2, the serviceTerritoryRefresh API http request is triggered.

Optimization Request

Before Insert:

- Sets the FSL__Text_Status__c text field with the value of the FSL__Status__c picklist field. The FSL__Text_Status__c is a text field marked as an External ID and is used to improve query performance.

Before Update:

- Prevents the FSL__Status__c picklist field from changing from In Progress to Queued. This trigger returns the status to In Progress.
- Validates that a record with the FSL__Status__c value of Aborted doesn't change to Completed. If it does, this trigger prevents the change and throws an error.
- If the status is changed to Aborted, this trigger checks the following conditions:

  - The change wasn't caused by the optimization user.
  - The Optimization Request wasn't aborted by the auto-kill feature. That is, the feature that kills requests that stay in a specific status more than what's defined in the custom settings.
  - The user has Abort_Optimization_Request custom permission on their profile or one of their permission sets.

If all the previously mentioned criteria are met, the request is aborted.

Resource Absence

Before Insert:

- Changes the scale of the geolocation values (latitude and longitude) to 6 digits after the decimal point.
- Removes seconds and milliseconds from the resource absence DateTime fields.
- Fills the InternalSLRGeolocation fields (latitude and longitude) based on the geolocation values. The InternalSLRGeolocation fields are used in street-level routing and predictive route calculations.

After Insert:

- If the inserted resource absence is the type nonavailability then:
  - If a multi-day service appointment was affected by the resource absence and there's just one service appointment, this trigger updates its duration.
  - If Fix Overlaps is enabled in the org and Travel Trigger is disabled, this trigger calls Fix Overlaps.
  - If Travel Trigger is enabled in the org and no multi-day appointment was updated in step 1, this trigger recalculates travel for all services scheduled in the same day of the absences' start and end dates.
  - If the calendar sync feature is turned on then the trigger creates a calendar event for the resource absence.
- Otherwise, this trigger does nothing.

Before Update: (content is the same as Before Insert)

- Changes the scale of the geolocation values (latitude and longitude) to 6 digits after the decimal point.
- Removes seconds and milliseconds from the resource absence's DateTime fields.
- Fills the InternalSLRGeolocation fields (latitude and longitude) based on the geolocation values. The InternalSLRGeolocation fields are used in street-level routing and predictive routes' calculations. We don't check for a change in the geolocation values—we update them like in the Before Insert trigger.

After Update:

- If the updated resource absence is the type non availability then:
  - If a multi-day service appointment was affected by the resource absence change and there's just one service appointment, this trigger updates its duration.
  - If Fix Overlaps is enabled in the org and Travel Trigger is disabled, this trigger calls Fix Overlaps.
  - If Travel Trigger is enabled in the org and no multi-day appointment was updated in step 1, this trigger recalculates travel for all services scheduled in the same day of the absences' start and end dates.
  - If the calendar sync feature is turned on then the trigger creates a calendar event for the resource absence.
- Otherwise, this trigger does nothing.

Before Delete:

- Deletes the matching calendar event if it exists. See the Calendar Sync's Knowledge Article for more information.

After Delete:

- If the deleted resource absence is the type nonavailability then:
  - If a multi-day service appointment was affected by the resource absence deletion and there's just one service appointment, this trigger updates its duration.
  - If Fix Overlaps is enabled in the org and Travel Trigger is disabled, this trigger calls Fix Overlaps.
  - If Travel Trigger is enabled in the org and no multi-day appointment was updated in step 1, this trigger recalculates travel for all services scheduled in the same day of the absences' start and end dates.

Scheduling Policy

Before Insert:

- Validates that the Travel Mode checkbox is checked only when this feature is enabled in the org.
- Validates that In-Day Optimization is checked only when Enhanced Optimization is enabled in the org.

After Insert:

- Creates Earliest Start Time and Due Date basic time rules if they don't exist.
- Create the Scheduling Policy Work Rule junction object to assign these work rules to the inserted policy.

Before Update:

- Validates that the Travel Mode checkbox is checked only when this feature is enabled in the org.
- Validates that In-Day Optimization is checked only when Enhanced Optimization is enabled in the org.

Before Delete:

- Validate that the deleted record isn't the default scheduling policy for the appointment booking action.
- Validates that the Travel Mode checkbox is checked only when this feature is enabled in the org.
- Validates that In-Day Optimization is checked only when Enhanced Optimization is enabled in the org.
- Deletes the Earliest Start Time and Due Date'Scheduling Policy Work Rules junction objects.

Scheduling Policy Work Rule

Before Insert:

- Validates that there are no duplications. That is, validates that there's just one Scheduling Policy Work Rule record referencing a specific work rule in the same policy.
- Validates that there aren't more than two enhanced match rules in the same policy. If there are two, make sure they're referencing different Linking Objects.
- Validates that there aren't more than four Count Rules with Count Type of CustomValue.
- Validates that all Count Rule fields are valid.

Before Update:

Same as Before Insert.

Before Delete:

- Prevents the deletion of a Scheduling Policy Work Rule record related to Due Date and Earliest Start Time basic rules.
- If you delete the Scheduling Policy, it passes over this validation and deletes all related Scheduling Policy Work Rules.

Service Appointment

Before Insert:

- Remove seconds from the Service Appointment's DateTime fields.
- If there's a geolocation (latitude and longitude) on the created ServiceAppointment record, limit the decimal places after the decimal point to a maximum of 6 digits.
- If there's a geolocation (latitude and longitude) on the created ServiceAppointment record, populate the FSL__InternalSLRGeolocation__Latitude__s, FSL__InternalSLRGeolocation__Longitude__s fields. These fields are used when calculating street-level routing routes.
- If the Derive the Service Appointment due date from its Work Type setting is enabled, this trigger derives the Duration field from the work order parent record and populates the Due Date field with the value of Earliest Start Permitted plus Due Date offset, which comes from the parent's work type field.
- If the Set your default service appointment duration to one hour setting is enabled, this trigger sets the service appointment duration to 1 hour if one of the following occurs:
  - The duration field is null.
  - The duration is shorter than 1 minute.

After Insert:

- If the Enable User Territories sharing setting is enabled, share the created service appointment with the service territory public group. This behavior provides sharing to the dispatchers who are part of the service appointment's territory.
- If the Enable User Territories sharing setting is enabled and the respective service appointment parent sharing setting is enabled, that is, Share parent Work Order when Service Appointment is shared, this trigger shares the created service appointment's parent with the service territory public group. This behavior provides sharing to the dispatchers who are part of the service appointment's territory.
- If the Derive the Service Appointment due date from its Work Type setting is enabled, this trigger derives the Duration field from the work order parent record and populates the Due Date field with the value of Earliest Start Permitted plus Due Date offset, which comes from the parent's work type field. This behavior is intentionally in both the Before Insert and the After Insert triggers.
- Schedules created service appointments in batches if they have the Auto Schedule field value set to true. The default policy, which is set in the Field Service settings page, is used when auto-scheduling service appointments upon creation.
- After the batch was invoked, the Auto Schedule flag is set to false automatically.
- If the Use polygons to assign service territories setting is enabled, this trigger auto populates the created service appointment's service territory field based on the service appointment's address and its match of a Map Polygon Service Territory field.
- If there's no match between the service appointment's address and the organization map polygons, the service territory field remains unchanged.
- If the Appointment Bundling feature is enabled; the Use Bundle Apex Mode custom setting equals 2; and the inserted service appointment is a bundle or a bundle member; then the afterServiceAppointmentUpdate API http request is triggered.
- If the Appointment Bundling feature is enabled; automatic bundling is enabled; the inserted service appointment isn't a bundle or a bundle member; the modified fields of the service appointment exist within the restrictions of the automatic mode; the service territory of the service appointment is in the defined ApptBundlePolicySvcTerr; the Recordeset Filter Criteria matches; the status of the service appointment exists in the bundle statuses; the service appointment isn't pinned; and the due date isn't in the past while considering the timezone of the service appointment's service territory; then the automatic API request is triggered.

Before Update:

- Remove seconds from the service appointment's DateTime fields.
- If there's a geolocation (latitude and longitude) on the created ServiceAppointment record, this trigger limits the decimal places after the decimal point to a maximum of 6 digits.

- If Service Appointment Status Transitions are enabled in the Field Service settings page and there was a change in the service appointment status field, this trigger validates the status transition legality.
- If the updated service appointment's Status Category was changed to either Canceled or New, this trigger removes the Scheduled Start Time and Scheduled End Time field values.
- Prevents updating the Scheduled Start Time, Scheduled End Time, Latitude, and Longitude field values if the Is Pinned field value is set to true.
- If the service appointment is a multi-day service appointment, this trigger checks if there's a need to recalculate the service duration. If so, it recalculates the length.
- If the Drip-Feed feature is enabled in the settings page, this trigger dispatches the next service appointment in the day. This action is done in case the current service appointment status categories changed to Completed, Canceled or,Couldn't Complete. For more information, see Drip Feed Service Appointments in Salesforce Help.
- If the Mention assigned user when the Service Appointment is dispatched setting is enabled, mention the user upon service cancellation.
- Validates that the updated service appointment duration is longer than 1 minute. if the duration is null or 1 minute, change the duration to 1 hour.
- If there's a geolocation (latitude and longitude) on the updated service record and it has changed within the update, this trigger populates the FSL__InternalSLRGeolocation__Latitude__s, FSL__InternalSLRGeolocation__Longitude__s fields. Those fields are used when calculating street-level routing routes.
- If the FSL__Prevent_Geocoding_For_Chatter_Actions__c field is checked, this trigger disables Data.com's geolocation cleanup, which happens on address change, and returns the value of FSL__Prevent_Geocoding_For_Chatter_Actions__c field to false.
- If the Auto Schedule field on the updated service appointment record is set to true upon the service appointment's update process, this trigger prepares a set of service appointments that are called on the After Update trigger operation and sets the field to false.
- If the Use polygons to assign service territories setting is enabled and their geolocations fall within relevant Map Polygons records, this trigger classifies the updated service appointment territories based on a geographic area. For more information, see Create and Manage Map Polygons and Enable Map Polygons in Salesforce Help.

After Update:

- Shares the updated service appointments with the assigned resources if the status categories were changed to Dispatched. Shares with the service territory public groups if the service territories changed.
- Mentions the assigned resource user if the relevant setting is enabled.
- Performs this operation in an asynchronous method if not already in an async context.
- Creates and deletes Salesforce calendar events based on the service appointment status changes.
  - If the SA Status category changed from New/Scheduled to Dispatched, this trigger creates a calendar event.
  - If the service appointment status category changed from Dispatched to Scheduled/New, this trigger deletes the calendar event.
  - If a service appointment assigned resource was changed and the service appointment remained dispatched, this trigger updates the relevant Salesforce calendar event accordingly.
- Updates the Resource Capacity object according to the service appointment change. If a service appointment was scheduled to a contractor, this trigger updates their capacity to reflect the actual scheduled working time and working items allocated.
- Schedules updated service appointments in batches if they have the Auto Schedule field value set to true. The default policy, which is set in the Field Service settings page, is used when auto-scheduling service appointments upon update.
- After the batch is invoked, the Auto Schedule flag is set to false automatically.
- If the Make assigned resources followers of service appointments that are Dispatched or In Progress setting is enabled in the settings page, and the Status Category field was updated, this trigger makes the assigned resources followers of the updated service appointments.

- If an updated service appointment is assigned to a Service Crew, this trigger creates/updates/deletes the service's assigned resources according to the newly updated service appointment to reflect the actual Service Crew scheduling. To learn more, see Considerations for Scheduling Service Crews in Salesforce Help.
- If the travel trigger is enabled in the org, this trigger recalculates travel for all services scheduled in the same day of the services' start and end dates.
- If the Scheduled Start and End Times of the updated service appointments are null but the service appointments have an assigned resource related to them, this trigger deletes the assigned resources.
- Updates the schedule mode according to the actual scheduling operation, that is, Optimization, Automatic, or Manual.
- Evaluates the scheduling recipes according to the actual service appointment update, that is, Canceled, Shortened, Late-end, or Emergency.
- To learn more, see Create Scheduling Recipes for Common Events in Salesforce Help.
- If the Appointment Bundling feature is enabled; the Use Bundle Apex Mode custom setting equals 2; and the updated service appointment is a bundle or a bundle member; then run the bundle service appointment logic.
- If the Appointment Bundling feature is enabled; the Use Bundle Apex Mode custom setting equals 2; and the updated service appointment is a bundle or a bundle member; then the afterServiceAppointmentUpdate API http request is triggered.
- If the Appointment Bundling feature is enabled; the Use Bundle Apex Mode custom setting equals 2; and the updated service appointment is a bundle member; then the trigger doesn't update the SchedStartTime and the SchedEndTime fields when the RelatedBundleId field hasn't changed. Only if the RelatedBundleId is updated, the SchedStartTime and the SchedEndTime fields change.
- If the Appointment Bundling feature is enabled; automatic bundling is enabled; the updated service appointment isn't a bundle or a bundle member; the modified fields of the service appointment exist within the restrictions of the automatic mode; the service territory of the service appointment is in the defined ApptBundlePolicySvcTerr; the Recordeset Filter Criteria matches; the status of the service appointment exists in the bundle statuses; the service appointment isn't pinned; and the due date isn't in the past while considering the timezone of the service appointment's service territory; then the automatic API request is triggered.

Before Delete:

- Deletes the matching calendar event if it exists. See the Calendar Sync's Knowledge Article for more information.
- Deletes assigned resource records from the deleted service appointments.
- Remove sharing on the parent record, that is, the Work Order, from the territories' public groups.

Service Objective

Before Insert:

- Validates that the service objective has a record type.
- If the service objective is a custom logic objective, validates and decodes the Custom Logic Data (must be filled).

Before Update:

Same as Before Insert.

Service Resource

Before Insert:

- If the Efficiency field isn't empty, this trigger checks if its value is in the valid range (between 0.1–10). If not, this trigger prevents the insertion and shows an error next to the Efficiency field.
- If the ServiceCrewId field is empty, this trigger checks if there's another Service Resource record pointing to the same service crew. If so, the trigger throws an error and prevents the insertion.

After Insert:

- If location-based sharing is turned on in your org, this trigger creates a ServiceResourceShare object for the user specified in the RelatedRecordId field. If location-based sharing is turned off it, this trigger does nothing. If the RelatedRecordId field is empty for a specific record, for example, a Service Resource of type Crew, this trigger doesn't create a share object for the specific record.

Before Update:

- If the IsCapacityBased checkbox changed from false to true, this trigger validates there's only one service territory member, excluding secondary service territory members, linked to that resource. Otherwise, this trigger throws an error and prevents the update.
- If the Efficiency field isn't empty, this trigger checks if its value is in the valid range (between 0.1–10). If not, this trigger prevents the update and shows an error next to the Efficiency field.
- If the IsCapacityBased checkbox is checked, this trigger validates that this service resource doesn't have any Service Crew Member records. Otherwise, this trigger throws an error and prevents the update.
- If the ServiceCrewId field changed and the new value isn't empty, this trigger validates that there's no existing Service Resource record pointing to that Service Crew. Otherwise, this trigger throws an error and prevents the update.

After Update

- If the RelatedRecordId changed in the update and location-based sharing is turned on in your org, this trigger deletes all manual sharing records for the Service Resource, including those records created by the user, and creates a ServiceResourceShare object for the new related user. If it fails to delete one of the manual sharing objects this trigger doesn't delete any of them.

Service Resource Capacity

The trigger is named TR012_CapacityAfterUpdate but it fires only before insert and before update.

Before Insert:

- Validates that monthly capacities are defined in the first day of the month.
- Validates that capacities of the same duration type don't overlap each other, that is, they don't share a resource and date.
- Validates that the Time Period and End Date fields are valid.
- Updates the MinutesUsed__c field according to the duration of services scheduled within the capacity.
- If the CapacityInWorkItems field isn't empty, then this trigger updates the Work_Items_Allocated__c field with the number of services scheduled within the capacity.
- HoursInUse__c field is a formula field based on MinutesUsed__c. This field updates accordingly.
- Updates the Last Updated Epoch field with the number of milliseconds since January 1, 1970, 00:00:00 GMT.

