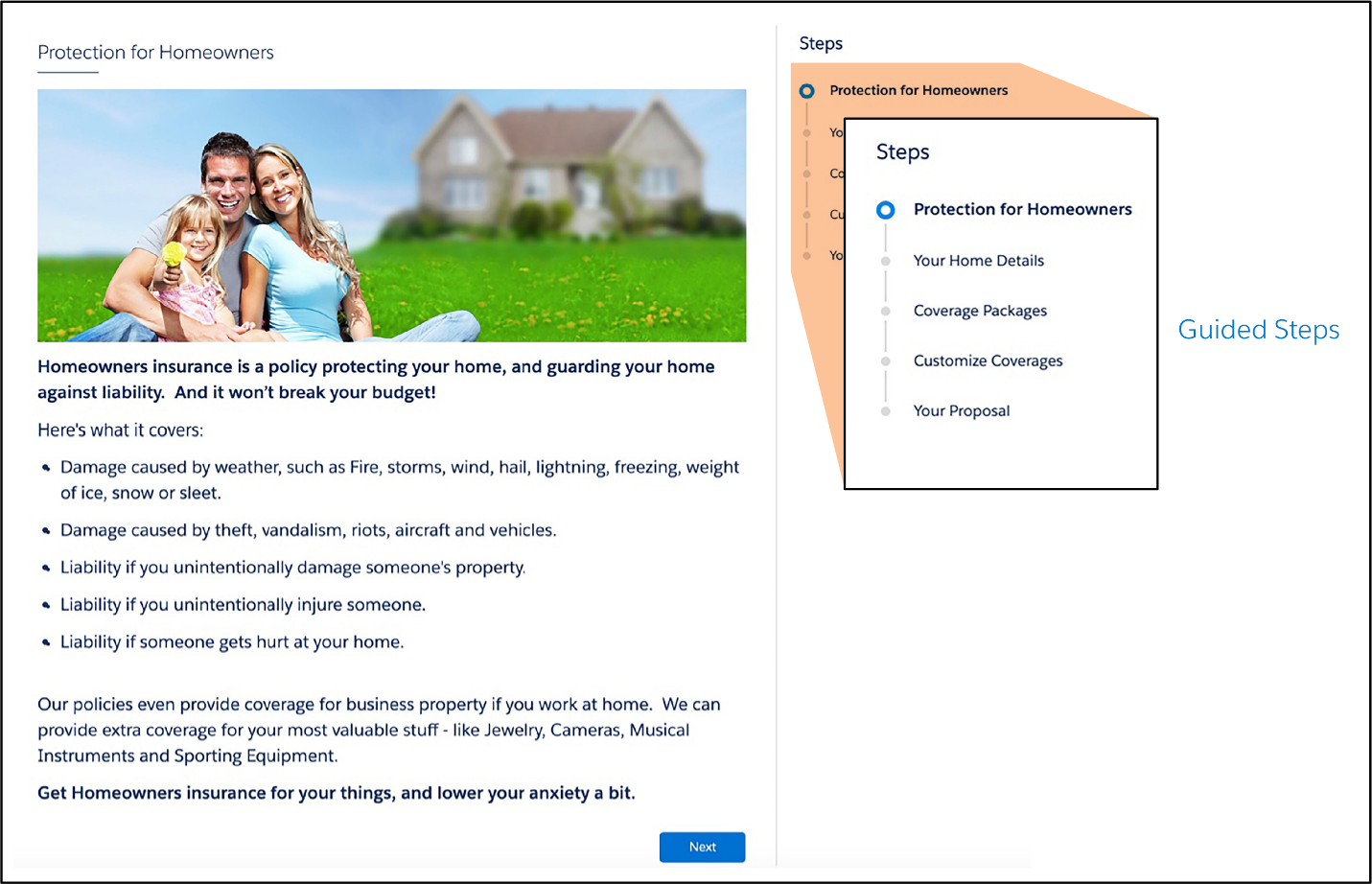
**The Lowdown on OmniScripts**

An OmniScript gives customers a guided path for completing a business process and serves as a configurable way of creating a seamless customer experience (which is *always* a good thing). In the following image, you see an OmniScript walking a customer through several steps to select an insurance policy. First they provide home details, then they select standard coverage packages. They are then prompted to select customizations for the packages. After all of this, they view their proposal for coverage. If they accept the proposal, they can complete the transaction and activate their coverage.



