Prodly™

eBook

The essential guide to

Reference Data Management

Salesforce Field Service Edition

This is no ordinary guide. It's a guide for achieving Salesforce Field Service reference data nirvana. It's a mouthful, we know. But this guide will awaken you to the fact that the pain you're experiencing (or about to experience) when moving the reference data that underpins low-code applications is not a natural law. This guide will show that this pain can be overcome. Nay, it can be eliminated.

In the pages to follow, we will embark on a journey. We will examine the strategic and technical challenges of moving configuration data for Salesforce Field Service, although our journey applies to other configuration data-based applications such as CPQ, Billing, Advanced Approvals, B2B Commerce, and other low-code apps developed by Salesforce ISVs.

We will also assess the impact that these challenges have on the broader business—an important topic too often ignored. And we will show you an easier, automated way to manage changes to the reference data that makes up low-code applications.

Let's get going!

Reference data defined

In this guide, we will be talking a lot about Salesforce reference data. We want to take a moment to explain what this term means.

Reference data, sometimes called configuration data, “define the set of permissible values to be used by other data fields. In Field Service, reference data include Address, Location, Map Polygon, Operating Hours, Product Required (Work Type), Resource Reference, Scheduling Policy, Scheduling Policy Objective, Scheduling Policy Work Rule, Service Crew, Service Crew Member, Service Objective, Service Resource, Service Resource Skills, Service Territory, Service Territory Location, Service Territory Member, Skill, Skill Requirement, Time Slot, User, User Territory, Work Rule, and Work Type.

To be clear, reference data are not metadata. Unlike metadata which is stored as code, reference data are records and stored in relational data tables. This means reference data is not accessible by Salesforce Change Sets. When companies move Field Service reference data between Salesforce orgs, they must take specific actions to move them separately from any metadata changes.

Here’s a quick view of how reference data differs from other types of data from a Salesforce Field Service perspective:

Meta	**Reference**	**Master**	**Transactional**
Objects	Address		Account
Fields	Location		Case
Apex Code (e.g., triggers)	Map Polygon		Contract
Validation Rules	Operating Hours		Opportunity
	Product Required		Order
	Scheduling Policy		Order Product
	Service Crew		Quote
	Service Territory		
	Skill		
	Skill Requirement		
	Time Slot		
	User Territory		
	Work Rule		
	Work Type		

The cost of maintaining Field Service

Early on, things are great. Your company invests precious resources in customizing its Salesforce environment, gradually installing managed applications, such as Field Service, and driving adoption. Processes are established. You wisely consolidate critical information onto a single platform and streamline business workflows around it.

Salesforce becomes a critical growth driver for your company—a single source of truth everyone relies on. Everyone's happy.

But this growth comes with a price: As your team is adding managed packages that are configured with clicks not code, the reference data invariably becomes unreliable and outdated.

Possible symptoms that show up in these implementations include:

- Missing reference records: e.g. A service territory may lack some or all of its assigned service territory members. As a result, a dispatcher may not be able to respond to some service requests in the service territory.
- Duplicate reference records: e.g. Multiple scheduling policies for the same territory and time slot can trip up the service scheduling optimizer, resulting in errors when assigning service resources to service requests.
- Errors in reference records: e.g. Dispatchers may be relying on outdated resource skills, dispatching the wrong resources to some service appointments.

Oh, where are those glory days of yesteryear?

The full business impact of Salesforce Field Service

How do you gauge the return on your investment in Salesforce managed applications? Is it by examining how consistently your services team uses them? Or maybe how much the team complains about them?

While it's important to study adoption and usage patterns of Salesforce managed packages, we must broaden our horizon of investigation and understand the impact these applications have on the business at large.

