

# Salesforce CPQ Data Management 101

Written by the team that created Salesforce CPQ



eBook

Moving the type of data that underpins low-code applications like Salesforce CPQ is a painstaking, complicated, and time-consuming process. But as we're about to show you, it doesn't have to be this way. And you can take our word for it—we're the team that created Salesforce CPQ.

In the following pages, we examine the technical and strategic challenges of moving CPQ data. We assess the impact these challenges have on the broader business, and we show you an easier, automated way to manage changes to the type of data that makes up low-code applications.

***Let's get going!***

# What is Salesforce CPQ data?

The type of data that underpins Salesforce CPQ is referred to as “configuration data,” “record data,” or “reference data.” It defines the set of permissible values other data fields may use, and it [achieves a specific functionality](#) in a low-code app.

In this ebook, we discuss Salesforce CPQ data, but other applications developed by Salesforce ISVs—such as Billing, Advanced Approvals, Field Service, and B2B Commerce—also use configuration data.

To be clear, configuration data is **not** metadata. Unlike metadata, which is stored as code, configuration data consists of records that are stored in relational data tables.

Here’s a quick overview of how configuration data differs from metadata in Salesforce CPQ:

## Configuration data:

- Custom action, block price, localization, price book, price book entry, price dimension, price rule, product, product feature, product option, product rule, quote template, quote term, solution group
- Records that are stored in relational data tables

## Metadata:

- Objects, fields, Apex code (e.g. triggers), validation rules
- Stored as code

Because configuration data isn’t metadata, it isn’t accessible by change sets. That means that when you want to move CPQ data between orgs, you have to take specific actions to move it *separately from any metadata changes*.

# The unspoken cost of Salesforce CPQ maintenance

When you first implement Salesforce, you invest considerably in customizing your environment. You gradually install managed applications—such as CPQ—and drive adoption throughout the company. You establish processes—plus, you consolidate essential information onto a single platform and streamline business workflows around it.

As a result, Salesforce becomes a single source of truth that everyone relies on, as well as a critical growth driver for your business.

But this growth comes at a cost. As you add managed packages that are configured with clicks, not code, the configuration data they're made up of invariably becomes unreliable and outdated—unless you implement an active process to combat [technical debt](#).

For example, you might notice:

- **Missing reference records**—If a Price Book doesn't contain recently released products, your sales reps won't have access to a new Product or bundle.
- **Duplicate reference records**—Sales reps see “duplicate field” errors in the CPQ quote line editor when they configure bundles.
- **Errors in reference records**—Sales reps might use an old Quote Template that contains outdated terms and conditions.

All of these examples cause friction and slow the pace of business down.

# The full business impact of Salesforce CPQ

How do you estimate the ROI on your investment in Salesforce CPQ? Is it by examining how consistently your sales team uses it? Or perhaps by how much the team complains about it?

Of course, you need to study the adoption and usage patterns of CPQ. On top of that, you should broaden your investigation and examine its impact on the business at large.

Consider this: When your Salesforce CPQ implementation deteriorates or takes too long to keep in excellent working order, you might experience the following consequences:

- Lower customer satisfaction resulting from misconfigured sales quotes
- Lower revenue due to mispriced products or excessive discounting
- Missed sales opportunities caused by delays in introducing new products and promotions

To quantify the drag that improperly configured CPQ 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 Salesforce CPQ misconfiguration.

## Cost of lost bookings due to quoting errors

	%	\$	Comments
<b>Annual bookings</b>	100%	\$50,000,000	Quoting errors and delayed implementation of changes to CPQ have a major impact on bookings. Prodlly dramatically improves the accuracy and timeliness of CPQ product catalog updates.
<b>Quoting error rate</b>	1%	\$500,000	
<b>Lost opportunities from uncompetitive quotes</b> (slow CPQ updates)	1%	\$500,000	
<b>Total lost bookings</b>	<b>2%</b>	<b>\$1,000,000</b>	

TABLE 1

# Apply DevOps best practices to Salesforce CPQ change delivery

How do Salesforce CPQ implementations deteriorate? A lot of it has to do with how you move configuration data across Salesforce orgs.

When you make changes in Salesforce, it's best practice to use a DevOps methodology. DevOps involves moving (or migrating) configuration changes through a release pipeline with separate environments for development, integration, testing, training, and release. This helps minimize errors and ensures your sales reps are using accurate, reliable, and current information.