Before Update:

- Validates that monthly capacities are defined in the first day of the month.
- Validates that capacities of the same duration type don't overlap, that is, they don't share a resource and date.
- Validates that the the Time Period and End Date fields are valid.
- Validates if one or more of the following fields have changed: TimePeriod, StartDate, CapacityInHours, CapacityInWorkItems, ServiceResourceId.
- If any of these fields have changed, this trigger updates the MinutesUsed__c field according to the duration of services scheduled within the capacity.
- If the CapacityInWorkItems field isn't empty, then this trigger updates the Work_Items_Allocated__c field with the number of services scheduled within the capacity.
- HoursInUse__c field is a formula field based on MinutesUsed__c. This field updates accordingly.
- Updates the Last Updated Epoch field with the number of milliseconds since January 1, 1970, 00:00:00 GMT.

Service Resource Skill

Before Insert:

- Removes seconds and milliseconds from the Effective Start Date and Effective End Date fields.

Before Update:

Same as Before Insert.

Service Territory

Before Insert:

- If the Enable Territory Name Duplicates custom setting is off (the default value is off), then this trigger makes sure there's no duplication in the names of the inserted territories.
- If the territory has a geolocation, then this trigger fills up the internal street-level routing geolocation values.

After Insert:

- If the Enable User Territories sharing setting is enabled, this trigger creates a new public group with the name of the territory, if no such group exists, and creates a sharing record for that group.
- Adds the public group of the territory as a group member to the public group of the parent territory, if it has one.
- If the Enable Service Auto Classification custom setting is on (the default is on), this trigger sets the FSL__TerritoryLevel__c field according to the parent territory and top-level territory values of all territories. This field represents the level of the territory in the hierarchy and is used in the polygon's algorithm.
- If the Appointment Bundling feature is enabled; and the Use Bundle Apex Mode custom setting equals 2; then the serviceTerritoryRefresh API http request is triggered.

Before Update:

- If the Enable Territory Name Duplicates custom setting is off (the default value is off) and the Name field has changed, then this trigger makes sure there's no duplication in the names of the updated territories.
- If the territory has a geolocation, then this trigger fills up the internal street-level routing geolocation values.

After Update:

- If the name or owner fields changed, this trigger updates the public group of the territory accordingly.
- If the Parent Territory field changed, this trigger updates the Parent Territory's public group and recalculates the FSL__TerritoryLevel__c field value to all records according to the new hierarchy.
- If the Appointment Bundling feature is enabled; the Use Bundle Apex Mode custom setting equals 2; and the Operating Hours property was updated; then the serviceTerritoryRefresh API http request is triggered.

Before Delete:

- Deletes the Service Territory's public group.
- Updates children territory's FSL__TerritoryLevel__c field according to the new hierarchy.
- If the Appointment Bundling feature is enabled; and the Use Bundle Apex Mode custom setting equals 2; then the serviceTerritoryRefresh API http request is triggered.

Service Territory Member

Before Insert:

- Removes seconds from the service territory member DateTime fields.

- Checks for date collisions. If a primary or relocation service territory member exists in the same date, the trigger fails.
- If a secondary service territory member exists in the same date and the created service territory member is in the same service territory, the trigger fails.
- If there's a geolocation (latitude and longitude) on the created service territory member record, this trigger populates the FSL__InternalSLRGeolocation__Latitude__s, FSL__InternalSLRGeolocation__Longitude__s fields. Those fields are used when calculating street-level routing routes.
- Prevents creating service territory members of type Relocation to contractors.
- Prevents creating more than one service territory member, excluding Secondary service territory members, to contractors.

Before Update:

- Checks for date collisions. If a primary or relocation service territory member exists in the same date, the trigger fails.
- If a secondary service territory member exists in the same date and the created service territory member is in the same service territory, the trigger fails.
- If there's a geolocation (latitude and longitude) on the updated service territory member record, this trigger populates the FSL__InternalSLRGeolocation__Latitude__s, FSL__InternalSLRGeolocation__Longitude__s fields. Those fields are used when calculating street-level routing routes.
- Prevents creating multiple service territory members of type Relocation or Primary to contractors.
- Prevents creating more than one service territory member (excluding Secondary service territory members) to contractors.

Skill Requirement

All triggers are empty.

Time Dependency

After Insert:

- Prevents creating duplicated dependencies between service appointments.
- Prevents adding an Immediately Follow dependency type to an already created time dependency chain.
- Adds service appointments to an existing chain according to the created Time Dependency. If needed, this trigger merges two chains according to the created Time Dependency.

Before Update:

- Prevents updating a Time Dependency if an existing Time Dependency has the same two service appointments.
- Prevents adding an Immediately Follow dependency type to an already created time dependency chain.

After Delete:

- Splits two dependencies according to the deleted Time Dependency. For example, if chain A contains SerA SerB SerC SerD SerE and the deleted Time Dependency is SerC SerD, this trigger splits the chain into two: SerA SerB SerC and SerD SerE.

Time Slot

Before Insert:

- Removes seconds from DateTime fields.

Before Update:

- Removes seconds from DateTime fields.

User Territory

Before Insert:

- If the Enable User Territories sharing setting is enabled, this trigger checks for uniqueness within all user territories. In other words, this trigger ensures that the same user isn't related to the same territory.

After Insert:

- If the Enable User Territories sharing setting is enabled, this trigger adds users to the respective public groups according to the created User Territory. Make the User Territory.Service Territory name the same as the public group name.

BeforeUpdate:

- If the Enable User Territories sharing setting is enabled, this trigger checks for uniqueness within all user territories. In other words, this trigger ensures that the same user isn't related to the same territory.

After Update:

- If the Enable User Territories sharing setting is enabled, this trigger adds and removes users to or from the respective public groups according to the updated user territory. For example, if a user territory is updated from User: David, Service Territory: LA to User: David, Service Territory: Washington, this change results in removing David from the LA public group and adding him to the Washington public group.

After Delete:

- If the Enable User Territories sharing setting is enabled, this trigger removes users from the respective public groups according to the deleted user territory.

Work Order

Before Insert:

- If the AccountId field isn't empty and the VisitingHoursId field is empty, this trigger populates VisitingHoursId with the related account's operating hours ID.

Before Update:

- If the AccountId or VisitingHoursId fields changed, the AccountId field isn't empty, and the VisitingHoursId field is empty, this trigger populates VisitingHoursId with the related account's operating hours ID.
- If the FSL__Prevent_Geocoding_For_Chatter_Actions__c field is checked, this trigger disables Data.com's geolocation cleanup, which happens on an address change, and returns the value of FSL__Prevent_Geocoding_For_Chatter_Actions__c field to false.

Work Order Line Item

All triggers are empty.

Work Rule

Before Update:

- If the work rule is Availability Rule, this trigger prevents setting a minimum gap if the fixed gap checkbox is disabled.
- If the work rule is Availability Rule, this trigger prevents setting a break time if the Break And Travel trigger is disabled.
- If the work rule is Match Boolean Rule, this trigger prevents updating the work rule if the resource property field isn't populated.
- If the work rule is a Time Rule, this trigger prevents updating a time rule with Scheduled Start Time Equal To/Before Arrival Window Start/End.

- If the work rule is an Enhanced Match Rule or a Count Rule, this trigger prevents updating the rules with invalid field values.
- If basic time work rules don't exist in the org or are changed, this trigger creates valid basic time rules (Early Start Permitted & Due Date work rules).

Before Delete:

- Prevent deletion of the basic time work rules (Early Start Permitted & Due Date).

Field Service Code Examples

Use these code examples to get started working programmatically with Field Service features.

IN THIS SECTION:

Create a Service Report with Apex

The Create Service Report action on work orders, work order line items, and service appointments can also be called using Apex code. The code example shown creates a service report with two signatures by making an Apex callout to the `createServiceReport` action REST API resource.

Generate Work Orders on Maintenance Plans with Apex

The **Generate Work Orders** action on maintenance plans can also be called using Apex code. The following code sample creates work order records by making an Apex callout to the *`generateWorkOrder`* action REST API resource.

Code Examples: Dispatcher Console Custom Actions

Learn how to configure Apex classes or Visualforce pages that you want to link to a custom action in the dispatcher console.

Create Service Appointment Lists in the Dispatcher Console

You can create user-specific temporary appointment lists by adding the `Create Temporary Service Appointment List` custom permission in Field Service managed package. If the custom permission is enabled, the Field Service app creates a connection to the `CreateFilterEvent__e` platform event channel and subscribes to related messages.

Create a Service Report with Apex

The Create Service Report action on work orders, work order line items, and service appointments can also be called using Apex code. The code example shown creates a service report with two signatures by making an Apex callout to the `createServiceReport` action REST API resource.

Each service report contains data from its parent record and signatures from customers, technicians, or others who must sign off on the work. In your API call, you can define one or more signatures to be used in the report. The number and type of signatures must match the service report template's signature settings.

Note:

- This code example applies to API version 41.0 and later. Service reports generated with API version 40.0 can contain only one signature.
- This code example assumes that you have remote site settings enabled to make REST callouts. For more information, see Adding Remote Site Settings.
- The mock record ID in the code example is *`0WOxx000000001E`*. The mock service report template ID is *`0SLR00000004DBFOA2`*.

API Endpoint (v41.0): `/services/data/v41.0/actions/standard/createServiceReport`

```
String salesforceHost = System.Url.getSalesforceBaseURL().toExternalForm();
String url = salesforceHost + '/services/data/v41.0/actions/standard/createServiceReport';
// Create HTTP request
HttpRequest request = new HttpRequest();
request.setEndpoint(url);
request.setMethod('POST');
request.setHeader('Content-Type', 'application/json');
request.setHeader('Authorization', 'OAuth ' + UserInfo.getSessionId());
// Set the body as a JSON object
request.setBody('{"inputs" : [ {"entityId" : "0WOxx000000001E","signatures" :
[{"data":"BASE64_CODE_FOR_THE_CAPTURED_SIGNATURE_IMAGE","contentType":"image/png","name":"Customer
 Signature","signatureType":"Default","place":"San Francisco","signedBy":"John
Doe","signedDate":"2019-06-05 12:00:00"}],"templateId" : "0SLR00000004DBFOA2"} ]}');
Http http = new Http();
HttpResponse response = http.send(request);
// Parse the JSON response
if (response.getStatusCode() != 201) {
    System.debug('The status code returned was not expected: ' +
                response.getStatusCode() + ' ' + response.getStatus());
} else {
    System.debug(response.getBody());
}
```

Where `BASE64_CODE_FOR_THE_CAPTURED_SIGNATURE_IMAGE` is a variable.

This code example creates a payload with two signatures.

```
{
   "inputs":[
      {
         "entityId":"0WOxx000000001E",
         "signatures":[
            {
               "data":"Base64 code for the captured signature image",
               "contentType":"image/png",
               "name":"Customer Signature",
               "signatureType":"Customer",
               "place":"San Francisco",
               "signedBy":"John Doe",
               "signedDate":"Thu Jul 13 22:34:43 GMT 2017"
            },
            {
               "data":"Base64 code for the captured signature image",
               "contentType":"image/png",
               "name":"Technician Signature",
               "signatureType":"Technician"
            }
         ],
         "templateId":"0SLR00000004DBFOA2"
      }
   ]
}
```

Field Descriptions

- entityId: The ID of the work order, work order line item, or service report that you want to create the service report for.
- signatures: A list of JSON definitions for a digital signature.
  - data: (Required) The base64 code for an image.
  - contentType: (Required)The file type of the signature.
  - signatureType: (Required) The role of the person signing such as Customer. Signature Type picklist values are defined by the Salesforce admin ahead of time. Each signature block must use a different signature type, and the signature types that you define in your call must match the service report template's signature types.
  - name: The signature block title. This value appears on the generated service report.
  - place: The place of signing. This value appears on the generated service report.
  - signedBy: The name of the person signing. This value appears on the generated service report.
  - signedDate: The date of signing. This value appears on the generated service report.
- templateId: The ID of the service report template used for the report. To find the ID, run a SOQL query on the ServiceReportLayout object.

Limitations

If you're a guest user and are using `Future` annotation, the ContentDocument object can't be created and you can't create the Service Document.

SEE ALSO:

Salesforce Actions Developer Guide: Create Service Report Actions

Generate Work Orders on Maintenance Plans with Apex

The **Generate Work Orders** action on maintenance plans can also be called using Apex code. The following code sample creates work order records by making an Apex callout to the `generateWorkOrder` action REST API resource.

You can use this code sample in several different ways:

- Add it to Apex controller code for an Aura component, and tie it to a custom UI or app functionality
- Use it in an Apex trigger to semi-automate the creation of the work orders (for example, whenever a maintenance plan is created or updated)
- Use it in an Apex REST service to create work orders when called from an external integration service

Note:

- This code sample assumes that you have remote site settings enabled to make REST callouts. For more information, see Adding Remote Site Settings.
- The mock maintenance plan ID in the code sample is `1MPR000000000Bu`.
- This code sample references API version 45.0, but can be used with any version from 40.0 onward.

API Endpoint (v45.0): `/services/data/v45.0/actions/standard/generateWorkOrders`

```
String salesforceHost = System.Url.getSalesforceBaseURL().toExternalForm();
String url = salesforceHost + '/services/data/v45.0/actions/standard/generateWorkOrders';
// Create HTTP request
```

```
HttpRequest request = new HttpRequest();
request.setEndpoint(url);
request.setMethod('POST');
request.setHeader('Content-Type', 'application/json;charset=UTF-8');
request.setHeader('Authorization', 'OAuth ' + UserInfo.getSessionId());
// Set the body as a JSON object
request.setBody('{"inputs" : [{"recordId" : "1MPR000000000Bu"}]}');
Http http = new Http();
HttpResponse response = http.send(request);
// Parse the JSON response
if (response.getStatusCode() != 201) {
    System.debug('The status code returned was not expected: ' +
    response.getStatusCode() + ' ' + response.getStatus());
} else {
    System.debug(response.getBody());
}
```

SEE ALSO:

Salesforce Actions Developer Guide: Generate Work Orders Actions

Code Examples: Dispatcher Console Custom Actions

Learn how to configure Apex classes or Visualforce pages that you want to link to a custom action in the dispatcher console.

Custom actions can either call an Apex class or open a Visualforce page, and can be run on records in several areas of the dispatcher console. To learn how to create custom actions, see Create Custom Actions for the Dispatcher Console.

Note: All Apex classes implementing the quick action interfaces must be declared as Global to be accessible from the dispatcher console.

Creating Apex Classes

When you create an Apex class to link to a dispatcher console custom action, implement one of the following three interfaces in your class.

Interface	Description
`CustomGanttServiceResourceAction`	For actions on service resources. The parameters are the service resource record ID, the service territory member record ID reflected on the Gantt, and the start and end dates of the current Gantt view. No additional parameters are included. Use the following format: `String action(Id resourceId, Id stmId, Datetime strGanttStartDate, Datetime strGanttEndDate, Map<String, Object> additionalParameters)`
`CustomGanttServiceAppointmentAction`	For actions on service appointments. The parameters are the service appointment record IDs—used for bulk actions—and the start and end dates of the current Gantt view. No additional parameters

Interface	Description
	are included. The `Map<String, Object> additionalParameters` input parameter is reserved for future use but must be included to run the code. Use the following format: `String action(List<Id> serviceAppointmentsIds, Datetime strGanttStartDate, Datetime strGanttEndDate, Map<String, Object> additionalParameters)` When this action is implemented, multiple service appointments can be returned. In your method, we recommend creating an `if` statement to check how many IDs are returned. First, validate that at least one ID was returned: `serviceAppointmentsIds.size()>1`. Then, you can take different actions depending on whether 0, 1, or more appointment IDs were returned.
`CustomGanttResourceAbsenceAction`	For actions on resource absences. The parameters are the resource absence record ID, the absence type ('na' or 'break'), and the start and end dates of the current Gantt view. No additional parameters are included. The `Map<String, Object> additionalParameters` input parameter is reserved for future use but must be included to run the code. Use the following format: `String action(Id absenceId, String absenceType, Datetime strGanttStartDate, Datetime strGanttEndDate, Map<String, Object> additionalParameters)`

These functions must be global and require that a string be returned. If the string isn't empty, it is used in the Gantt notification shown when a user clicks the related action.

