**OmniStudio DataRaptors**

So what exactly is a DataRaptor? A DataRaptor is a mapping tool that enables you to read, transform, and write Salesforce data. Think of it this way: for every digital customer interaction or business process, your system needs to extract data to display it. When the user changes that data or enters new data, it must be saved too. That’s where OmniStudio DataRaptors come in.

There are four different types of DataRaptors (and we cover each in detail shortly), but in general, DataRaptors supply data to OmniScripts and OmniStudio FlexCards from Salesforce, and write updates from OmniScripts and FlexCards back to Salesforce. They typically do this via OmniStudio Integration Procedures.

Here’s an example of the data flow.

1. **Get Data:** An OmniScript calls a DataRaptor Extract (via an Integration Procedure) to read data from Salesforce. For example, an Edit Account OmniScript needs to display data such as the account name, phone number, and website.
2. **Manipulate Data:** The OmniScript captures changed and new data based on user input. For example, an agent changes the account phone number.
3. **Save Data:** The OmniScript calls a DataRaptor Load (via an Integration Procedure) to write data back to Salesforce. For example, the updated account phone number is saved back to the Account record.

Although Apex classes can read, write, and transform data, DataRaptors offer a few more perks. DataRaptors take less time to create and are easier to maintain; that’s why we recommend using DataRaptors as a best practice.

Now let’s look at what each DataRaptor does.

| **What It Is** | **What It Does** |
| --- | --- |
| DataRaptor Turbo Extract | Gets data from a single Salesforce object (sObject) |
| DataRaptor Extract | Gets data from one or more Salesforce sObjects |
| DataRaptor Load | Saves data to one or more Salesforce sObjects by:   * Updating Salesforce records * Creating Salesforce records * Migrating CSV data into Salesforce records |
| DataRaptor Transform | Manipulates any data that comes from inside or outside Salesforce |

DataRaptor Extract, DataRaptor Load, and DataRaptor Transform also do the following.

* Trim, map, restructure, and transform data in JSON, XML, or Custom Schema.
* Use formulas and functions to transform data.

Interested to know more? Next, we break down the capabilities of each type of DataRaptor.

## DataRaptor Turbo Extract

DataRaptor Turbo Extract retrieves and filters data from a single Salesforce object type with support for fields from related objects. You can filter the data and select the fields to return. Unlike a standard DataRaptor Extract (which we describe next), a DataRaptor Turbo Extract doesn’t support formulas or complex output mappings.

It has two advantages over a standard DataRaptor Extract.

* Simpler configuration
* Better performance at runtime

An example of when to use a DataRaptor Turbo Extract is to retrieve contacts for an account (having a specified Id).

## DataRaptor Extract

DataRaptor Extract, the second type of DataRaptor that pulls data from Salesforce, reads Salesforce data and returns results in JSON, XML, or custom formats via complex field mappings. You typically use DataRaptor Extracts to provide OmniScripts and FlexCards with any internal Salesforce data they need to display. They support formulas and complex output mappings.

Use DataRaptor Extracts if you are:

* **Extracting data from a single object:** For example, to retrieve account data such as Account Name and other details from the Account object in Salesforce.
* **Extracting data from three related objects:** For example, create a DataRaptor Extract for use by a case-handling OmniScript. The agents who handle cases need to look up a case using the case number, and need to be able to view the name of the account, the description of the case, and all the contacts for the account.
* **Paging through sorted data using data values or offset values:** If you expect a DataRaptor Extract to retrieve a lot of records, use paging to retrieve a few at a time based on field values or offset values. For example, page through a long list of accounts by Account Id or contacts by last name.

## DataRaptor Load

DataRaptor Loads write data to Salesforce objects from JSON or XML input. A DataRaptor Load updates records with changed data and also creates new records at the same time.

* To modify the input data, you define formulas, transform values, and change the output data type.
* To specify how the resulting data is written to Salesforce objects, you map fields from the output JSON to fields in Salesforce objects.

The DataRaptor Load applies its mappings and formulas to the input data to create the output data, then loads the output data into Salesforce objects according to the mappings.

Use DataRaptor Loads if you are:

* **Creating a contact and using a formula:** A DataRaptor Load creates a new Contact record. A formula checks whether the contact is over 18 years old; if so, a custom Authorized field is set to true.
* **Creating a contact for an existing account:** A DataRaptor Load creates a new Contact record. A link to an Account record with a specific Id ensures that the new Contact is related to that account.

DataRaptors can access external objects and custom metadata as well as Salesforce objects. No special syntax or additional configuration is required to do this.

## DataRaptor Transform

DataRaptor Transforms do the following.

* Convert JSON input to XML output, and vice versa.
* Restructure input data and rename fields.
* Substitute values in fields (all DataRaptors can substitute values).

Use DataRaptor Transforms for the following situations.

* When an OmniScript must populate a DocuSign template.
* When an OmniScript must fill fields in a PDF document.

In the following example, an insurance company implemented DataRaptor Transform so that its labels make more sense to the user. Policy information is stored in the Salesforce Asset sObject, but DataRaptor Transform changes the label the user sees from “Asset” to “Policy.” It also changes “vlocity\_ins\_\_AnnualPremium\_\_c” to “Annual Premium.” And it changes two labels for Contacts from “Contact” to “Policy Holder,” and from “Contact” to “Policy Beneficiary.”