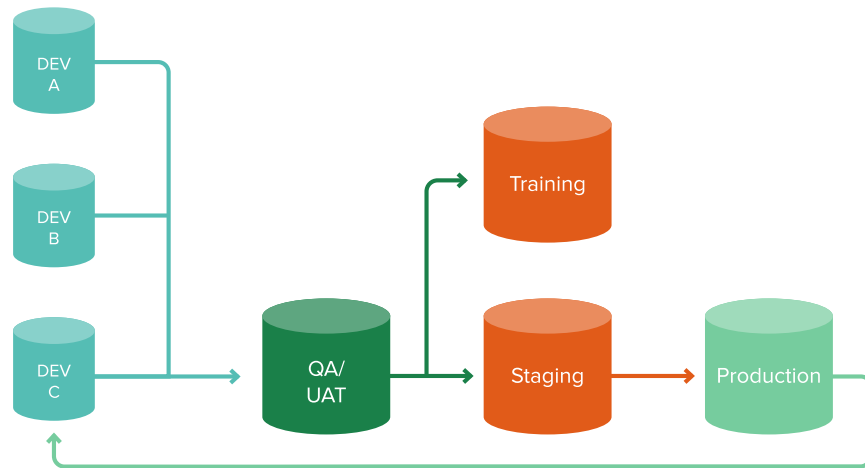


Image 1: The DevOps change delivery process

Unfortunately, things can easily go awry during the deployment process. A deployment can fail or silently introduce a bug you don't detect. The main cause of bugs is inadequate testing in the QA stage prior to moving data from a sandbox to production.

Insufficient testing often results from a failure to separate your development and testing environments. In this scenario, you promote new or updated configuration data records after you've already performed testing.

Another issue can be that you simply don't have enough time to migrate sample test records between orgs—so you wind up cutting corners... with everyone working in a single sandbox.

This is a problem because with a shared sandbox, it's difficult to keep track of the status of everyone's work. What's more: Version control becomes super challenging because development and testing takes place in the same org.

Last, but certainly not least, subpar testing can result from a lack of time for QA and UAT. Oftentimes, it takes so long to deploy data between orgs that you don't have enough time left in the schedule to perform the necessary testing.

# The revenue impact of Salesforce CPQ mistakes in production

In table 1, we saw an example of how an improperly implemented or maintained Salesforce CPQ application can negatively affect revenue. And in the previous section, we discussed why—if you don't test thoroughly—configuration data problems make their way into your production environment and cause bugs. But what does this look like in practice?

Let's examine three scenarios where Salesforce CPQ updates go wrong. Each example includes:

- A configuration data problem that works its way into production
- A resulting CPQ performance issue
- A subsequent business impact