Note: Use the Datetime class for all date and time calculations.

Code Example: Service Resource Custom Action

This action creates a resource absence of type Non Availability for the selected service resource that spans the days visible on the Gantt.

```
global class BlockResourceVisibleTime implements FSL.CustomGanttServiceResourceAction {

    global String action(Id resourceId, Id stmId, Datetime ganttStartDate, Datetime
ganttEndDate, Map<String, Object> additionalParameters) {

        ResourceAbsence na = new ResourceAbsence();

        // get Resource Absence record type - NA
        RecordType recordTypeNA = [
            SELECT
                Id, SobjectType, Name
```

```
            FROM
                RecordType
            WHERE
                DeveloperName =: 'Non_Availability'
                AND
                SObjectType =: ResourceAbsence.getSobjectType().getDescribe().getName()
            ];

        na.RecordTypeId = recordTypeNA.Id;
        na.ResourceId = resourceId;
        na.FSL__Approved__c = true;
        na.Start = ganttStartDate;
        na.End = ganttEndDate;

        insert na;

       ServiceResource resource = [SELECT Name FROM ServiceResource WHERE Id =: resourceId];


        return 'Blocked availability to ' + resource.Name + ' from ' +
ganttStartDate.format() + ' to ' + ganttEndDate.format();

    }

}
```

Code Example: Service Appointment Custom Action

This action toggles the In Jeopardy field between True and False.

```
global class toggleServiceAppointmentJeopardy implements
FSL.CustomGanttServiceAppointmentAction {

   global String action(List<Id> serviceAppointmentsIds, Datetime ganttStartDate, Datetime
 ganttEndDate, Map<String, Object> additionalParameters) {

        List<ServiceAppointment> saList = [SELECT FSL__InJeopardy__c, AppointmentNumber
FROM ServiceAppointment WHERE Id in : serviceAppointmentsIds];
        String reply = '';
        List<String> saNames = new List<String>();

        for (ServiceAppointment s : saList) {
            s.FSL__InJeopardy__c = !s.FSL__InJeopardy__c;
            saNames.add(s.AppointmentNumber);
        }

        upsert saList;

        reply = String.join(saNames, ', ');
        return 'Service Appointments successfully processed: ' + reply;
    }

}
```

Code Example: Resource Absence Custom Action

For resource absences of type NA, this action creates a duplicate absence on the following day.

```
global class copyAbsenceToNextDay implements FSL.CustomGanttResourceAbsenceAction {

   global String action(Id absenceId, String absenceType, Datetime ganttStartDate, Datetime
 ganttEndDate, Map<String, Object> additionalParameters) {

      ResourceAbsence resourceAbsence = [SELECT Id, AbsenceNumber, Start, End, ResourceId,
 RecordTypeId, FSL__Approved__c FROM ResourceAbsence WHERE Id =: absenceId LIMIT 1];

       ResourceAbsence raClone = resourceAbsence.clone(false, true, false, false);
       raClone.Start = resourceAbsence.Start.addDays(1);
       raClone.End = resourceAbsence.End.addDays(1);
       raClone.ResourceId = resourceAbsence.ResourceId;
       raClone.RecordTypeId = resourceAbsence.RecordTypeId;
       raClone.FSL__Approved__c = true;
       insert raClone;

       return 'Resource Absence successfully copied.';
    }

}
```

Creating Visualforce Pages

When you create a Visualforce page, use the following GET parameters.

For actions on...	Description
Service appointments	Use the following format: `services [if multiple], id (comma delimited if multiple), start (current Gantt start date, string), end (current Gantt end date, string)` Example for a Visualforce page used to update a single service appointment: `?id=08p4E000000Kj5hQAC&start=5-7-2018&end=5-8-2018` Example for a Visualforce page used to update multiple service appointments: `?services=08p4E000000Kj5hQAC,08p4E000430Kj5hAPP&start=5-7-2018&end=5-8-2018`
Service resources	Use the following format: `id, stm (ID of service resource's current service territory member record), start (current Gantt start date, string), end (current Gantt end date, string)` Example: `?id=0Hn4E0000001OMQSA2&stm=0Hu4E0000005cpPSAQ&start=5-7-2018&end=5-8-2018`

For actions on...	Description
Resource absences	Use the following format: `id, type ('break' or 'na'), start (current Gantt start date, string), end (current Gantt end date, string)` Example: `?id=0Hw4E00000091HSSAY&type=break&start=5-7-2018&end=5-8-2018`

To close the Visualforce lightbox from your code, use: `parent.postMessage('closeLightbox','*');`

Note: Custom dispatcher console actions can't open Visualforce pages that are part of a managed package.

SEE ALSO:

Create Custom Actions for the Dispatcher Console

Create Service Appointment Lists in the Dispatcher Console

You can create user-specific temporary appointment lists by adding the `Create Temporary Service Appointment List` custom permission in Field Service managed package. If the custom permission is enabled, the Field Service app creates a connection to the `CreateFilterEvent__e` platform event channel and subscribes to related messages.

For information on allocation limits consumed by subscribing to the event channel, see Change Data Capture Allocations.

The custom filters are created and accessible only by users who created the `CreateFilterEvent__e` platform event, and not for other users. Based on a custom filter, you can create temporary lists with identical names. Each list is created as a new instance and doesn't override an existing one. Each list can include a maximum of 300 service appointments.

After a temporary appointment list is loaded to the dispatch console, the list remains static. To auto-refresh the list, enable **Enhanced Live Updates** in the Field Service Admin app and make sure you have read access to all fields in the service appointment's field sets. When the dispatch console is refreshed, the temporary list is no longer available. To delete a temporary list without refreshing, click **Delete Filter**.

To create a custom filter for appointments, define and publish a new `CreateFilterEvent__e` platform event by adding these custom fields.

- `FSL__ServiceApptIds__c`: JSON array of service appointment Ids. For example, *`["08pO10000049k7lIAA","08pO10000049k7mIAA"]`*
- `FSL__FilterName__c`: Name of the temporary list based on the specified service appointment IDs. For example, *`High Priority Appointments`*
- `FSL__Description__c` (optional): Description. For example, *`This filter displays high priority appointments`*.
- `FSL__FilterCategory__c`: Value must be set to *`GENERAL`*
- `FSL__LoadFilterImmediately__c`: When set to `true`, the dispatcher console switches to the new temporary list.

This code sample creates a `FSL__CreateFilterEvent__e` platform event and loads a temporary list created from the queried list of service appointment IDs to the dispatcher console.

```
List<id> ids = new List<id>();
for (ServiceAppointment s :[select id from ServiceAppointment limit 10])
```

```
                        { ids.add(s.id);
                        }

FSL__CreateFilterEvent__e filterEvent = new FSL__CreateFilterEvent__e();

filterEvent.FSL__ServiceApptIds__c = JSON.serialize(ids);
system.debug(filterEvent.FSL__ServiceApptIds__c);
filterEvent.FSL__FilterName__c = 'My Temporary List';
filterEvent.FSL__Description__c = 'a temporary list that was created to demonstrate this
feature';
filterEvent.FSL__FilterCategory__c = 'GENERAL';
filterEvent.FSL__LoadFilterImmediately__c = true;

Database.SaveResult eventRunningResult = EventBus.publish(filterEvent);
```

SEE ALSO:

Salesforce Field Service: Set Custom Permissions for Field Service

Salesforce Field Service: Create Custom Actions for the Field Service Dispatcher Console

Field Service Mobile App

Configure mobile-specific features, such as deep linking, for the Field Service app.

IN THIS SECTION:

Get Started with Lightning Web Components in the Field Service Mobile App

Learn how to create and use Lightning web components (LWCs) in the Field Service mobile app. LWCs offer a versatile and efficient framework to meet the needs of your Field Service operations, from enhancing user interfaces to extending functionality.

Build Custom Lightning Web Components for Service Documents

Customize templates for your specific business and branding needs by creating custom components with Lightning web components (LWCs).

Configure Deep Linking for the Field Service Mobile App

Your field technicians rely on their mobile devices to perform complex tasks, and some of those tasks require them to navigate between multiple tabs, flows, or actions. With deep linking, you can send users directly to the right spot in the app.

Add Lighting Web Components for Plug-Ins to the Field Service Mobile App

Enhance the Field Service mobile app by creating Lightning web components (LWCs) for plug-ins.

Get Started with Lightning Web Components in the Field Service Mobile App

Learn how to create and use Lightning web components (LWCs) in the Field Service mobile app. LWCs offer a versatile and efficient framework to meet the needs of your Field Service operations, from enhancing user interfaces to extending functionality.

This guide is a companion document to the Mobile and Offline Developer guide. Together these guides help you create seamless and intuitive experiences for your field service mobile workers. Familiarize yourself with the content in the Mobile and Offline Developer Guide first, then return here to learn about and get set up for working with the Field Service mobile app.

IN THIS SECTION:

Considerations for LWC Offline in the Field Service Mobile App
The following considerations apply to LWC Offline when you run your components in the Field Service mobile app.

Field Service Org Setup
Define and assign the required permission sets.

iOS Simulator Setup
During development it's convenient to test your code in a virtual device. Use Xcode to create a device simulator, and install the virtual device build of your mobile app into it.

Android Emulator Setup
During development it's convenient to test your code in a virtual device. Use Android Studio to create a device emulator, and install the virtual device build of your mobile app into it.

Debug Lightning Web Components in the Field Service Mobile App
The best way to develop and debug your Lightning web components is the same way you develop and debug *anything* built with HTML, CSS, and JavaScript: with the debugging tools built into your web browser.

What Can I Do with Lightning Web Components in the Field Service Field Service Mobile App?
You can create custom components that are accessed from the Actions menu. Actions built with LWCs can be simple or complex, depending on the problem you're solving, and can work even when mobile devices are offline. Use them to add and annotate an image to attach to a service report, order a part, contact a SME, or create a custom dashboard for your mobile workers. Use LWCs in Field Service to do pretty much anything Salesforce can do.

Configure Offline Mode in the Field Service Mobile App
Learn what to expect when Automatic Offline Mode is turned on in the Field Service mobile app and how you can adjust its settings.

Design with Lightning Web Components to Look Like Salesforce
Elevate your field service operations with visually cohesive mobile experiences. The Field Service mobile app seamlessly integrates the following Lightning web components with the Salesforce UI style.

Considerations for LWC Offline in the Field Service Mobile App

The following considerations apply to LWC Offline when you run your components in the Field Service mobile app.

Note: Developing with Lightning web components is likely best suited for seasoned developers, although very advanced admins with coding experience can also be successful.

- Global quick actions are available on all pages that have the Actions menu. They do **not** receive the record ID of the current record when invoked from a record detail page.
- A Community license user who opens a service appointment can experience a missing record error. This is a known issue with this specific user type.
- Task objects added to a briefcase aren't primed for offline use. If a briefcase contains a Task object and priming is attempted, an error message is displayed. However, other objects in the briefcase are primed. To resolve the error, remove Task objects from the briefcase.
- There's a conflict between Appointment Assistant and LWC Offline that can result in missing URLs. See this known issue for details.
- (iOS only) Changes to Lightning web components are loaded into the app only when the app is fully quit and then relaunched (a "cold start").

Field Service Org Setup

Define and assign the required permission sets.

Note: This step is required for Field Service orgs. If you're using LWC Offline in the Salesforce mobile app, skip to the next step.

Lightning web components for LWC Offline-enabled mobile apps is an opt-in feature. To enable it for your org, you must define and assign the permissions in the steps provided below. These are required to access and run LWCs within the Field Service mobile app. (While not technically required to develop LWCs, you'll have a hard time developing components when you can't run them.)

.

IN THIS SECTION:

Define a Permission Set for Your Org
Create a permission set that applies the permissions required to enable Lightning Data Service for Field Service mobile app users.

Assign the Permission Set to a Mobile User
Assign Lightning Data Service permissions to Field Service mobile app users.

Define a Permission Set for Your Org

Create a permission set that applies the permissions required to enable Lightning Data Service for Field Service mobile app users.

1. From Setup, enter `Permission Sets` in the Quick Find Box and select **Permission Sets**.
2. Click **New**.
   - For **Label**, enter `Field Service - Lightning Data Service`.
   - For **Description**, enter `Assign to Field Service Mobile users to give them permission to use features that use Lightning Data Service`.
   - For **License**, select **Field Service Mobile**.
3. Click **Save**.
4. In the Find Settings box, enter `Lightning SDK for Field Service Mobile` and click it.
5. Click **Edit**.
6. Select **Lightning SDK for Field Service Mobile** and save your work.

Assign the Permission Set to a Mobile User

Assign Lightning Data Service permissions to Field Service mobile app users.

Create your permission set.

1. From Setup, in the Quick Find box, enter and select **Permission Sets**.
2. Scroll to the permission set you created, in this case **Field Service - Lightning Data Service**, and click on it.
3. Click **Manage Assignments**, then **Add Assignments**.
4. Select the users you'd like to assign this permission set to.

   Note: You can assign an expiration date at which time the permission set is removed for the selected users. If you want to assign the permission set indefinitely, leave **No expiration date** selected.
5. Click **Assign**.
6. On the Assignment Summary screen, click **Done**.

iOS Simulator Setup

During development it's convenient to test your code in a virtual device. Use Xcode to create a device simulator, and install the virtual device build of your mobile app into it.

IN THIS SECTION:

Configure Minimum Required iOS Simulator Settings
Ensure your virtual device meets the minimum device and iOS version requirements to run your mobile app.

Install the Field Service App for iOS
To run the Field Service mobile app in an iOS simulator, download and install a virtual device build of the app. After it's installed, open the app and log into your development org.

Configure Minimum Required iOS Simulator Settings

Ensure your virtual device meets the minimum device and iOS version requirements to run your mobile app.

Review the minimum requirements for your mobile app.

- Field Service Mobile App Requirements
- Salesforce Mobile App Requirements

1. Download and install the latest version of Xcode. If you already have Xcode installed, there's no need to reinstall it.
2. Launch Xcode.
3. In the menu bar, select **Xcode** > **Open Developer Tool** > **Simulator**.
   A new program called Simulator opens, displaying a mobile screen.
4. Go to **File** > **Open Simulator** to choose your preferred device.
   If you want to create a simulator, go to **File** > **New Simulator** and follow the prompts.

After the device simulator launches you can close Xcode, but keep the Simulator app's window open to install the Field Service app in the next section.

Install the Field Service App for iOS

To run the Field Service mobile app in an iOS simulator, download and install a virtual device build of the app. After it's installed, open the app and log into your development org.

1. Download the iOS virtual device build of the Salesforce Field Service mobile app, in the form of a zip file.
   A link to download the latest version of the app is available in this Trailblazer Community post. If you encounter an insufficient permissions page, log in to Trailhead and try the link again.
2. Double-click the downloaded zip file to extract the app file.
3. Drag the downloaded `.app` file into the Simulator window.
4. In Simulator, click (a simulated tap in this case) the newly installed Field Service app to open it.
   If you don't see the app on the first page, it's probably installed on a different app page. You can swipe the page with your mouse to see additional pages to find the app.
5. Click (simulator tap, you get the idea) the Field Service app. Click **Get Started**.
6. Click the screen to run through the tutorial or click **Skip**.
7. Click **Log In**.

8. Click **I Agree** to agree to the Order Form Supplement agreement.
9. Click and select a connection.

   Warning: If you're logging in as a community user for the first time, click to add a new connection.

   - For **Host**, enter your org's URL in the following format: `https://[yourURL].my.salesforce.com`.
   - For **Label**, enter a nickname for your connection.
10. Click **Done**.
11. Enter your username and password for your org.
12. Click **Log In**.
13. Click **Allow** to allow the app to access your Salesforce information.
14. Click through the various permissions screens and allow them the appropriate access.
    When finished, you arrive at the app's home screen.