Here are a few other instances of when you might use OmniScripts.

* A customer service agent adds a new customer and captures details for a service implementation, such as network configuration requirements.
* A customer steps through a selling process, such as choosing a new insurance plan.
* An insurance agent updates a policy.
* A customer completes a self-service interaction such as troubleshooting a service outage.
* A customer completes forms for different services, such as government benefits, insurance policies, and healthcare coverage.

With an OmniScript, you configure interactive business processes that are easy to use, yet have complex functionality occurring behind the scenes. OmniScripts also have built-in branching capability, which means they show different pages and groups of fields based on choices the user makes. For the customer, this translates into a dynamic and personalized experience.

## Key Capabilities

**Build OmniScripts Quickly with Drag and Drop and Low to No Code**

OmniScript is a declarative scripting tool you create with clicks, not code. Using the OmniScript Designer, you drag and drop items to build the structure of the OmniScript, then preview and debug your work using the built-in troubleshooting tools.

this means you can quickly create and easily maintain OmniScripts, which saves lots of time.

**Use OmniScripts on Any Device and Any Channel**

OmniScripts are not restricted to OmniStudio Interaction Consoles for agents to use. You can deploy them on any device and any channel, such as a mobile device, or a consumer portal. Here’s what an OmniScript looks like when viewed on a mobile device and online.

Being able to view the same OmniScript on multiple channels without having to change the configuration is another time-saver!

**OmniScripts Have Modular Architecture**

An OmniScript’s look and feel (frontend) is separated from its functionality (backend). OmniScripts separate the JavaScript Object Notation (JSON) metadata structure (1), the stylesheets (2), and the data (3) from each other.

This modular architecture supports prototyping and building the user experience quickly. It also promotes using data from anywhere, reuse of JSON metadata, and ease in applying branding standards.

**Display Data from Multiple Data Sources**

An OmniScript can display both internal data from Salesforce and external data from a website or a third-party legacy system.

OmniScript’s data-oriented Actions elements call application programming interfaces (APIs), OmniStudio Integration Procedures, and other tools to access data from anywhere. You integrate data from multiple sources (Salesforce or third-party), manipulate the data, and send it back to its source, all from within the OmniScript. The data is captured in the standard JSON format.

Client-side execution improves performance and reduces API calls.

**Rebrand OmniScripts to Suit Your Customers**

You can control both the style and appearance of OmniScripts. You change the appearance of an OmniScript two ways. First, by using custom Lightning stylesheets to determine whether the guided interaction has a horizontal or vertical mode, branding, or any other aspects you wish to see. Second, by using the Newport Design System (NDS). NDS includes a complete set of customizable, global styles and is a Cascading Style Sheets (CSS) framework. It lets designers and web developers easily restyle all of their OmniStudio components in a single place and generate a custom, optimized CSS file that can be used in all future pages. It can even be used for non-OmniStudio and non-Salesforce pages.

**Manage Signed Documents with OmniScripts**

You can create dynamic documents from templates (MS Word, PDF, and HTML outputs), merge data from any data source (for example, Salesforce objects, or user inputs such as sales quotes, order forms, and contracts), and then create and operate on these documents within OmniScripts. You attach them to Salesforce records and email them to recipients, plus delegate to DocuSign for eSignatures. The signed document received from DocuSign is automatically attached back to the latest version of a contract.

## Get a Little Guidance from OmniScripts

OmniScript guided interactions do exactly what their name suggests: They guide users through sales and service processes, enabling them to quickly and easily achieve their goals. These interactions are dynamic, agile, scalable—and often personalized—creating customer experiences that are engaging and efficient.

**Build an OmniScript with Clicks, Not Code**

The OmniScript Designer lets you create the structure of an OmniScript using “clicks, not code.” It has a drag-and-drop interface with what-you-see-is-what-you-get (WYSIWYG) editing.

With it, you can:

* Preview elements inside steps.
* View property changes live.
* Access contextual guidance with in-product help.

More specifically, as you build the OmniScript, you’re able to:

* Add actions that extract or save data or send an email.
* Add input fields to enter data such as a user’s name, address, and other information.
* Configure calculations and messages that provide immediate feedback and error checking to the user.
* Create a function, such as a formula for performing a calculation within the OmniScript.
* Create branches that dynamically adjust the controls and enable or disable steps depending on choices the user makes in the guided process.
* Group items together by creating a step or displaying a list of items for the customer to select from.
* Refine the display by using a headline or text block.