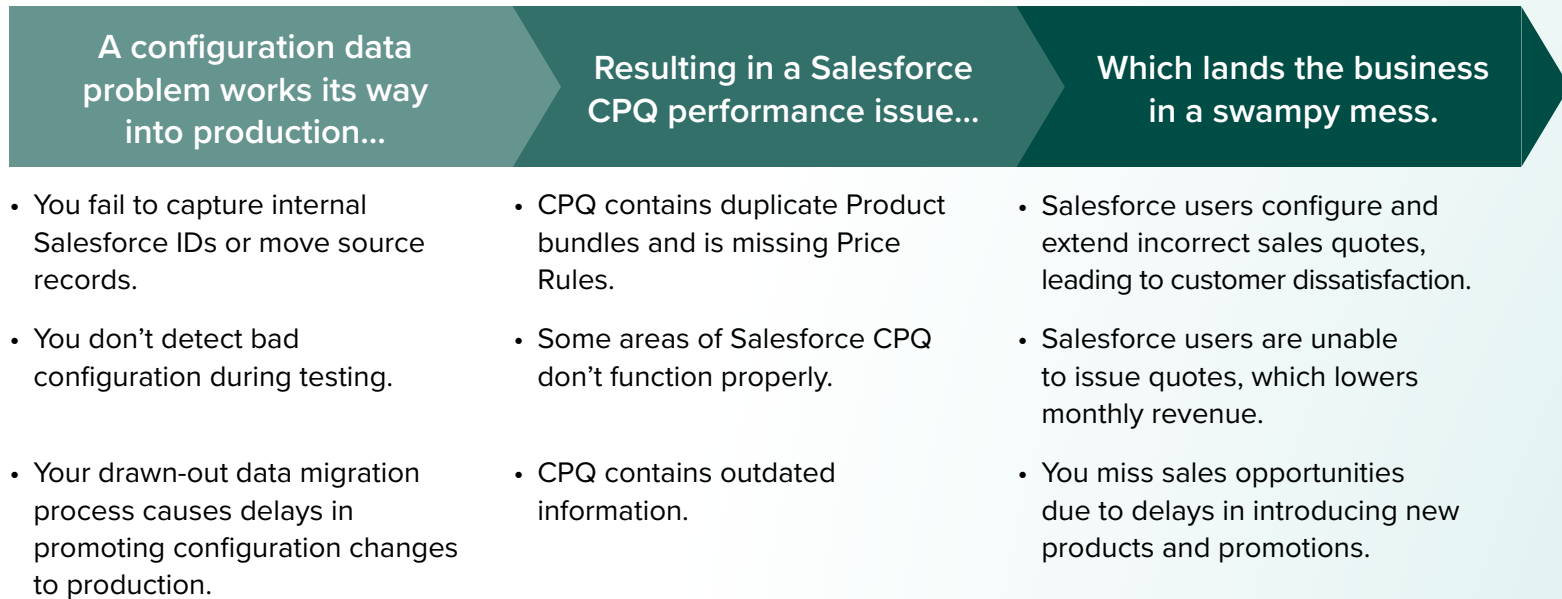


TABLE 2



## Legacy tools can't handle configuration data

As we've seen, inadequate testing can cause you to deploy faulty CPQ data to production. However, there's another—very important—reason that contributes more directly. Legacy tools for deploying data—like Data Loader—aren't designed for Salesforce CPQ or other applications that rely on configuration data instead of metadata.

Think about it: The current CPQ model includes over 50 objects that would take a seasoned architect over 20 hours to move between Salesforce orgs. In fact, the Product object alone has 26 related objects you typically need to include in your configuration data deployments. This involves remapping every parent-child relationship—all while taking self-referencing fields into consideration.

The Salesforce CPQ data schema incorporates complex relational data. That's why it requires a configuration data deployment that maintains the parent-child relationship between:

- Multiple objects
- Multiple levels
- Multiple relationships per object
- Self-referencing records

It's easy to understand how these interdependencies complicate the process of migrating CPQ data between orgs tremendously.



# Changing IDs make Data Loader challenging

Conventional data loaders only move data for one object at a time. So they need multiple consecutive data moves to remap record IDs across orgs and push out the data. This process is tedious, prone to error, and time consuming.

One major roadblock is the simple fact that when you create a record, a unique Salesforce record ID is assigned to it. However, when you move that record from one org to another, Salesforce assigns it a brand-new ID.

That means it's impossible to base record relationships on source org IDs because in the destination org, they're replaced with new IDs.

There's one exception: Salesforce considers Product, Price Book, and Price Book Entry—while configuration data objects—to be standard objects. When you create a new sandbox, the records in Products, Price Books, and Price Book Entries are copied from production along with the metadata. And the record IDs for these three configuration data objects remain the same as in production.

However, once you've created the new sandbox, any new records you add to either production or the sandbox have unique IDs. You also have to remap them if you want to copy them to another org.

[Learn more about when Salesforce record IDs change.](#)

# How to manually migrate Salesforce CPQ data

The following steps describe the process of manually migrating configuration data in Salesforce CPQ. Remember: Whenever you change your source org configuration data, you have to follow this sequence for each group of objects—out of the more than 50 Salesforce CPQ objects—that you want to move.

Assuming you haven't defined external IDs, the steps are:

1. Export the parent object records with the source record IDs from the source org to a spreadsheet.
2. Manually search for any existing duplicate parent records and remove them.
3. Use Data Loader to import these records into the destination org.
4. Export the parent records with their source record IDs, as well as their new record IDs, from the destination.
5. Export the child object records with their parent lookup IDs from the source org to a spreadsheet.
6. Manually search for any existing duplicate child records and remove them.
7. Using a spreadsheet VLOOKUP formula, map the parent lookup IDs for the children from the source to the new parent record IDs. If you have multiple parent lookups for a child record, this can be quite complicated.
8. Use the data loader to import the child object records—along with their new lookup IDs— into the destination org.


This is clearly too time consuming. (Not to mention headache-inducing.)

# Automate the full Salesforce CPQ development cycle with Prodlly DevOps

By now, it's obvious that repeatedly moving CPQ objects with [conventional data deployment tools](#) is extremely inefficient.

That's why we automated this mind-numbing, labor-intensive, and error-prone process. Prodlly DevOps is an all-in-one DevOps platform for Salesforce CPQ. It allows you to accelerate innovation in your pricing and packaging strategy with minimal risk of disrupting mission-critical revenue processes.