Android Emulator Setup

During development it's convenient to test your code in a virtual device. Use Android Studio to create a device emulator, and install the virtual device build of your mobile app into it.

IN THIS SECTION:

Configure Minimum Required Android Emulator Settings
Ensure your virtual device meets the minimum device and Android API version requirements to run your mobile app.

Install the Field Service App for Android
To run the Field Service mobile app in an Android emulator, download and install a virtual device build of the app. After it's installed, open the app and log into your development org.

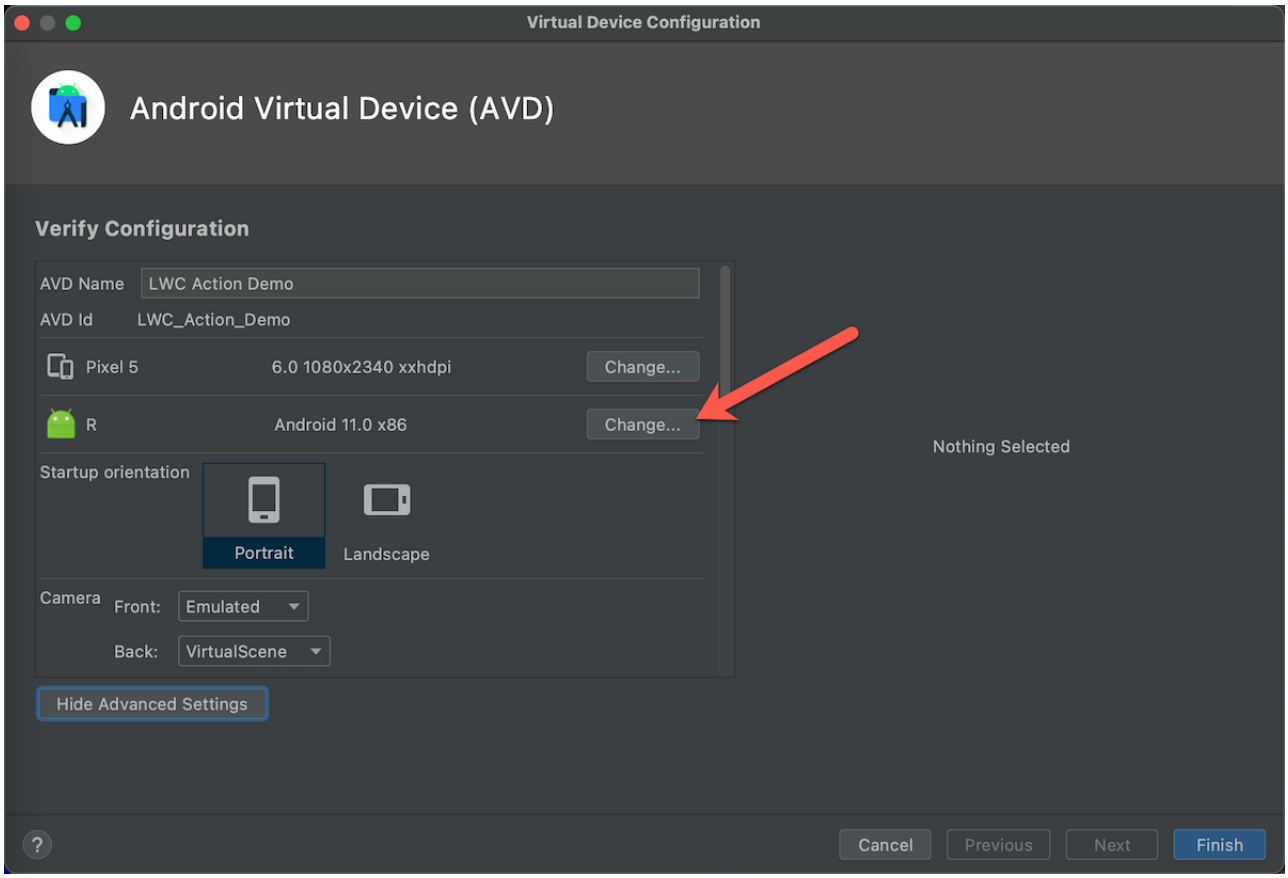
Configure Minimum Required Android Emulator Settings

Ensure your virtual device meets the minimum device and Android API version requirements to run your mobile app.

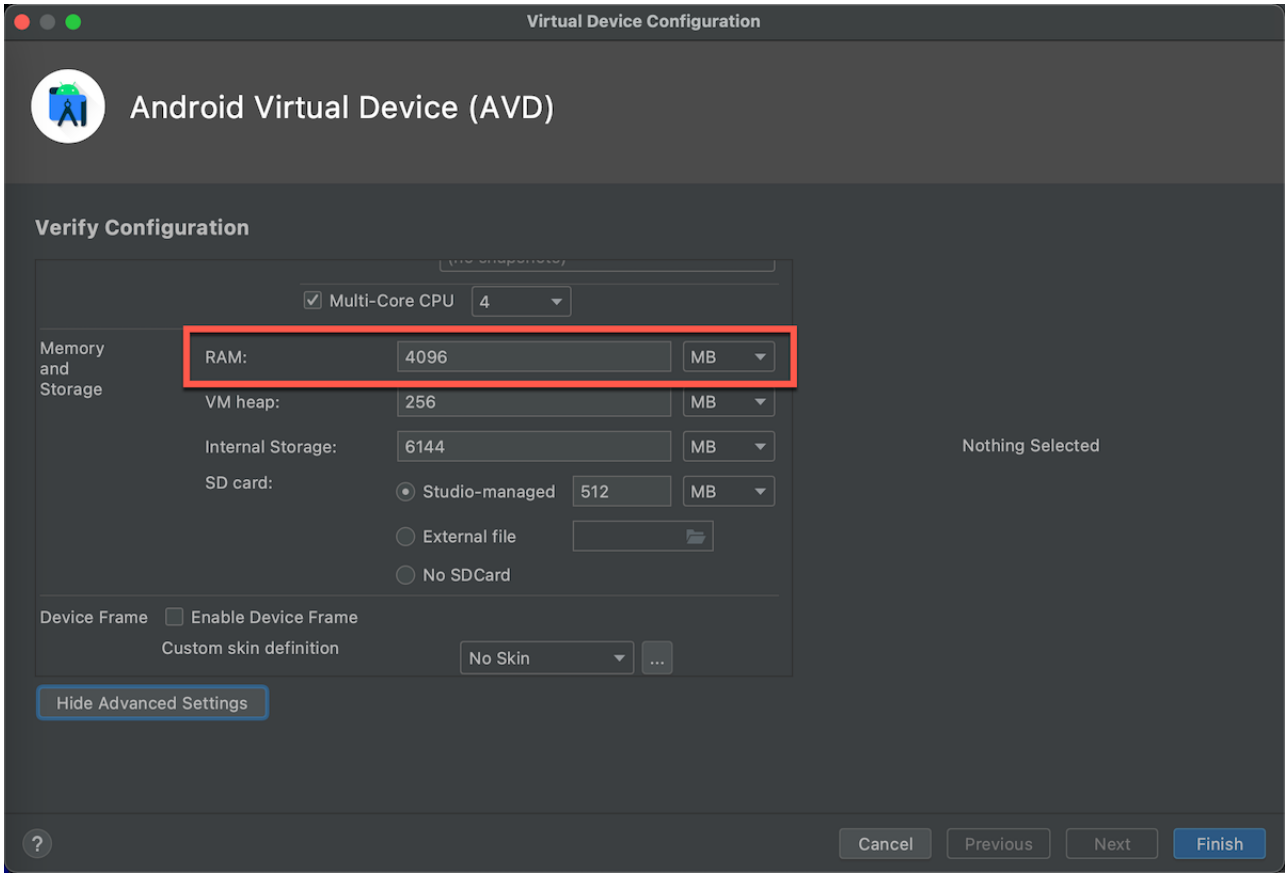
Review the minimum requirements for your mobile app.

- Field Service Mobile App Requirements
- Salesforce Mobile App Requirements

1. Download and install the latest version of Android Studio.
   If you already have Android Studio installed, there's no need to reinstall it.
2. Launch Android Studio.
3. Click **More Actions** or in the top left, depending on your version of Android Studio, and then select **Virtual Device Manager** from the dropdown.
4. Click the in the Actions column of the device you'd like to edit.
   Or, if you want to create an emulator, click **Create Device** and follow the prompts.
5. Click **Change** in the line that displays the version number.



6. In the pop-up window, select API 30 version or later.
7. Click **OK**.
8. Scroll down in the window to the Memory and Storage section.



9. Click in the **RAM** field and enter *4096*.

   If you can't change the RAM value, use a newer device for the emulator.

10. Click **Finish**.

11. Click in the Actions column of the device to launch the Android Emulator.

After the device emulator launches you can close Android Studio, but keep the Android Emulator app window open to install the Field Service app in the next section.

Install the Field Service App for Android

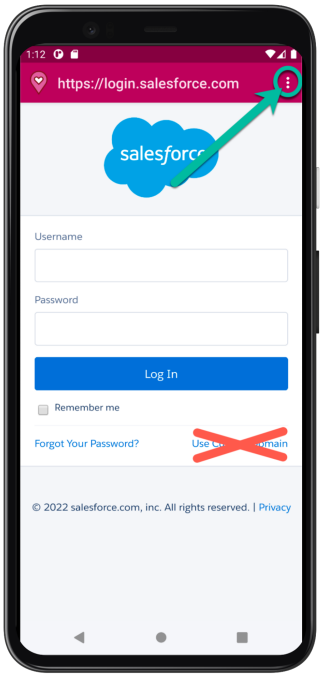
To run the Field Service mobile app in an Android emulator, download and install a virtual device build of the app. After it's installed, open the app and log into your development org.

1. Download the Android virtual device build of the Salesforce Field Service mobile app in the form of an APK file.

   A link to download the latest version of the app is available in this Trailblazer Community post. If you encounter an insufficient permissions page, log in to Trailhead and try the link again.

2. Drag the downloaded `.apk` file into the Android Emulator window.
3. Click a blank space on the Android Emulator's screen and drag up to view the installed apps.
4. Click the newly installed Field Service app.
5. Click **I Agree** to accept the Order Form Supplement agreement.

6. Click the vertical dots button on the top right and select **Change Server**.



7. Select a connection.

   Warning: If you're logging in as a community user for the first time, click **Add New Connection**, fill in the form as indicated in the following bullet list, and then click **Apply** to save the changes.

   - For **Name**, enter a nickname for your connection.
   - For **URL**, enter your org's URL. It must be in the following format: `https://[yourURL].my.salesforce.com`

8. Click the arrow in the top left to go back to the login screen.
9. Enter your username and password for your org.
10. Click **Log In**.
11. Click **Allow** to allow the app to access your Salesforce information.
12. Click through the various permissions screens and allow them the appropriate access.
    When finished, you arrive at the app's home screen.

Debug Lightning Web Components in the Field Service Mobile App

The best way to develop and debug your Lightning web components is the same way you develop and debug *anything* built with HTML, CSS, and JavaScript: with the debugging tools built into your web browser.

- For debugging on Android, use Chrome DevTools
- For debugging on iOS, use Safari Web Inspector

Next, this section walks you through connecting your desktop browser's developer tools to the WebView within the Field Service mobile app. From there, debugging an LWC running on a mobile device is like any other web app debugging session.

IN THIS SECTION:

Install Local Development Server Plugin
Whether you're debugging your component for iOS or Android, you must first install the LWC Development Server for mobile.

Debug in iOS
Connect Safari Web Inspector on your desktop to the WebView in the Field Service mobile app where your LWC is running.

Debug in Android
Connect Chrome DevTools on your desktop to the WebView in the Field Service mobile app where your LWC is running.

Install Local Development Server Plugin

Whether you're debugging your component for iOS or Android, you must first install the LWC Development Server for mobile.

1. In a terminal window in VS Code or Terminal, run the following command to ensure you're using the latest version of Salesforce CLI.

```
sf update
```

   Note: If you encounter an error when updating Salesforce CLI, see Update Salesforce CLI for troubleshooting instructions.

2. In the same terminal window, run the following command to install the LWC Development Server for mobile.

```
sf plugins install @salesforce/lwc-dev-server
```

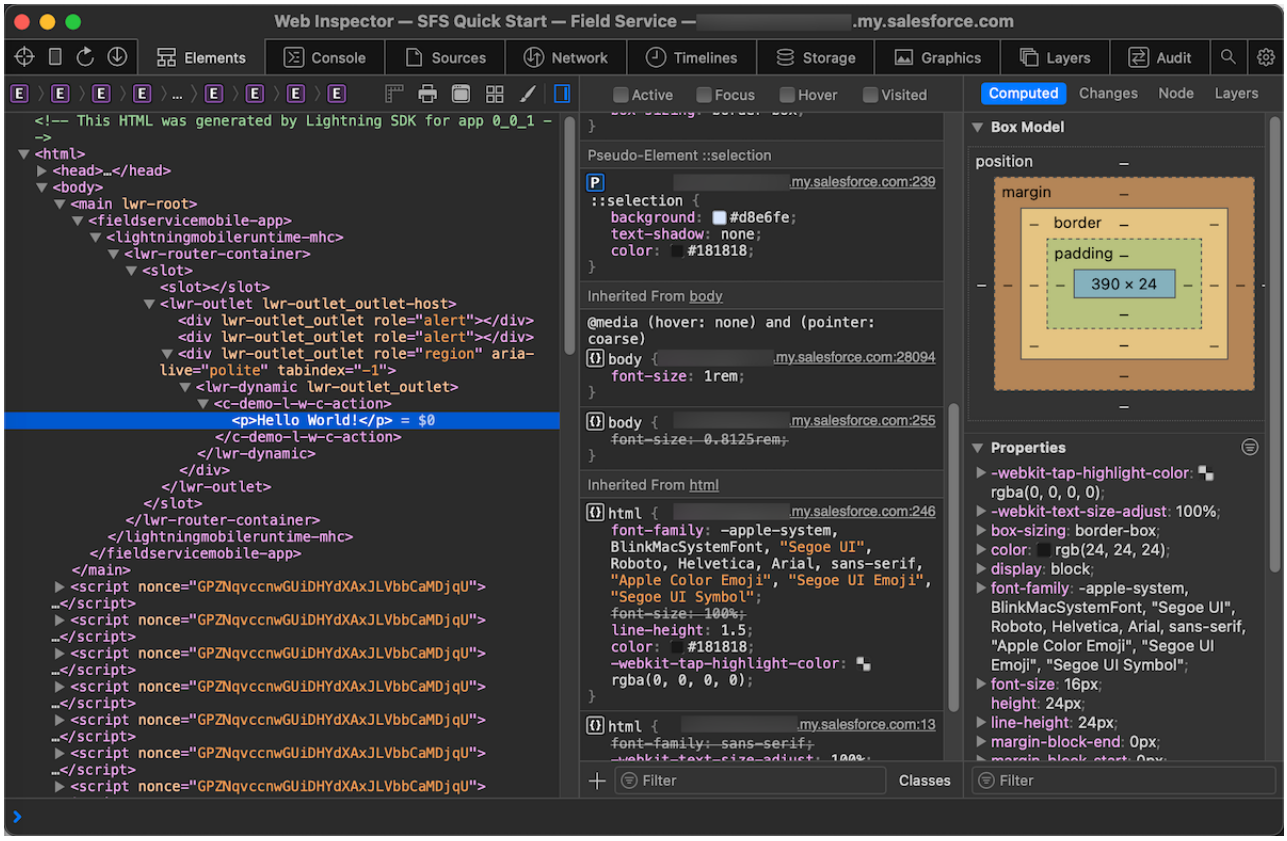
Debug in iOS

Connect Safari Web Inspector on your desktop to the WebView in the Field Service mobile app where your LWC is running.

Note: Debugging in iOS currently only works with Big Sur or later, and requires using the Safari Technology Preview browser.