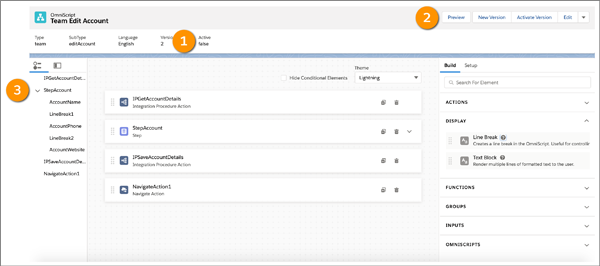
**The Designer’s Fantastic (and Fast) Features**

With OmniScript Designer’s features, you can:

* Build your OmniScripts on a wide and adjustable canvas, and instantly view changes made to element properties.
* Search for and drag elements onto the canvas from the Build panel.
* Reposition, clone, and adjust the width of elements with a 12-column snap-to-grid.
* Configure elements from the Properties panel.
* Configure script-wide settings from the Setup panel.
* Access inactive elements and navigate between them in high-level and detailed views from the navigation panel.
* Preview, test, and debug your script in Preview.
* View contextual in-product help to discover and learn about elements and properties without leaving your script.

Let’s take a closer look at the OmniScript Designer.

**Header and Navigation Panel**

In the header (1), you:

* Perform actions related to your script using the navigation bar.
* View high-level metadata about your OmniScript, such as Type, SubType, Version, Language, and Activation status.

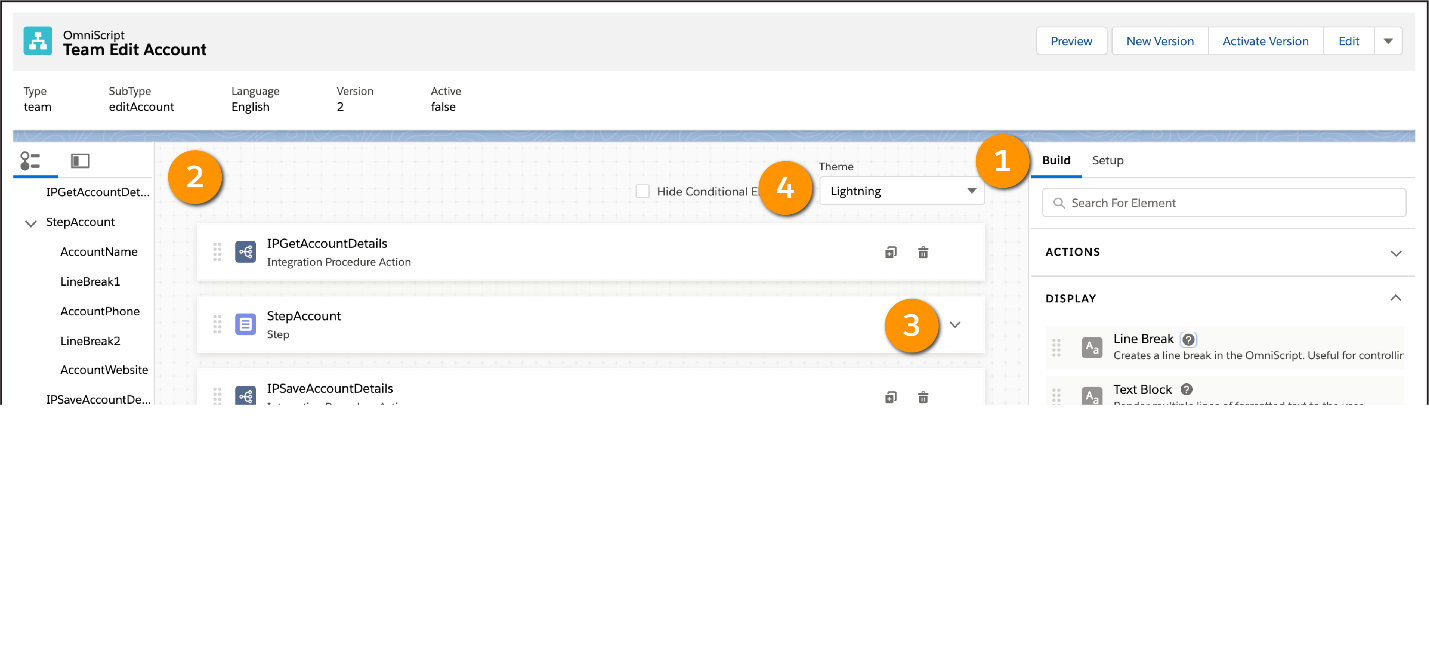
In the actions navigation bar (2) in the header, you:

* Toggle between Design and Preview views.
* Create a New Version, and Activate or Deactivate the current Version.
* Edit basic settings.
* Download your OmniScript.
* Get launch instructions.