Here's an example of how much you can save using Prodlly DevOps:

	Without Prodlly	 With Prodlly
<b>Updates per year</b> (sprints x updates per sprint)	78	78
<b>Hours per update</b>	20	2
<b>Labor costs per hour</b>	\$100	\$100
<b>Total labor costs</b>	<b>\$156,000</b>	<b>\$15,600</b>

**Savings: \$140,400**

TABLE 3

## Comments

The labor-intensive process involved with manually updating CPQ or other data represents a waste of specialized talent. It's also a source of job dissatisfaction and burnout for your most experienced people. With Prodlly, updating CPQ is significantly easier and can be accomplished by a less senior person.

## In another example, a company with:

- A Salesforce team of three employees
- Annual revenue of \$100,000,000
- 12 releases per year
- An average of four hours to recover from failure

## Would achieve the following cost savings with Prodlly:

- Reduced TCO due to time savings: \$51,000
- Reduced risk due to less downtime: \$187,500
- Increased revenue due to enhanced agility: \$755,208

# Prodly's pre-built CPQ data sets

Our pre-built data sets or templates for Salesforce CPQ provide a fully tested solution for deploying CPQ configuration data between orgs. Use them to autoselect the correct objects, fields, and relationships for deployments. The 16 data sets cover all 50+ CPQ configuration data objects. They're also modularized so you can update specific aspects of the configuration data. Another option is to use a deployment plan to run multiple data sets in the correct sequential order.

Template file	Data set elements ( <b>bold</b> indicates the root element)
SBQQ custom action data set template.json	Custom action condition, <b>custom action</b> , search filter
SBQQ custom script data set template.json	<b>Custom script</b>
SBQQ discount category data set template.json	<b>Discount category</b>
SBQQ import format data set template.json	Import column, <b>import format</b>
SBQQ localization data set template.json	Line column, <b>localization</b> , price dimension, product, product feature, product option, quote template, quote term, search index, template content
SBQQ lookup data data set template.json	<b>Lookup data</b>
SBQQ price book data set template.json	<b>Price book</b> , price book entry, product
SBQQ price rule data set template.json	Lookup query, price action, price condition, <b>price rule</b> , summary variable
SBQQ product data set template.json	Additional document attribute item, attribute set, block price, configuration attribute, configuration rule, cost, discount category, discount schedule, discount tier, error condition, lookup query, option constraint, price action, price book, price condition, price dimension, price rule, product action, product attribute, product attribute, <b>product</b> , product feature, product option, product rule, upgrade source, summary variable
SBQQ product rule data set template.json	Configuration rule , error condition, product action, product, product feature, <b>product rule</b> , summary variable
SBQQ quote process data set template.json	Process input condition, process input, <b>quote process</b>
SBQQ quote template data set template.json	Additional document, line column, <b>quote template</b> , template content, template section
SBQQ quote term data set template.json	<b>Quote term</b> , summary variable, template content, term condition
SBQQ solution group data set template.json	<b>Solution group</b>
SBQQ theme data set template.json	<b>Theme</b>

# Secure, fast, and error-free

Prodly DevOps automates work management, release management, regression testing, versioning, and compliance—increasing the productivity of your Salesforce team by 80 percent.

*“Prodly is a great tool for CPQ projects. When configuration data is the life of your development, yet you want to commit code and metadata daily, how do you automate the finicky configuration data? One answer: Prodly!”*

**Michael Marsh**

Salesforce Product Development  
Manager, Johnson and Johnson

*“Prodly is such a great tool for data moves between orgs. We have two Salesforce instances, and both orgs require a huge amount of setup data to support 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

*“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 these environments without having 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 Protectionn

# Pre-deployment checklist

Whether you use Prodlly DevOps or a data loader and spreadsheets, use this pre-deployment checklist to successfully deploy CPQ data.

- ✓ Ensure you have the same major version of the Salesforce CPQ managed package installed in both your source and destination orgs.
- ✓ Check that the schemas for all objects in the deployment exactly match in the source and destination orgs.
- ✓ Make sure that the CPQ global picklist exactly matches in the source and destination orgs.
- ✓ [Clean the data](#) in your source and destination orgs to eliminate duplicate records.
- ✓ If you're using a production org for your source or destination, ensure there's a user with a CPQ license in your production org.
- ✓ In the production org or sandbox, assign the CPQ admin permission set to the connection user.
- ✓ If you use multi-currency in your source org, make sure multi-currency is turned on in your destination org. The currency ISO codes must match in your source and destination orgs. Refer to the [Salesforce Enable Multiple Currencies](#) help page for details.
- ✓ In your destination org, disable triggers. Navigate to Setup > Installed Packages > Salesforce CPQ > Configure > Additional Settings. Select Triggers Disabled, and click Save.
- ✓ In your destination org, make sure the Standard Price Book is active.
- ✓ Determine the most appropriate strategy for avoiding creating duplicate records for your use case.