When implementations of Salesforce Field Service deteriorate, or take too long to keep in excellent working order, businesses may experience the following consequences:

- Lower customer satisfaction, as a result of poor service due to insufficient service crew skills
- Lower revenue, as a result of incorrect scheduling policies that limit the number of appointments serviced
- Missed sales opportunities due to delays in introducing new territories and products (work types)

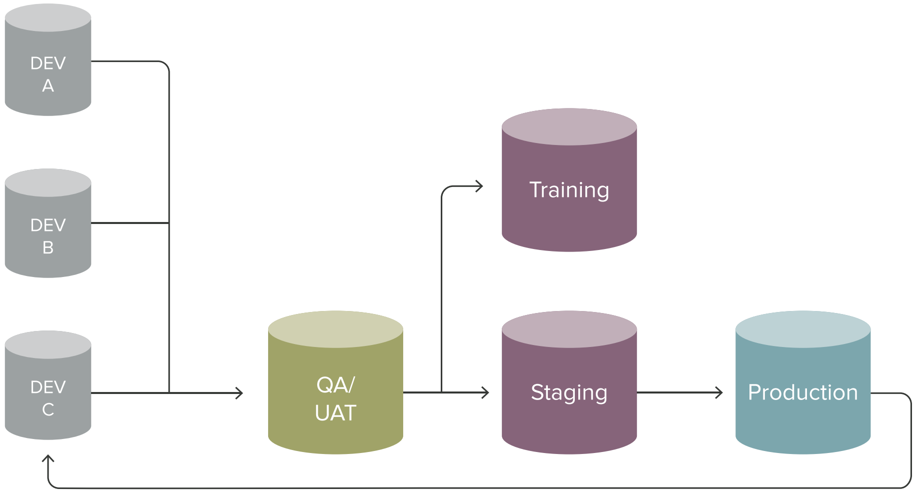
To quantify the drag that improperly configured Field Service can have on your business, use a calculator like the one shown below. Once you understand the impact, you can dig deeper into the causes of Field Service mis-configuration.

IT/Field Ops Hours Saved	W/O Prodly	With Prodly	**Comments**
Updates Per Year (Sprints X Updates per Sprint)	78	78	The laborious process of updating Field Service or other data represents a waste of specialized talent. It is also a source of job dissatisfaction and job burnout for your most experienced people. With Prodly, updating Field Service data is easier and can be accomplished by a less senior person.
Hours Per Update	20	2	
Labor Costs-Per-Hour	$100	$100	
Labor Costs	$156,000	$15,600	

The challenge of applying agile to Field Service

How do Salesforce Field Service implementations deteriorate? Much of it has to do with how their reference data move across Salesforce organizations.

Smart companies follow an agile development process that moves reference data across separate Salesforce orgs for development, testing, training, and release. This helps to minimize errors and to ensure that Service reps are using accurate, reliable, and trustworthy information.



Agile reference data development process

But it's in these reference data moves that things go awry. Often, data deployments fail or silently introduce errors that go undetected. Errors are primarily introduced from inadequate testing (the Q/A stage in the image above) prior to moving data from a sandbox to production. Lack of separation of development and testing environments is one reason for failed testing. In this scenario, dev pushes new or updated configuration data records after testing is already done.

Another reason is failing to have sufficient time to move reference data or sample test records between orgs, so the team ends up cutting corners and works in a single sandbox. This is problematic because it is difficult to keep track of the status of the work performed by different admins and developers. Version control is difficult because development and testing is done in the same org.

A third reason for insufficient testing is lack of time for QA and UAT. Often, it takes so long to deploy data between orgs that there is not enough time left in the schedule to do testing.

The revenue impact of bad Field Service deployments

Earlier, we examined the drag that an improperly implemented Field Service application could have on business performance. We also looked at how insufficient testing fails to capture reference data problems that ultimately make their way into the production environment.

Let's look at three scenarios where Field Service implementations go astray. Each example includes a reference data issue that works its way into production, a resulting Field Service performance issue, and the subsequent business impact.