1. Launch Safari on your desktop.
2. Select **Safari** > **Preferences**.
3. Select **Advanced**.
4. Enable **Show Develop menu in menu bar**. Close the Preferences panel.
5. Select **Develop** > **Simulator -** *`device`* **-** *`version`*, where the Simulator is the one that you've opened with the Field Service app for testing your new Lightning web component.

A window appears that shows the Safari Web Inspector developer tools connected to your simulator. You can use these tools and standard web development techniques to refine and improve your component.



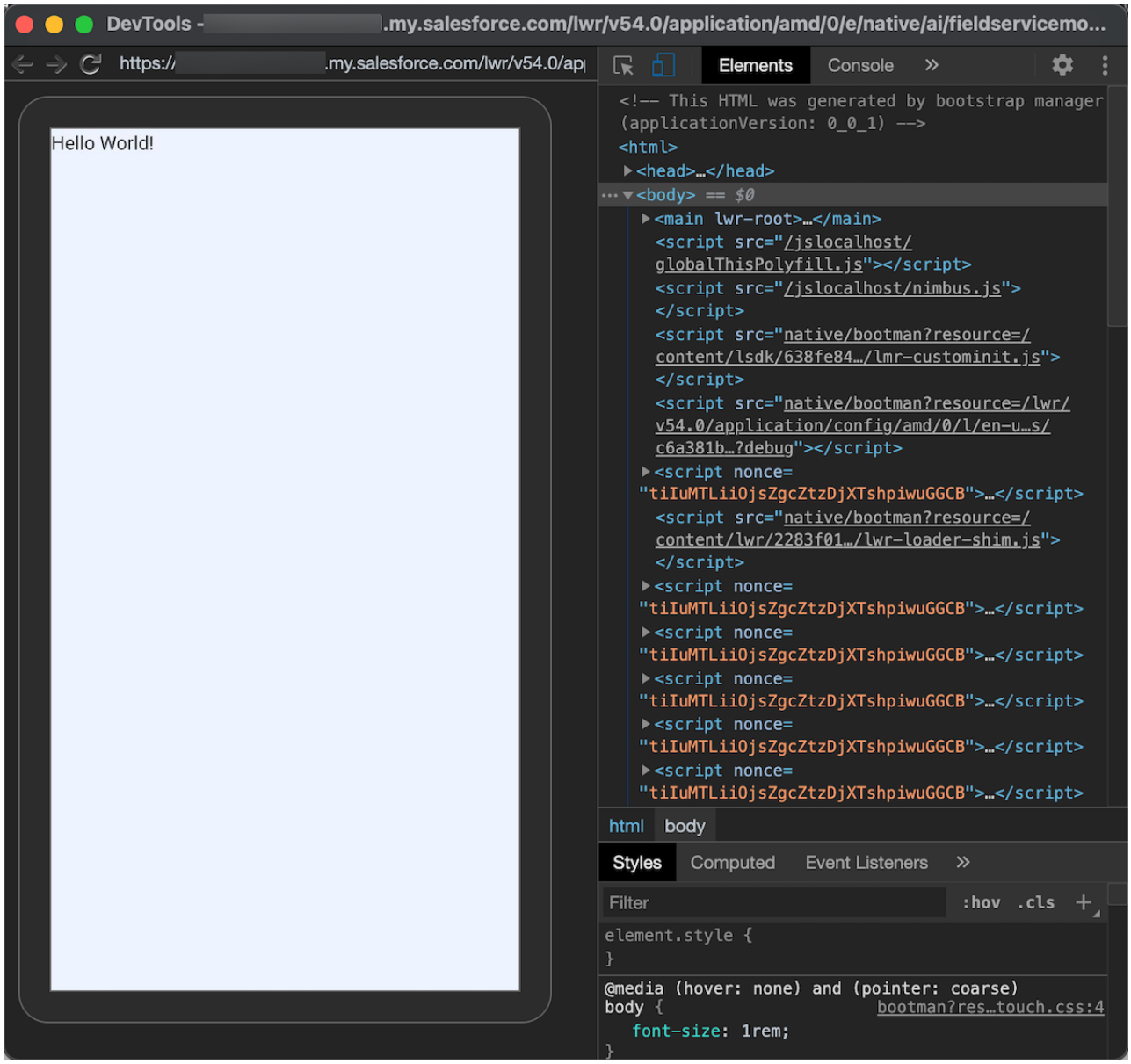
See Apple Web Development Tools for additional information about the Web Inspector and how to use it.

Debug in Android

Connect Chrome DevTools on your desktop to the WebView in the Field Service mobile app where your LWC is running.

1. On your emulated Android device, open the Settings app.
2. Enter *`About emulated device`* into the search bar and click it.
3. Scroll to the bottom of the page and click **Build number** seven times.
   The message "You are now a developer!" appears when you click it enough, indicating that developer mode is enabled for the emulator.
4. Launch Chrome on your desktop.
5. In the location bar, enter *`chrome://inspect/#devices`*.
6. Click **Inspect** for the WebView under the Remote Target emulator you're using.

A window appears containing your emulator with Chrome DevTools connected to it. You can use Chrome DevTools to inspect the LWC element, set breakpoints, and see the console output. As with iOS, you can use the standard web development techniques for working with HTML, JavaScript, and CSS to run, test, debug, and improve your LWC.



See Remote debugging WebViews for additional basics, and Debugging embedded JavaScript in an Android app using Chrome DevTools for a deep dive on debugging JavaScript in an embedded WebView of an Android mobile app.

What Can I Do with Lightning Web Components in the Field Service Field Service Mobile App?

You can create custom components that are accessed from the Actions menu. Actions built with LWCs can be simple or complex, depending on the problem you're solving, and can work even when mobile devices are offline. Use them to add and annotate an image to attach to a service report, order a part, contact a SME, or create a custom dashboard for your mobile workers. Use LWCs in Field Service to do pretty much anything Salesforce can do.

IN THIS SECTION:

Work with LWCs
Use the resources in the following table to learn more about developing with LWCs for the Field Service mobile app.

Work with LWCs

Use the resources in the following table to learn more about developing with LWCs for the Field Service mobile app.

Task	Where to Find It
Learn about and develop with LWCs.	Salesforce Developers: Lightning Web Components Developer Guide
Use the open-source LWCs to find nearby resources, reschedule appointments, and upload and annotate images in the Field Service mobile app.	GitHub: LWC-Mobile-Samples • FindNearby • RescheduleAppointments • UploadAndAnnotateImages
Use the unlocked packages to schedule follow-up appointments and view a mobile dashboard in the Field Service mobile app.	GitHub: LWC-Mobile-Samples • FollowupAppointment • MobileDashboard
Create LWCs that use plug-ins.	Field Service Developer Guide: Add Lighting Web Components for Plug-Ins to the Field Service Mobile App
Find a Partner to help you develop with LWCs.	Find the right Salesforce expertise: Salesforce Partners

Configure Offline Mode in the Field Service Mobile App

Learn what to expect when Automatic Offline Mode is turned on in the Field Service mobile app and how you can adjust its settings.

The Field Service mobile app switches offline when no cellular signal and no wifi connection are available. When the app is in offline mode, it uses cached data, and it refreshes its data when it returns online.

The app determines whether to stay in offline mode by sending regular connection requests to the Salesforce API. After repeated failed requests, it enters or remains in offline mode. This means that in certain situations, the app can be in offline mode even when a cellular signal is available, wifi connection, or both.

1. As an admin, go to the Manage Connected Apps page in Setup.
2. Click **Salesforce Field Service for iOS** or **Salesforce Field Service for Android**. If your team is using both platforms, add a configuration for each.
3. In the custom attribute list, click **New**.
4. Add custom attributes to configure the way the app uses offline mode.
5. After you add custom attributes to configure the app, ask your mobile workers to log out and then log in to the app for the changes to take effect.

You can change the following settings related to offline mode.

Setting	Default Value	Setting Name
Interval of the next Salesforce API connection check after a successful check when the app is online	15 seconds	AO_DELAY_FOR_ONLINE_STATUS
Maximum number of failed requests before the app goes offline	3	AO_CONNECTION_FAILURE_THRESHOLD
Maximum number of successful requests for app to return online	3	AO_CONNECTION_SUCCESS_THRESHOLD
Interval of the next Salesforce API connection check after a failed check	1 second	AO_DELAY_FOR_CHANGING_STATUS
The time the connection request can take before being considered a failed attempt	5 seconds	AO_CONNECTION_CHECK_TIMEOUT

Considerations

- The app can switch between online and offline mode frequently depending on your settings. For example, if you set the Salesforce API connection check to ping every second and configure the app to go offline after one failed check, it can repeatedly switch between online and offline mode.
- Connection checks don't accurately reflect data retrieval calls. As a result, this feature keeps the app online, but data retrieval calls can still be slow.
- If a flow is in progress when the app switches between offline and online mode, the flow can be interrupted.
- If the app is online and you use it in areas of low network connectivity, some users experience long wait times for the data to refresh.

Design with Lightning Web Components to Look Like Salesforce

Elevate your field service operations with visually cohesive mobile experiences. The Field Service mobile app seamlessly integrates the following Lightning web components with the Salesforce UI style.

- Labels for Lightning elements
  - font size
  - Max length of label is three lines
- lightning-input
  - Text + Number
    - width
    - height
    - background-color-focus
    - border-color-focus
    - border-width
    - caret-color-focus
    - margin
    - padding
    - background-color-error-focus

- Datetime Picker
  - width
  - height
  - background-color-focus
  - border-color
- Checkbox Toggle
  - background-color-checked-focus
  - background-color-checked
  - shadow
- Checkbox Button
  - background-color
- lightning-textarea
  - width
  - height
  - background-color-focus
  - border-color-focus
  - border-width
  - caret-color-focus
  - margin
  - padding
  - background-color-error-focus
  - font size
- lightning-combobox (picklist)
  - width
  - height
  - background-color-focus
  - border-color-focus
  - border-width
  - margin
  - padding
- lightning-checkbox-group
  - border around checkbox
  - label font size
  - checkbox size
  - checkbox color border
  - checkbox color background
  - checkbox mark color
  - error state background focus

- lightning-radio-group
  - border around checkbox
  - label font size
  - radio size
  - radio color border thickness
  - radio mark color
- lightning-progress-bar
  - progress bar color
- lightning-progress-indicator
  - progress indicator bar color
- lightning-button (variant - base, neutral, brand)
  - border
  - background-color
  - background-color-focus

Build Custom Lightning Web Components for Service Documents

Customize templates for your specific business and branding needs by creating custom components with Lightning web components (LWCs).

If you need a component for your document template that isn't available in Document Builder standard components, you can easily customize templates for your specific business and branding needs. Creating custom components is simple with LWCs. LWCs for service documents require specific print styling, so if you built custom components for other platforms, such as Field Service Mobile, you must make components for Document Builder.

IN THIS SECTION:

Lightning Web Components for Document Builder: Requirements
Here are the requirements for creating Lightning web components (LWCs) for Document Builder.

Lightning Web Components for Document Builder: Considerations
Considerations and limitations for building Lightning web components (LWCs) for the Document Builder feature, including a list of supported and unsupported utilities, HTML components, and APIs.

Lightning Web Components for Document Builder: Offline Priming
Ensure that your custom Lightning web components (LWCs) for Document Builder work when your mobile workers are offline.

Build Custom Tables for Service Documents with LWCs
Build custom tables for Document Builder with Lightning web components (LWCs).

Code Examples for Document Builder
Build custom components for Document Builder with Lightning web components (LWCs).

Lightning Web Components for Document Builder: Requirements

Here are the requirements for creating Lightning web components (LWCs) for Document Builder.

Use Salesforce Lightning Design System (SLDS) wire adapters

- Don't use aura controllers.
- Don't use Apex. While Apex works online, it returns an empty array when offline.
- Calls to a third-party data source to retrieve data don't work offline.

Use wire adapters for create, update, and delete operations

You can use only these operations, which are logically primed: create, update, delete, `getRecord`, `getRecords`, `getObjectInfo`, and `graphQL`. Use of `graphQL` is recommended. `getRelatedListRecords` is not fully supported for offline use.

You can use these wire adapters to create, update, delete, and get records.

- `import { getRecord } from 'lightning/uiRecordApi'`
- `import { createRecord } from 'lightning/uiRecordApi'`
- `import { deleteRecord } from 'lightning/uiRecordApi'`
- `import { updateRecord } from 'lightning/uiRecordApi'`
- `import { getObjectInfo } from 'lightning/uiObjectInfoApi'`
- `import { gql, graphql } from 'lightning/uiGraphQLApi'`

Remove Aura dependencies

Field Service developers using the Lightning Software Developers Kit (LSDK) can't use any code that relies on Aura because Field Service must be able to execute offline. To do so, use only LWCs.

LWCs that depend on Aura run into trouble at run time. To avoid problems, you must fully evaluate your component's dependencies and go all the way down the chain, ensuring that no code uses Aura.

Ensure LWCs Are Statically Analyzable

If your LWC isn't statically analyzable, also known as Komaci-friendly, Komaci doesn't prime the LWC. Komaci is a suite of packages that provides data flow graph generation and processing capabilities. It defines a schema for an Abstract Data Graph (ADG), which is a serializable metadata-based representation of a data flow graph. Komaci includes an ADG generation service. It also includes a framework for building plug-ins to read, process, and transform an ADG and a runtime engine for hosting an ADG with live inputs and full reactivity.

Komaci provides an abstraction layer for the data flow through an LWC component tree. It includes a utility to analyze an LWC module and generate an ADG and runtime helpers for runtime integration with LWC wires. It also provides an extension for LWC applications to apply progressive data resolution and integrate with the Komaci engine.

Important: For custom LWCs to appear on mobile document preview, you must include this code in your `js-meta.xml` file: `<supportedFormFactor type="Small" />`

Lightning Web Components for Document Builder: Considerations

Considerations and limitations for building Lightning web components (LWCs) for the Document Builder feature, including a list of supported and unsupported utilities, HTML components, and APIs.

Supported Utilities

Here are the supported utilities for building LWCs for Document Builder.

- Borders

- Box
- Description List
- Grid
- Horizontal List
- Hyphenation
- Layout
- Margin
- Name Value List
- Padding
- Print
- Text
- Themes
- Vertical List

Supported Base Components

Here are the supported base components for building LWCs for Document Builder.

- Combobox
- Formatted Date-Time
- Formatted Location
- Formatted Phone
- Formatted Time
- Formatted URL
- Relative Date-Time
- Formatted Address

  Tip: You can't use showMapLink to show Google maps in an iframe

- Formatted Name
- Formatted Number
- Formatted Rich Text
- Formatted Text
- Progress Indicator
- Progress Ring
- Tile

Unsupported Salesforce APIs

For consistency between desktop, mobile, and print, don't use Form Factor GVP or Width Aware Salesforce APIs.

Unsupported HTML Concepts

You can't use these HTML concepts when building LWCs for Document Builder.

- Don't use iframes.

  Iframes don't work offline, and page numbers don't show up properly.

- Don't use CSS Screen media queries.

  CSS Screen media queries cause discrepancies in how your custom component looks on desktop and mobile.

- Don't use CSS `position:fixed`.

  When content is printed, fixed positioning is irrelevant, and elements with position:fixed don't behave as expected. If you want repeating elements, drag an object into the header or footer sections of the canvas.

- Don't use CSS to show objects side by side.

  To show objects side by side, use the standard grid component. Avoid CSS properties such as `float`, `display: flex`, `display: inline`, `display: inline-block`, `display: inline-flex`, `display: grid`, `display: inline-grid`, `display: table`, or `display: table-row`.

- Don't use `:nth-of-*` CSS Selectors.

  When working with content that spans multiple pages, pseudo-selectors such as `:nth-of-type` or `:nth-child` typically don't work. As a workaround, use `lwc:if` templates and application of classes to customize the style of your service document. Learn more about `lwc:if` in the Lightning Web Components Developer Guide.

Pagination Considerations

When creating LWCs for Document Builder, consider these pagination limitations.