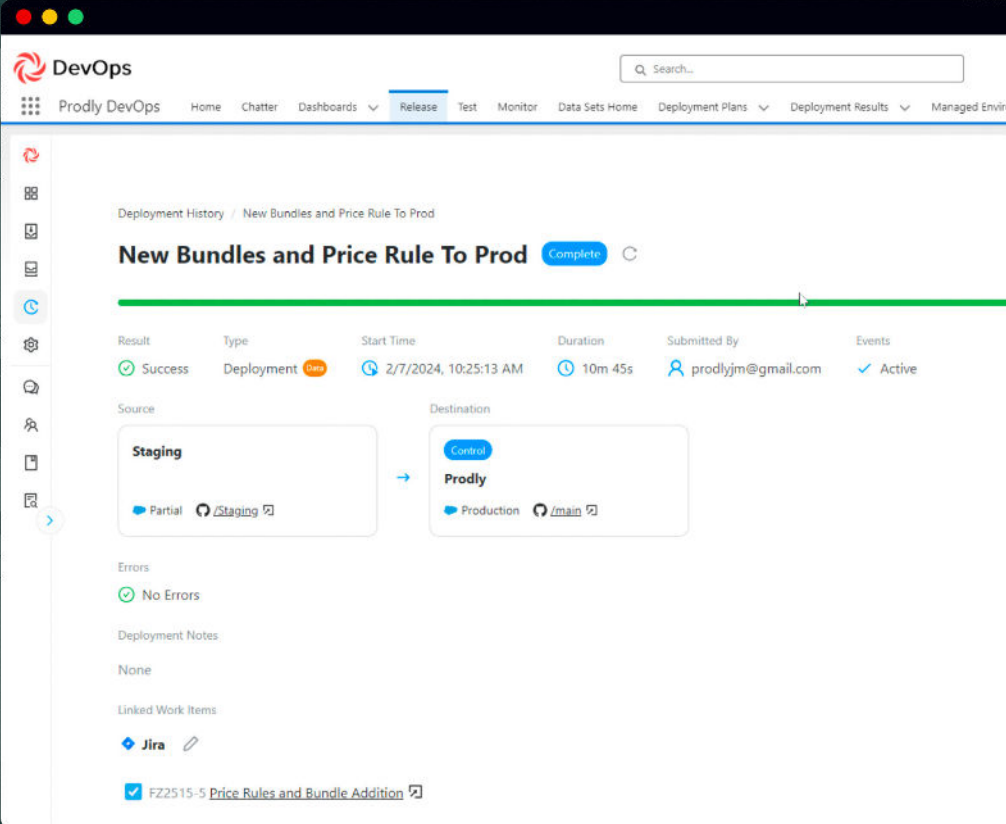
## See Prodlly DevOps in action!

We hope you've found this guide helpful—and that you now understand CPQ data deployments don't have to be challenging or negatively impact your business.

To drive digital transformation in your organization, it's critical to implement a robust DevOps process for CPQ data. You simply cannot let manual configuration data deployments slow you down—so you need to automate them.

To see just how fast and easy CPQ deployments are with Prodlly DevOps, request your personalized demo.

[Request your personalized demo](#)



The screenshot displays the Prodlly DevOps web application interface. At the top, there is a navigation bar with the Prodlly logo and the text 'DevOps'. Below this, a secondary navigation bar includes links for 'Prodly DevOps', 'Home', 'Chatter', 'Dashboards', 'Release', 'Test', 'Monitor', 'Data Sets Home', 'Deployment Plans', 'Deployment Results', and 'Managed Envir...'. A search bar is located on the right side of the top navigation bar.

The main content area shows a deployment history entry for 'New Bundles and Price Rule To Prod', which is marked as 'Complete'. Below this, a table provides details for the deployment:

Result	Type	Start Time	Duration	Submitted By	Events
Success	Deployment	2/7/2024, 10:25:13 AM	10m 45s	prodlyjm@gmail.com	Active

Below the table, the deployment process is visualized with a 'Source' box labeled 'Staging' (containing 'Partial' and 'Staging') and a 'Destination' box labeled 'Prodly' (containing 'Control' and 'Production'). An arrow points from the source to the destination.

Additional sections include 'Errors' (No Errors), 'Deployment Notes' (None), and 'Linked Work Items' (Jira). At the bottom, there is a checkbox for 'FZ2515-5 Price Rules and Bundle Addition'.