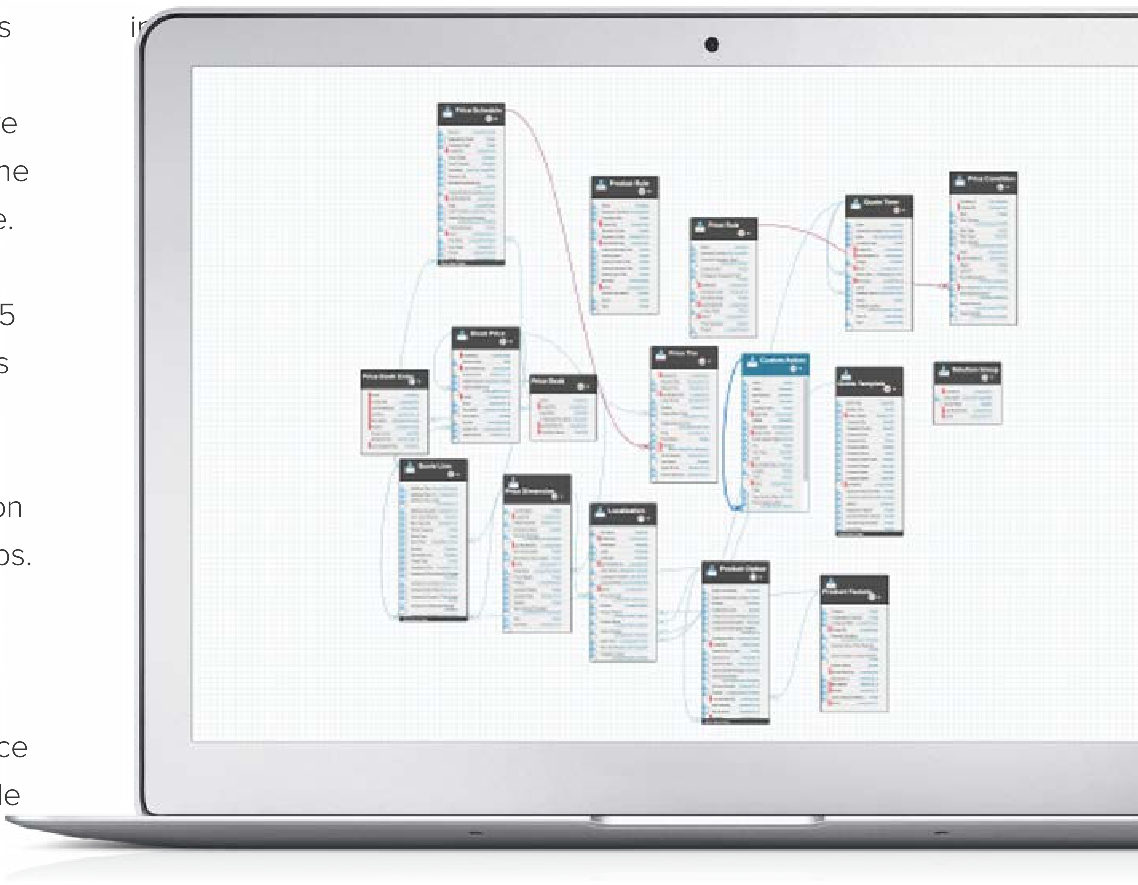
A reference data issue impacting production…	…resulting in Field Service configuration issue…	…which lands the business in a swampy mess
Internal Salesforce IDs are not captured, or some source records aren't moved.	Field Service contains multiple scheduling policies for the same territory.	Dispatchers fail to service requests in the service territory, lowering revenue.
Bad configuration isn't detected during testing.	Some areas of Field Service are not functioning properly.	Technicians with insufficient skills are assigned to service appointments, lowering customer satisfaction.
Prolonged data move process causes delays in pushing config changes.	Field Service contains outdated information.	Missed sales opportunities due to delays in introducing new territories and products (work types).