- Service documents are limited to 250 pages. Document Builder can't process a document with more than 250 pages into a PDF.
- Avoid CSS page declarations. Don't declare `@page` on custom components. The ability to modify page formatting for print is built into Document Builder.
- If an image is too large to fit onto the remainder of a page, it's pushed to the next page. If an image, by itself on a page, is still too large to fit, it's resized to fit on one page.
- If the page number component is dragged onto the body section of the page, it shows the page number in which it shows up. We recommend that you use this component in header or footer regions.
- The standard signature component isn't split across pages.
- If the field component spans pages, the field labels are inline and repeat.
- Column headers repeat if related list content spans across pages.
- Use the standard grid component for creating layouts because it provides comprehensive layout options. If you must use SLDS grid classes, don't apply slds-col and slds-grid CSS classes to a single element. Instead, apply each layout class to its own element. Here's an example.

  ```
  <!-- Bad --><div class="slds-col slds-grid"></div><!-- Good --><div class="slds-col"><div
   class="slds-grid">
  ```

  The simpler the grid, the better. Features such as column wrapping, reordering, gutters, and vertical grids aren't supported.
- PDF conversion doesn't compress static resources. Don't place large static resources (contentAssets) on the page. Resize and compress them before using them on a service document.

Lightning Web Components for Document Builder: Offline Priming

Ensure that your custom Lightning web components (LWCs) for Document Builder work when your mobile workers are offline.

To ensure that your custom components have offline support, install the Komaci VS Code Plug-in. With the plug-in, you can analyze and identify which of your custom LWC components mobile workers can use offline. Without the plug-in, you can't tell whether a component works while a mobile device is offline. When a component can be used offline, mobile workers can still capture data, even when they don't have access to a network. Before the user goes offline, we cache the data required to make the component functional. Then, when the user has wireless access again, we sync the data that's captured while mobile devices are offline.

Install the Komaci ESLint Visual Studio Code (VS Code) Plug-in

We suggest that you use VS Code. If you're using VS Code, install the ESLint VS Code plug-in, and then press CTRL+SHIFT+P to open the command palette to ESLint: Restart ESLint Server. Then, in the bottom output tab (with ESLint) selected, you can debug it.

You can also run `yarn lint:lwc` inside VS Code's terminal.

If you're not using VS Code, check out the public repository of the ESLint Komaci Plug-in that can configure your tool.

If you want a deep dive, check out npm for information about the plug-in, including code. But if you're ready to implement the code you can use either of these commands.

- Command 1: `yarn add --dev @salesforce/eslint-plugin-lwc-graph-analyzer`
- Command 2: `npm install —save-dev @salesforce/eslint-plugin-lwc-graph-analyzer`

After you input the code, check your project directory and update the `.eslintrc` file to reference our recommended configuration.

The Komaci engine is fully portable to any JavaScript runtime environment and has no dependencies on web-based technologies or APIs, such as the DOM. You can use Komaci as an independent data fetching service for LWC applications. The engine can analyze LWC modules and prefetch their respective data dependencies in a host environment outside of a runtime window, such as a Web Worker, JavaScriptCore, or V8 host environment.

Build Custom Tables for Service Documents with LWCs

Build custom tables for Document Builder with Lightning web components (LWCs).

While Document Builder offers a standard component to create tables for related records, you can also build your own custom tables to suit your company's needs.

Design Your Table to Show Data Properly

To ensure that your tables show data properly, use HTML tables to build your LWC for custom tables. The right HTML table formatting organizes your data properly, even when the table overflows or breaks between pages. If information in the table does overflow, column headers repeat on the next page.

Here's a sample markup that calls out the header row, ensuring that the header is repeated across pages.

Example:

```
<table>
  <thead>
       <tr>
           <th>Company</th>
           <th>Contact</th>
           <th>Country</th>
       </tr>
   </thead>
   <tbody>
       <tr>
           <td>Alfreds Futterkiste</td>
```

```
            <td>Maria Anders</td>
            <td>Germany</td>
        </tr>
        <tr>
            <td>Centro comercial Moctezuma</td>
            <td>Francisco Chang</td>
            <td>Mexico</td>
        </tr>
    </tbody>
</table>
```

Code Examples for Document Builder

Build custom components for Document Builder with Lightning web components (LWCs).

With custom components, the sky's the limit, but these code samples can get you started on your own Document Builder customization journey.

Important: For custom LWCs to appear on mobile document preview, you must include this code in your `js-meta.xml` file: `<supportedFormFactor type="Small" />`

Static Text

The header and medium header base components come with color and padding configuration. This code creates a medium header component.

```
// Below is the code for the HTML block:
<template>
    <h2 class="slds-text-heading_medium" style={inlineStyle}>{text}</h2>
</template>

// Below is the code for the javascript block:
import { LightningElement, api } from "lwc";

export default class MediumHeaderText extends LightningElement {
    @api text;
    @api colorhex;
    @api topPadding;
    @api leftPadding;
    @api rightPadding;
    @api bottomPadding;

    get inlineStyle() {
        return `color:#${this.colorhex};
padding:${this.topPadding}px ${this.rightPadding}px ${this.bottomPadding}px
${this.leftPadding}px`;
    }
}

// Below is the extensible markup language block
<?xml version="1.0" encoding="UTF-8" ?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>57.0</apiVersion>
```

```
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__ServiceDocument</target>
    </targets>
    <targetConfigs>
        <targetConfig targets="lightning__ServiceDocument">
            <supportedFormFactors>
                <supportedFormFactor type="Large" />
            </supportedFormFactors>

            <property name="text" type="string" default="A medium header" />
            <property name="colorhex" type="string" default="1B3971" />
            <property name="topPadding" type="string" default="0" />
            <property name="rightPadding" type="string" default="0" />
            <property name="leftPadding" type="string" default="0" />
            <property name="bottomPadding" type="string" default="0" />
        </targetConfig>
    </targetConfigs>

</LightningComponentBundle>
```

Static Images

Here's some sample code to create custom static images. This code creates a Static Image component.

Important: Don't use static resources, as they aren't supported offline and are deprecated. Only use ContentAsset in your LWC. See Using Assets in LWC, How to Create Assets, and Viewing and Editing Assets.

```
// Below is the HTML block
<template>
    <div>
        <div class="image" style={inlineStyle}>
            <img src={salesforceUrl} />
        </div>
    </div>
</template>

// Below is the JavaScript block
import { LightningElement, api } from "lwc";
import SALESFORCE_LOGO from "@salesforce/contentAssetUrl/salesforce";

export default class DemoSalesforceLogo extends LightningElement {
    @api width;
    @api height;

    // Expose the static resource URL for use in the template
    get salesforceUrl() {
        return SALESFORCE_LOGO;
    }

    get inlineStyle() {
        return `width: ${this.width}px;height: ${this.height}px`;
    }
}
```

```
// Below is the Extensible Markup Language block
<?xml version="1.0" encoding="UTF-8" ?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
   <apiVersion>54.0</apiVersion>
   <isExposed>true</isExposed>
   <targets>
       <target>lightning__ServiceDocument</target>
   </targets>
   <targetConfigs>
       <targetConfig targets="lightning__ServiceDocument">
           <supportedFormFactors>
               <supportedFormFactor type="Large" />
               <supportedFormFactor type="Small" />
           </supportedFormFactors>

           <property name="width" type="integer" />
           <property name="height" type="integer" />
       </targetConfig>
   </targetConfigs>

</LightningComponentBundle>
```

This code uses SFDX to deploy contentAssets.

```
// Below is the XML block:
<?xml version="1.0" encoding="UTF-8" ?>
<ContentAsset xmlns="http://soap.sforce.com/2006/04/metadata">
   <isVisibleByExternalUsers>true</isVisibleByExternalUsers>
   <language>en_US</language>
   <masterLabel>salesforce</masterLabel>
   <relationships>
       <organization>
           <access>VIEWER</access>
       </organization>
       <workspace>
           <access>INFERRED</access>
           <isManagingWorkspace>true</isManagingWorkspace>
           <name>sfdc_asset_company_assets</name>
       </workspace>
   </relationships>
   <versions>
       <version>
           <number>1</number>
           <pathOnClient>salesforce.png</pathOnClient>
       </version>
   </versions>
</ContentAsset>
```

Configure Deep Linking for the Field Service Mobile App

Your field technicians rely on their mobile devices to perform complex tasks, and some of those tasks require them to navigate between multiple tabs, flows, or actions. With deep linking, you can send users directly to the right spot in the app.

Deep linking enables mobile apps to interact with each other using links called URIs, or Uniform Resource Identifiers. URIs have the power to point users to a specific location in a mobile app. Field Service has a pre-defined URI scheme that allows users to launch the Field Service app from a link in an email, website, or third-party application. An administrator or developer can create custom URIs to display specific tabs in the Field Service app. For example, you could integrate a third-party app that your field technicians use for billing.

If an unauthenticated user taps on a Field Service URI, they will be taken to the login screen. Once they log in, they need to tap the URI again to view the linked page.

IN THIS SECTION:

Deep Linking Schema for the Field Service Mobile App
Use the predefined URI schema for the Field Service mobile app on Android and iOS devices.

Hide Deep Linking Security Dialog
Each time a user opens a deep link to an action from the Field Service mobile app, a security dialog prompts the user to confirm the action. You can hide this Launch action? dialog by configuring the deep link URL with a security key.

Pass Data to an LWC with Deep Linking
Pass data between Lightning web components (LWC) or from external apps to an LWC with deep linking. LWC with deep linking is available for the Field Service mobile app on both Android and iOS devices.

Deep Linking Schema for the Field Service Mobile App

Use the predefined URI schema for the Field Service mobile app on Android and iOS devices.

URI Schema Format

The available URI schema formats are:

- `<com.salesforce.fieldservice>://v1/sObject/<id>/<action>?<params>`
- `<com.salesforce.fieldservice>://v1/globalaction/<api_name>?<params>`

Let's break down each parameter of the URI scheme format.

- `com.salesforce.fieldservice`—The supported Salesforce schema name.
- `v1`—The current version of deep linking. The version is a static value.
- `sObject`—Indicates a single instance of a Salesforce object. The schema supports all object types that are supported in the Field Service app, such as work order, service appointment, and assets.
- `id`—The unique single instance of a Salesforce object. `id` lengths of 15 and 18 characters are supported.
- `action`—A supported URI schema operation. Valid actions are listed in the Supported URI Schemes section. If no action is specified, URIs direct the work orders, service appointments, and other object types to the Overview or Details tab.
- `globalaction`—The name of the global quick action.
- `api_name`—Some deep links require an API name in order to perform their function. For instance, the quick action URI requires a quick action API name, and the flow URI requires a flow API name.
- `params`—The field and value pairs used in the global quick action.

Supported URI Schemes

`com.salesforce.fieldservice://v1/sObject/<id>`
Links to the Overview or Details tab.

- Work order and work order line item IDs are directed to the Overview tab.

- Service appointment IDs are directed to the Details tab, with these exceptions.
  - If the service appointment parent is a work order, the deep link is directed to the work order Overview tab for iOS and the service appointment Details tab for Android.
  - If the service appointment parent is a work order line item, the deep link is directed to the work order line item Overview tab for iOS and the service appointment Details tab for Android.
- All other object types are directed to the object's Details tab.

Example: `com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA`

This example links to the Overview tab for a work order.

`com.salesforce.fieldservice://v1/sObject/<id>/details`

Links to the Details tab for all object types.

Example: `com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/details`

This example links to the Details tab for a work order.

`com.salesforce.fieldservice://v1/sObject/<id>/related`

Links to the Related tab for all object types.

Example: `com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/related`

This example links to the Related tab for a work order.

`com.salesforce.fieldservice://v1/sObject/<id>/products`

Links to the Products tab for all applicable object types. If an object doesn't have a Products tab, it links to the Details tab instead.

Example: `com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/products`

This example links to the Products tab for a work order.

`com.salesforce.fieldservice://v1/sObject/<id>/feed`

Links to the Feed tab for all applicable object types. If an object doesn't have a Feed tab, it links to the Details tab instead.

Example: `com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/feed`

This example links to the Feed tab for a work order.

`com.salesforce.fieldservice://v1/sObject/<id>/location`

Links to the Location tab for all applicable object types. If an object doesn't have a Location tab, it links to the Details tab instead.

Example: `com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/location`

This example links to the Location tab for a work order.

`com.salesforce.fieldservice://v1/sObject/<id>/edit`

Links to the edit page for the specified Salesforce record.

Example:

`com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/edit?Subject=Example%20Subject`

This example links to the edit page for this record and fills the Subject field with the Example Subject value.

`com.salesforce.fieldservice://v1/sObject/<id>/flow/<api_name>`

Launches a Field Service Mobile Flow from within the mobile app. For example, you can use this URL to launch a custom service closure flow. To use this feature, the app extension must reference this flow. You can find the extension in Setup under Field Service Mobile Settings. Verify that an extension in the App Extension section includes this flow. To learn more, see Connect a Flow to the Field Service Mobile App. Use the URL-encoded name of the extension in the URI.

Example:

`com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/flow/service_response_flow?Subject=Example%20Subject`

This example launches the "service_response_flow" flow passing in "Example Subject" as the value for the "Subject" input variable.

`com.salesforce.fieldservice://v1/sObject/<id>/quickaction/<api_name>`

Launches a quick action from within the mobile app. These quick action types are supported: Create a Record, Update a Record, and Field Service Mobile Extension. You can pass inputs into the quick action using parameters on this URL.

Example:

```
com.salesforce.fieldservice://v1/sObject/0WO5500000001UZGAA/quickaction/close_order
```

This example launches the `close_order` quick action.

`com.salesforce.fieldservice://v1/globalaction/<api_name>?<params>`

Launches a global quick action from within the mobile app. These global quick action types are supported: Create a Record, Update a Record, and Field Service Mobile Extension. You can pass inputs into the global quick action using parameters on this URL.

Example:

```
com.salesforce.fieldservice://v1/globalaction/Create_Work_Order?Subject=Example%20Subject
```

This example launches a global quick action to create a work order.

`com.salesforce.fieldservice://v1/login/<server_name>/<server_url>`

Adds a connection to a server using the supplied URL. To access the server, users click the deep link and enter their existing login credentials. The server is saved to the Change Server list, which gives users quick access because it eliminates the need for repeated log-in attempts.

Parameters:

- `<server_name>` is a user-friendly name used to identify the server. Avoid spaces, punctuation, and other "illegal" characters, especially if workers must log in to multiple sites.
- `<server_url>` is the URL used by the Salesforce Field Service Mobile app to log in. Don't include "http" or "https" when specifying the URL.

Example: `com.salesforce.fieldservice://v1/login/employee/my.site.com/contractor/login`

This example sets the login URI, sends users directly to `https://my.site.come/contractor/login`, and saves it in the Change Server list as `employee`.

Security Considerations:

- Ensure that you trust the source of a link before clicking it, especially when adding usernames and passwords. For this reason, we recommend that your workers are trained on best practices and that the login link isn't sent directly via email.
- In many email clients and messaging apps, regardless of the mobile operating system, the text body doesn't support active links to mobile apps. Opening the app when a link is launched is supported, but it depends on your technology and distribution choices. If you encounter issues with your link, first, try testing it in a web browser. If the link opens in the app, it is formatted properly. In cases where the web link works but your chosen distribution method doesn't, find a different way to share the link. For a seamless experience, use a web link that redirects to the app works. Other options like QR codes, website link, PDF, or text message work well, depending on the app and the operating system used to open the link. It isn't guaranteed that all third-party apps on iOS and Android support opening the links.

Parameter Passing for Deep Linking

To pass parameters in a deep linking schema such as a quick action or flow URL, these parameter types are valid.

- URL-encoded text parameters. For example, *`Short%20text%20input`*.
- Numbers or currency. For example, *`1`* or *`1.45`*.
- Boolean values. For example, *`true`* or *`false`*. These values aren't case-sensitive. If no parameter is passed, the default is *`false`*.

- URL-encoded date and dateTime parameters. For example, `2019-12-11T17%3A01%3A00.000%2B0000`. Include a URL-encoded timezone, such as +0000 for GMT or -0400 for AST.