Conventional change management goes astray

The chief reason why faulty Field Service configuration makes its way into production is that conventional data deployment tools used to move data between Salesforce orgs were never designed to handle the reference data model of low-code applications like Field Service.

The current Field Service data model now includes more than 25 objects that would take a seasoned architect more than 10 hours to move between Salesforce orgs. The Product object alone has 26 related objects that customers typically need to include in their configuration data deployments and remap all of their parent-child relationships. It also has self-referencing fields to take into consideration.

The Field Service data schema incorporates complex relational data. It requires a reference data deployment between Salesforce orgs that maintains the parent-child relationship between multiple objects, multiple levels, multiple relationships per object, and self-referencing records. This image conveys some of the objects in the Field Service data model and their



Using Data Loader: The changing ID problem

Conventional data loaders, which move data for only one object at a time, require sequential data moves, or multiple passes, to remap record IDs across organizations and push out the data. This is an error-prone process that takes a long time to execute.

A major roadblock here is the simple fact that Salesforce record IDs are unique and assigned at creation time. When you move a data record from one Salesforce org to another, the record is assigned a brand new ID. It is, therefore, impossible to base record relationships on source org IDs—those IDs are replaced by other IDs in the destination org.

There is one exception: standard Salesforce objects. When a Salesforce sandbox is created, the records in these objects are copied from production along with the metadata. The record IDs for standard objects are actually the same as production. However, after the sandbox is created, any new records added to either org will have unique IDs and will require remapping when copying them to another org.

Check out the Prodly blog to learn more about the ins and outs of Salesforce record IDs.

One step at a time takes a really long time

The following steps show the process of manual configuration data moves. Recall that this sequence must be followed for any group of objects out of the 25+ Field Service reference objects that you wish to move any time you change your source org reference data. You will move these objects sequentially, one object at a time.

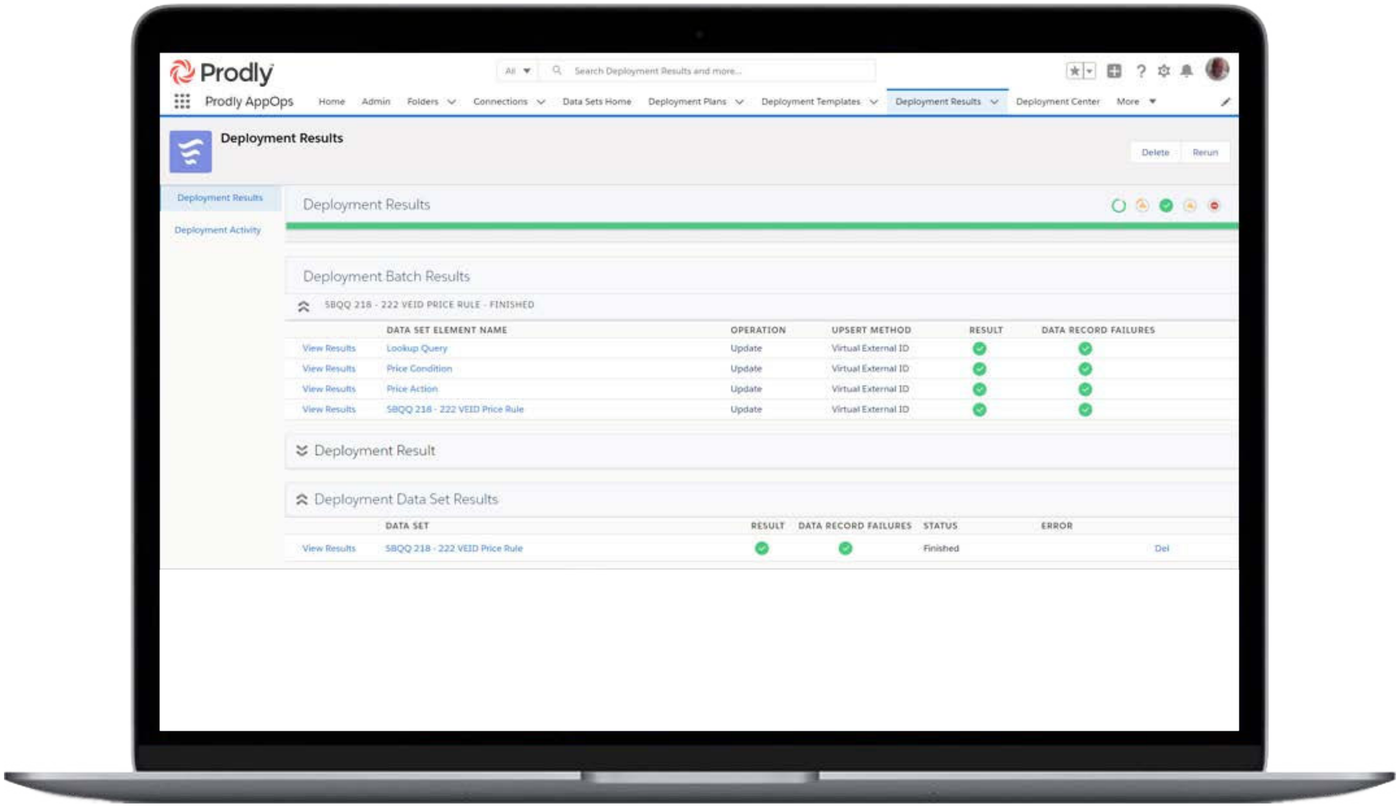
Assuming you don’t have Salesforce external IDs defined, the sequential steps are:

1. Export to a spreadsheet the parent object records from the source org with the source record IDs.
2. Manually search for and remove any existing duplicate parent records.
3. Use the data loader to import these records into the destination org.
4. Export parent records with their new and old (source) record IDs from the destination.
5. Export child object records with their parent lookup IDs from the source org to a spreadsheet.
6. Manually search for and remove any existing duplicate child records.
7. Map parent lookup IDs for children from the source to the new record IDs for the parents using a spreadsheet VLOOKUP formula. This can be quite involved when you have multiple parent lookups for a child record.
8. Use the data loader to import the child object records with their new lookup IDs into the destination org.

Automate the full Field Service development life cycle with AppOps

Repeatedly moving complex Field Service objects using conventional data deployment tools is simply too arduous a journey to take. This mind-numbing, labor-intensive, and error-prone task calls for automation.

Meet Prodly AppOps, a platform to completely automate the full life cycle of low-code development. AppOps Release makes building and iterating Salesforce applications easy and automates the manual process of transferring reference data between Salesforce orgs. With AppOps Release, you can maximize your return on investment in Salesforce Field Service Lightning, CPQ, Billing, B2B Commerce, and other reference data-driven native Salesforce applications by ensuring they always use accurate and trustworthy configuration data.



Deployment Results

Deployment Results

Deployment Batch Results

SBQQ 218 - 222 VEID PRICE RULE - FINISHED

	DATA SET ELEMENT NAME	OPERATION	UPSERT METHOD	RESULT	DATA RECORD FAILURES
View Results	Lookup Query	Update	Virtual External ID	✓	✓
View Results	Price Condition	Update	Virtual External ID	✓	✓
View Results	Price Action	Update	Virtual External ID	✓	✓
View Results	SBQQ 218 - 222 VEID Price Rule	Update	Virtual External ID	✓	✓

Deployment Result

Deployment Data Set Results

	DATA SET	RESULT	DATA RECORD FAILURES	STATUS	ERROR	
View Results	SBQQ 218 - 222 VEID Price Rule	✓	✓	Finished		Del

AppOps Release is a Salesforce reference data deployment solution that is scalable, reliable, and repeatable.