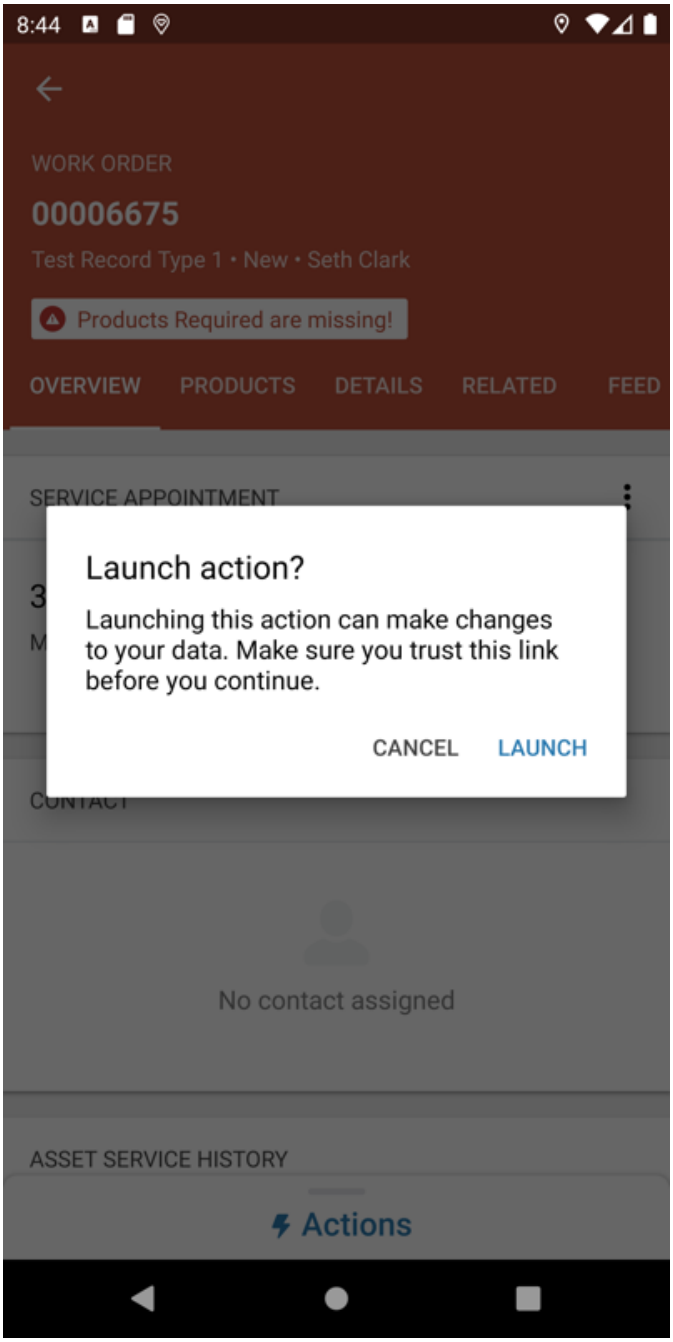
Other parameters aren't supported in deep linking URI schema.

To pass more than one parameter, use an ampersand (&) between them. For example, `firstname=John&lastname=Doe`.

We don't limit the number of parameters, and each parameter value can contain up to 100,000 characters. While deep links can encode up to 1 MB, it's best practice to encode smaller amounts of data.

Hide Deep Linking Security Dialog

Each time a user opens a deep link to an action from the Field Service mobile app, a security dialog prompts the user to confirm the action. You can hide this Launch action? dialog by configuring the deep link URL with a security key.



Complete these steps to configure the security key.

Step 1: Create a private and public key

1. For Linux/Mac, run these commands in the terminal to create a private and public key. The .pem key files are created in the folder where you execute the commands.

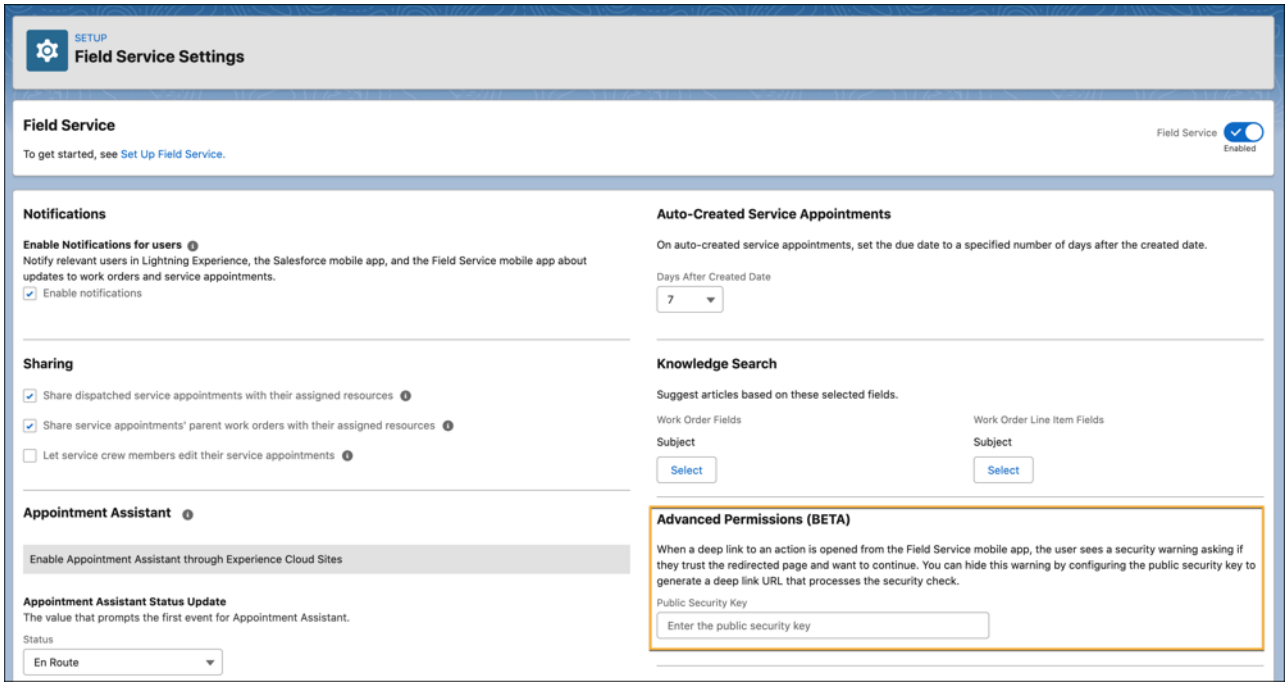
   Note: We don't officially support Apex for signing the keys, but if you're signing deep links in Apex, try using the ECDSA-SHA256 algorithm.

```
openssl ecparam -genkey -name prime256v1 -noout -out private.pem
openssl ec -in private.pem -pubout -out public.pem
```

2. Keep a copy of the private.pem and public.pem files so that you can sign URLs in the future without generating new key pairs. The private.pem file contains the security key.
3. Open the public.pem file, and copy the public key, excluding the header and footer and without spaces or a new line. Here's a sample public key.

```
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEkvkDcFieJenYABN8wOLlE2VomNt2
9/tcTyj+B06ZndRkTjs7+XwrjHe/wOZvjkdYvewhIByLI6uDTYZixDhO1A==
```

4. Copy the public key into the Advanced Permissions section of the Field Service Settings UI. This step is required to turn on this feature.



Step 2: Sign the deep link URL

1. In the terminal, change the directory to the folder containing the private.pem file.
2. Generate the deep link URL, and copy it to your clipboard.

```
// Base URL.
com.salesforce.fieldservice://v1/sObject/<id>/<action>

// URL with additional query parameters.
com.salesforce.fieldservice://v1/sobject/<id>/<action>?param1=value1&param1=value2
```

3. To create a signature using the private key, run this command.

```
pbpaste | openssl dgst -sha256 -sign private.pem | openssl base64 | tr '/+' '_-' | tr
-d '=' | tr -d '\n' | pbcopy
```

The command creates a signature using OpenSSL and Base64 encoding. Here's a breakdown of the command parts.

- The `pbpaste` command pastes from the pasteboard.
- The `openssl dgst` command creates a hash with the sha256 algorithm signed by the private key.
- The `openssl` command makes that the output base64 encoded.
- The `tr` command replaces "/+" characters with "_-", and then deletes all "=" characters and new line characters "\n".

- The `pbcopy` command copies the output back to the pasteboard.

If you're using a dynamic deep link URL, generate a new signature for each unique set of URL parameter values. The signature is unique for a URL string. For example, if the `<id>` parameter is dynamic, generate a new signature when the `<id>` value changes.

4. Sign the deep link URL by adding the query parameter `__signature=<sig>` to the end of the URL. Replace `<sig>` with the signature, which you can paste from your clipboard. If you have additional query parameters, place the signature parameter after them.

```
// Signed URL.
com.salesforce.fieldservice://v1/sObject/<id>/<action>?__signature=<sig>

// Signed URL with additional query parameters.
com.salesforce.fieldservice://v1/sobject/<id>/<action>?param1=value1&param1=value2&__signature=<sig>
```

Now you can send the deep link URL with the signature as before. When the technician clicks the URL, the deep link opens in the Field Service app. The app verifies the signature using the public key. If the public key hash matches, the URL is loaded without the Launch action? dialog. If the public key hash doesn't match, the user must confirm the action.

Pass Data to an LWC with Deep Linking

Pass data between Lightning web components (LWC) or from external apps to an LWC with deep linking. LWC with deep linking is available for the Field Service mobile app on both Android and iOS devices.

To pass parameters using deep linking, such as passing data into an LWC form, use the quick actions URI schema with your LWC quick action as the `<api_name>`, followed by the parameter key and value pairs.

```
com.salesforce.fieldservice://v1/sObject/<id>/quickaction/<api_name>?<parameterKey1>=<parameterValue1>&<parameterKey2>=<parameterValue2>&...
```

Here's an example of a deep link URL that passes the first and last name of Jane Doe to an LWC using a quick action called `LWC_Pass_Fields`.

```
/quickaction/LWC_Pass_Fields?FirstName=Jane&LastName=Doe
```

To pass the data, also update the LWC source code of the current page reference, the LWC that the deep link opens. Here's an example of an LWC that passes the parameter value into the corresponding parameter key of the current page reference.

```
import {CurrentPageReference} from 'lightning/navigation';

// Declare the variable for the parameter value.
parameterValue;

// Call the page reference that describes the current LWC page.
@wire(CurrentPageReference)
setCurrentPageReference(currentPageReference) {
  // Pass parameter values using the currentPageReference state attribute.
  // Replace <parameterKey> with the parameter key name used in the deep link URL.
  this.parameterValue = currentPageReference.state.<parameterKey>;
}
```

Add Lighting Web Components for Plug-Ins to the Field Service Mobile App

Enhance the Field Service mobile app by creating Lightning web components (LWCs) for plug-ins.

- **Access a Mobile Device's Biometrics Capabilities** (Mobile and Offline Developer Guide)

A Lightning web component can use a device's biometrics functionality to prompt a user to confirm their identity. When these biometrics-related actions occur, the result is returned to the Lightning web component that invoked it. Biometrics checks are managed locally on the mobile device, and don't need a network connection. However, BiometricsService requires access to platform-specific APIs that are available only within compatible Salesforce mobile apps.

- **Scan Documents on a Mobile Device** (Mobile and Offline Developer Guide)

  A Lightning web component can use a device's camera and optical character recognition (OCR) to scan documents. When a document is successfully scanned, text data extracted from the scanned document is returned to the Lightning web component that invoked it. DocumentScanner recognizes machine printed text, not handwriting.

- **Monitor Geofence Regions on a Mobile Device** (Mobile and Offline Developer Guide)

  A Lightning web component can use a mobile device's location features to determine a user's current location to the user's proximity to areas that may be of interest, or to perform location-related tasks. The longitude, latitude, and radius define a geofence around the regions of interest. Geofence location is determined locally on the mobile device, and doesn't need a network connection. However, GeofencingService does require a GPS signal from the device. For Android devices, Google Location Accuracy must be enabled in the system settings. GeofencingService does require access to platform-specific APIs that are available only within compatible Salesforce mobile apps.

- **Use Location on a Mobile Device** (Mobile and Offline Developer Guide)

  A Lightning web component can use a mobile device's location features to determine the current location of the device and, by association, the person who is holding it. You can access the device's current location at a moment in time, or you can subscribe to location changes, and receive updates to the device's location when it changes significantly. Location is determined locally on the mobile device, and doesn't need a network connection. LocationService does require access to platform-specific APIs that are available only within compatible Salesforce mobile apps.

- **Interact with NFC Tags on a Mobile Device** (Mobile and Offline Developer Guide)

  A Lightning web component can use a device's native NFC functionality to read, erase, and write to NFC tags. When an NFC operation is successful, the text data extracted from the NFC tag or a simple success message is returned to the Lightning web component that invoked it.

- **Accept On-Site Payments with Tap-to-Pay** (Mobile and Offline Developer Guide)

  A Lightning web component can use a mobile device to let your customers use the Tap-to-Pay capability of the Payments plug-in to pay mobile workers directly. The Field Service mobile app then integrates with Pay Now to connect the Lightning web component to a secure payment system that processes the interaction. The PaymentsService plugin allows Field Service mobile workers to collect payments from their customers using Tap to Pay. This service integrates with Salesforce Payments and Stripe as a payment provider. Tap to Pay is supported on iOS and Android devices with Stripe as the payment provider.

IN THIS SECTION:

Scan Barcodes on a Mobile Device

A Lightning web component can use a mobile device's camera and mobile OS platform features to scan a barcode, such as a UPC symbol or QR code. When a barcode is successfully scanned, the data that was read from the barcode is returned to the Lightning web component that invoked it.

Capture a Space on a Mobile Device

Create a Lightning web component (LWC) that uses the AR SpaceCapture plugin to create a 2D or 3D model of a room. The model is created by interactively scanning the physical environment using the device's Lidar capabilities.

Scan Barcodes on a Mobile Device

A Lightning web component can use a mobile device's camera and mobile OS platform features to scan a barcode, such as a UPC symbol or QR code. When a barcode is successfully scanned, the data that was read from the barcode is returned to the Lightning web component that invoked it.

Scanning is performed locally on the mobile device and doesn't need a network connection. BarcodeScanner does require access to platform-specific APIs that are available only within compatible Salesforce mobile apps.

Important: BarcodeScanner does not and cannot function when running in a web browser, whether running on a desktop or mobile device.

BarcodeScanner provides to your component a string value of the data encoded in a scanned barcode. It doesn't attempt to interpret or process the decoded value.

Note: This barcode scanner plugin is for use only in the Field Service mobile app.

Use the BarcodeScanner API

To develop a Lightning web component with barcode scanning features, use the BarcodeScanner API.

- Import BarcodeScanner into your component definition to make the BarcodeScanner API functions available to your code.
- Test to make sure BarcodeScanner is available before you call scanning lifecycle functions.
- Use the scanning lifecycle functions to start, continue, and stop scanning.

Note: We recommend using the modern `scan()` and `dismiss()` APIs in your LWC scanning code to streamline your development experience. The legacy APIs `beginCapture()`, `resumeCapture()`, and `endCapture()` are still available, but will be retired in a future release.

Add BarcodeScanner to a Lightning Web Component

In your component's JavaScript file, import BarcodeScanner using the standard JavaScript `import` statement. Specifically, import the `getBarcodeScanner` factory function from the `lightning/mobileCapabilities` module, like so:

```
import { getBarcodeScanner } from "lightning/mobileCapabilities";
```

After it's imported into your component, use the factory function to get an instance of BarcodeScanner. With your BarcodeScanner instance, use the utility functions and constants to verify scanner availability and to configure scans. Use the scanning lifecycle functions to perform scanning operations.