Learn more about AppOps

"Prodly is such a great tool for data moves between orgs. We have two Salesforce instances, and both orgs require a huge amount of set up data to support an end-to end-testing. Now with this great tool we cut down our deployment time from days to hours, and once the initial data set creation is done it just comes down to minutes."

Kirthi Sidulwar
CRM Application Architect at K12

"I was completely new to Salesforce Field Service, and Prodly really helped me get up to speed faster. Prodly's pre-built Field Service templates made all of the data schema decisions for me. I didn't need to worry about maintaining the different object relationships; Prodly automatically sequences the data deployment in the correct order for me."

Zachary Hower
Salesforce System Administrator, Quest Software

"Prodly has been a lifesaver for me, a lone Administrator responsible for four sandbox environments and Production. The ability to manage and move data across all of these environments without having the need to worry about scripting, data loaders or duplicating data has been an invaluable time saver. In addition, the setup and configuration was extremely easy, and the support provided is second to none."

James Carey
Salesforce Administrator, Berkshire Hathaway Travel Protection

Pre-built Field Service Data Sets:

Prodly's pre-built data sets for Salesforce Field Service provide a fully tested solution for deploying Field Service reference data between Salesforce orgs. Use our pre-made templates to auto-select the correct objects, fields, and relationships for deployments. The 8 data sets cover all 25+ Salesforce Field Service reference data objects. They are also modularized so that you can update specific aspects of your Field Service reference data.

Or, you can use a Deployment Plan to run multiple data sets in the correct sequential order.

Template File	Data Set Elements (**bold** indicates the root element)
FS Location	Address, **Location**, Service Resource, Service Territory, Service Territory Location
FS Permission Set	**Permission Set**, Permission Set Assignment, User
FS Scheduling Policy	**Scheduling Policy**, Scheduling Policy Objective, Scheduling Policy Work Rule, Service Objective, Work Rule
FS Service Crew	**Service Crew**, Service Crew Member, Service Resource
FS Service Resource	**Service Resource**, User
FS Service Territory	Map Polygon, Operating Hours, Service Resource, **Service Territory**, Service Territory Member, Time Slot, User, User Territory
FS Skill	Service Resource, Service Resource Skill, **Skill**
FS Work Type	Product, Product Required, Skill, Skill Requirement, **Work Type**

Pre-deployment Checklist

Whether you use Prodly AppOps Release or a data loader and spreadsheets, you can use this pre-deployment checklist to increase your success in implementing Field Service.

1 Ensure you have the same major version of the Salesforce Field Service managed package installed in your source and destination organizations.

2 Assign the Field Service Admin License and Field Service Admin Permissions permission sets to the user who will create the Prodly AppOps connections to the source and destination organizations.

3 Ensure that the schemas for all objects in the deployment exactly match in the source and destination organizations.

4 Ensure that all technician, dispatcher, and agent users in the source organization also exist in the destination organization(s) with the same permission sets.

5 Clean the data in your source and destination organizations as best you can to eliminate duplicate records.

6 Determine the approach to avoiding creation of duplicate records appropriate for your use case.

We’ve reached our destination!

Wow, you’ve stuck with us throughout this guide and here we are at the end. Well done!

We hope that you enjoyed the journey, and that it was helpful to you in understanding that reference data deployments do not have to be difficult or painful or a drag on your business. Building up agile development processes for reference data has become an imperative today. We really cannot let manual reference data moves slow them down. It’s time to automate these tasks.

If you would like to learn more about Prodly AppOps and how it can help your organization email **info@prodly.co** or call us at **1-650-761-4876**.

Get a demo

About Prodly

Prodly helps companies build and continuously improve business applications faster, more reliably, and more frequently. We automate the full life cycle of low-code development, empower more non-developers to configure applications, remove bottlenecks in the development process, and provide IT governance to mitigate risk of agile development.

For more information visit prodly.co.

Prodly™