Test BarcodeScanner Availability

BarcodeScanner depends on physical device hardware and platform features. A component that uses BarcodeScanner renders without errors on a desktop computer, but scanning functions fail. To avoid these errors, test if BarcodeScanner functionality is available before you use it.

```
import { LightningElement } from 'lwc';
import { getBarcodeScanner } from 'lightning/mobileCapabilities';
export default class ImplementBarcodeScanner extends LightningElement {
  const myScanner = getBarcodeScanner();

  barcodeResults = 'Nothing scanned yet!';

  handleBeginScanClick(event) {
```

```
    if(myScanner.isAvailable()) {
      // Perform scanning operations
      let scanningOptions = {
        "barcodeTypes": ["code128","code39", "code93", "ean13", "ean8", "upca", "upce",
"qr", "datamatrix", "itf", "pdf417"], "instructionText":"Position barcode in the scanner
view.\nPress x to stop.",
"successText":"Successful Scan!"
      };
      myScanner.scan(scanningOptions)
        .then((results) => {
          // Do something with the results of the scan
          this.barcodeResults = '';
          results.forEach(result => {
            this.barcodeResults += 'type: ' + result.type + ', value: ' + result.value +
'\n';
          });
        })
        .catch((error) => {
          // Handle cancellation and scanning errors here
          this.barcodeResults = 'Error code: ' + error.code + '\nError message: ' +
error.message;
        })
        .finally(() => {
          myScanner.dismiss();
        });
  } else {
      // Scanner not available
      // Not running on hardware with a scanner
      // Handle with message, error, beep, and so on
      this.barcodeResults = 'Problem initiating scanner.  Are you using a mobile device?';

    }
  }
}
```

Supported Barcode Types

- aztec
- code39
- code93
- code128
- dataMatrix
- ean8
- ean13
- interleaved2of5
- pdf417
- qr
- upce

Scan a Barcode

Scan with BarcodeScanner using the scanning lifecycle functions.

- Start a scan with `scan(options)`.
- Handle the result of the scan, which is returned in the form of a promise.
- End the scan with `dismiss()`.

For example:

```
myScanner
  .scan(scanningOptions)
  .then((result) => {
    // Do something with the result of the scan
    console.log(result);
    this.scannedBarcode = result.value;
  })
  .catch((error) => {
    // Handle cancellation and scanning errors here
    console.error(error);
  })
  .finally(() => {
    myScanner.dismiss();
  });
```

See `scan(options)` for more details of how to handle scan results, handle errors, and so on.

SEE ALSO:

BarcodeScanner API

Capture a Space on a Mobile Device

Create a Lightning web component (LWC) that uses the AR SpaceCapture plugin to create a 2D or 3D model of a room. The model is created by interactively scanning the physical environment using the device's Lidar capabilities.

Note: This AR SpaceCapture plug-in is supported only by iOS devices that have Lidar (iPhone 12 pro+ and iPad Pro and above), and that are running on iOS 16.0+ and iPadOS 16.0+.

Note: This AR SpaceCapture plugin is for use only in the Field Service mobile app.

IN THIS SECTION:

Use the AR SpaceCapture API

To develop an LWC with AR Space Capture features, use the AR SpaceCapture API as your method for accessing a device's native AR functionality.

AR SpaceCapture User Experience

Your component can deliver any user experience you desire, but you must follow a common flow for any component that calls for an AR SpaceCapture scan.

AR SpaceCapture Example

The component's HTML template is minimal and includes a button that initiates the room scan.

AR SpaceCapture API
Access a device's camera and AR capabilities from an LWC.

SEE ALSO:
Apple's RoomPlan

Use the AR SpaceCapture API

To develop an LWC with AR Space Capture features, use the AR SpaceCapture API as your method for accessing a device's native AR functionality.

1. Add AR SpaceCapture to an LWC

In your component's JavaScript file, import ARSpaceCapture using the standard JavaScript import statement. Specifically, import the `getARSpaceCapture()` factory function from the `lightning/mobileCapabilities` module.

```
import {getARSpaceCapture} from 'lightning/mobileCapabilities';
```

After it's imported into your component, use the factory function to get an instance of ARSpaceCapture. With your ARSpaceCapture instance, use the utility functions and constants to verify availability. Then use the feature functions to perform the associated functionality.

2. Test AR SpaceCapture Availability

AR SpaceCapture depends on physical device hardware and platform features. A component that uses SpaceCapture renders without errors on a desktop computer or in a mobile browser, but SpaceCapture functions fail. To avoid these errors, test if SpaceCapture functionality is available before you use it.

```
handleCheckARSpaceCaptureClick(event) {
    const myARSpaceCapture = getARSpaceCapture();
    if(myARSpaceCapture.isAvailable()) {
  // Perform next operations
  } else {
 // AR SpaceCapture isn't available, or consuming app hasn't implemented it
 // Not running on hardware with AR functionality, etc.
 // Handle with message, error, beep, and so on
  }
}
```

3. Start an AR SpaceCapture Scan

Start a room scan with the `scanRoom()` function. You can then handle the JSON file that represents the USDZ file as required.

For example:

```
handleScanRoomClicked(event) {
    if (this.myARSpaceCapture != null && this.myARSpaceCapture.isAvailable()) {
        this.myARSpaceCapture.scanRoom()
            .then((arSpaceCaptureResult) => {
                console.log(JSON.stringify(arSpaceCaptureResult, undefined, 2))
            })
            .catch((error) => {
                console.log(error);
            });
```

```
    }
}
```

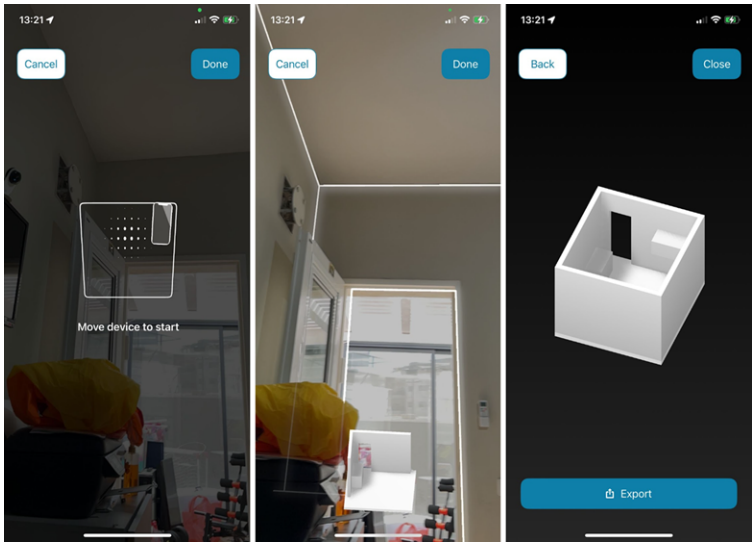
Considerations:

- The floor must be leveled when starting the scan.
- A scanned space includes only items that are on the same level.

AR SpaceCapture User Experience

Your component can deliver any user experience you desire, but you must follow a common flow for any component that calls for an AR SpaceCapture scan.

The user performs an action that triggers a space capture scan. The operating system provides messages for the user to aim the camera in the right direction and instructs the user to move the camera to the floors or ceilings. If the room scan isn't successful, the operating system provides an error message.



AR SpaceCapture Example

The component's HTML template is minimal and includes a button that initiates the room scan.

```
<template>
  <div style="height: 100%; padding: 0px;margin: 0px;">
    <table class="rootTable" style="width: 100%; height: 100%; padding: 0px; border-spacing:
 5px">
      <thead>
        <tr>
          <th colspan="3">
            <h1>AR SpaceCapture Demo</h1>
          </th>
        </tr>
      </thead>
      <tbody>
        <tr style="height: 1px;">
         <td><input type="button" class="lightningButton" onclick={handleBeginScanRoomClick}
 value='Scan Room' style="width: 100%; height: 50px; border: none; color: white; background:
```

```
 #0072d9; border-radius: 6px; font-size: medium; white-space: normal; word-wrap:
break-word;"/></td>
        </tr>
        <tr>
          <td colspan="3">
           <div lwc:ref="previewDivSummary" class="previewDivSummary" style="width: 100%;
 height: 100%; border: 1px solid #c3c3c3; border-radius: 6px;">
              Summary: <br>
              <div lwc:ref="outputDivSummary" id="outputDivSummary" style="margin:
10px;">Results will be shown here soon...</div>
            </div>
          </td>
        </tr>
        <tr>
          <td colspan="3">
           <div lwc:ref="previewDiv" class="previewDiv" style="width: 100%; height: 100%;
 border: 1px solid #c3c3c3; border-radius: 6px;">
              Full JSON: <br>
              <div lwc:ref="outputDiv" id="outputDiv" style="margin: 10px;">Results will
be shown here soon...</div>
            </div>
          </td>
        </tr>
      </tbody>
    </table>
  </div>
</template>
```

This example uses AR SpaceCapture to let the user scan a room.

```
import { LightningElement } from 'lwc';
import { ShowToastEvent } from 'lightning/platformShowToastEvent';
import { getARSpaceCapture } from 'lightning/mobileCapabilities';


export default class ArSpaceCaptureNew extends LightningElement {
 myARSpaceCaptureScanner;
 scanRoomDisabled = false;
 capturedRoomsData = '';


 // When the component is initialized, determine whether to enable the Scan Room button
 connectedCallback() {
  this.myARSpaceCaptureScanner = getARSpaceCapture();
  if (this.myARSpaceCaptureScanner?.isAvailable() != true) {
   this.scanRoomDisabled = true;
  }
 }


 handleBeginScanRoomClick() {
  // Reset capturedRoomsData to empty string before starting a new scan
  this.capturedRoomsData = '';
```

```
// Make sure AR SpaceCapture is available before trying to use it.
// Scan Room button also disabled when scanner unavailable
if (this.myARSpaceCaptureScanner?.isAvailable()) {
 let options = {};


 // Starting the scanning process
 this.myARSpaceCaptureScanner.scanRoom(options)
  .then((results) => {
   try {
    this.refs.outputDiv.innerHTML = "";
   } catch (ex) {


   }


   const capturedRooms = results.capturedRooms ?? [];
   if (capturedRooms) {
    // Array of Rooms
    let summary = {};


    try {
     summary.isSuccess = results.isSuccess;
     summary.wallsCount = results.capturedRooms[0].walls.length;
     summary.floorsCount = results.capturedRooms[0].floors.length;
     summary.openingsCount = results.capturedRooms[0].openings.length;
     summary.doorsCount = results.capturedRooms[0].doors.length;
     summary.windowsCount = results.capturedRooms[0].windows.length;


     if (results.capturedRooms[0].floors.length > 0) {
      summary.roomSizeWidth = results.capturedRooms[0].floors[0].dimensions[0];
      summary.roomSizeHeight = results.capturedRooms[0].floors[0].dimensions[1];
      summary.roomVolume = summary.roomSizeWidth * summary.roomSizeHeight;
     }


     if (results.capturedRooms[0].openings.length > 0) {
      summary.openingSizeWidth = results.capturedRooms[0].openings[0].dimensions[0];
      summary.openingSizeHeight = results.capturedRooms[0].openings[0].dimensions[1];
      summary.openingVolume = summary.openingSizeWidth * summary.openingSizeHeight;
     }


     if (results.capturedRooms[0].doors.length > 0) {
      summary.doorSizeWidth = results.capturedRooms[0].doors[0].dimensions[0];
      summary.doorSizeHeight = results.capturedRooms[0].doors[0].dimensions[1];
      summary.openingVolume = summary.doorSizeWidth * summary.doorSizeHeight;
     }


     this.refs.outputDivSummary.innerHTML = "<pre><code>" + JSON.stringify(summary,
```

```
undefined, 2) + "</code></pre>";
       this.refs.outputDiv.innerHTML = "<pre><code>" + JSON.stringify(results, undefined,
 2) + "</code></pre>";
      } catch (ex) {
       this.refs.outputDivSummary.innerHTML = ex.code + "<br>" + ex.message;
      }
     } else {
// Single Room


     try {
       let summary = {};


       summary.isSuccess = results.isSuccess;
       summary.wallsCount = results.capturedRoom.walls.length;
       summary.floorsCount = results.capturedRoom.floors.length;
       summary.openingsCount = results.capturedRoom.openings.length;
       summary.doorsCount = results.capturedRoom.doors.length;
       summary.windowsCount = results.capturedRoom.windows.length;


       if (results.capturedRoom.floors.length > 0) {
        summary.roomSizeWidth = results.capturedRoom.floors[0].dimensions[0];
        summary.roomSizeHeight = results.capturedRoom.floors[0].dimensions[1];
        summary.roomVolume = summary.roomSizeWidth * summary.roomSizeHeight;
       }




       if (results.capturedRoom.openings.length > 0) {
        summary.openingSizeWidth = results.capturedRoom.openings[0].dimensions[0];
        summary.openingSizeHeight = results.capturedRoom.openings[0].dimensions[1];
        summary.openingVolume = summary.openingSizeWidth * summary.openingSizeHeight;
       }


       if (results.capturedRoom.doors.length > 0) {
        summary.doorSizeWidth = results.capturedRoom.doors[0].dimensions[0];
        summary.doorSizeHeight = results.capturedRoom.doors[0].dimensions[1];
        summary.doorVolume = summary.doorSizeWidth * summary.doorSizeHeight;
       }


       this.refs.outputDivSummary.innerHTML = "<pre><code>" + JSON.stringify(summary,
undefined, 2) + "</code></pre>";
       this.refs.outputDiv.innerHTML = "<pre><code>" + JSON.stringify(results, undefined,
 2) + "</code></pre>";
      } catch (ex) {
       this.refs.outputDivSummary.innerHTML = ex.code + "<br>" + ex.message;
      }
     }
    })
```

```
    .catch((error) => {
     // There was an error while scanning
     this.refs.outputDivSummary.innerHTML = error.code + "<br>" + error.message;
    })
    .finally({
     // Close capture process regardless of whether we completed successfully or had an
error
     // this.myARSpaceCaptureScanner.dismiss();
    });
  } else {
   this.refs.outputDivSummary.innerHTML = 'AR SpaceCapture is not available on your device.';

  }
 }
}
```

AR SpaceCapture API

Access a device's camera and AR capabilities from an LWC.

```
function scanRoom(options) {
 return new Promise((resolve, reject) => {
  // Implement the logic to scan the room using the provided options
  // and return the ARSpaceCaptureResult
  // When the scan is complete, resolve the promise with the result
  resolve({
   // Populate the ARSpaceCaptureResult object with the appropriate data
  });
 });
}
```

INDEX

C

Classes
- FSL.AdvancedGapMatrix 116
- FSL.AppointmentBookingService 117
- FSL.AppointmentBookingSlot 127
- FSL.AppointmentInsightsResult 131
- FSL.GradeSlotsService 128
- FSL.OAAS 137
- FSL.OAASRequest 146
- FSL.PolygonUtils 150
- FSL.RecurringAppointmentsManager 152, 157
- FSL.RecurringPattern 155
- FSL.ResourceScheduleData 158
- FSL.ScheduleJobsApi 159
- FSL.ScheduleJobsApi.GroupData 135
- FSL.ScheduleResult 161
- FSL.ScheduleService 164
- FSL.SchedulingOption 169
- FSL.TimeInterval 170

Connect in Apex 111
ConnectAPI 111

F

Field Service Flow API 47
FieldServiceMobileSettings 52
FSL
- namespace 114

FSL.AdvancedGapMatrix
- classes 116

FSL.AppointmentBookingService
- classes 117

FSL.AppointmentBookingSlot
- classes 127

FSL.AppointmentInsightsResult
- classes 131

FSL.GradeSlotsService
- classes 128

FSL.OAAS
- classes 137

FSL.OAASRequest
- classes 146

FSL.PolygonUtils
- classes 150

FSL.RecurringAppointmentsManager
- classes 152, 157

FSL.RecurringPattern
- classes 155

FSL.ResourceScheduleData
- classes 158

FSL.ScheduleJobsApi
- classes 159

FSL.ScheduleJobsApi.GroupData
- classes 135

FSL.ScheduleResult
- classes 161

FSL.ScheduleService
- classes 164

FSL.SchedulingOption
- classes 169

FSL.TimeInterval
- classes 170

N

Namespaces
- FSL 114

S

ServiceReportTemplate 55