In the navigation panel (3), you:

* Access and navigate between active and inactive actions, steps, and step elements.
* Use the Slide View tab for a high-level view of the structure of large and complex scripts. Click on a slide to open the Properties panel for a step or action.
* Use the Tree View tab for a detailed view of the script's structure. Click a branch to open the Properties panel for a step, element, or action. Use the Tree View to access inactive steps or elements.

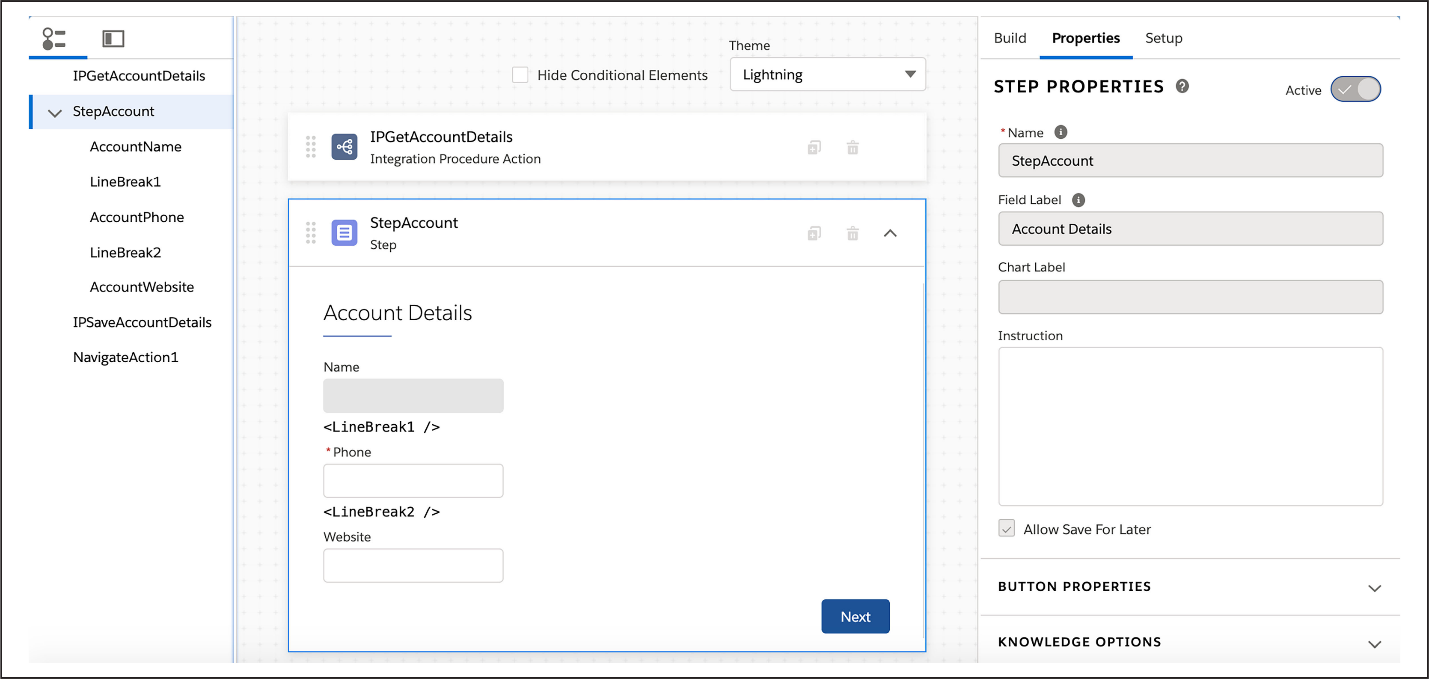
**Canvas and Build Panel**

Build your scripts by dragging elements from the Build panel (1) onto the canvas (2).

* Rearrange, clone, and delete elements as needed.
* Adjust the width of the canvas from either side.
* Expand steps (3) to preview and configure elements within them. Adjust the width of elements on a 12-column grid, and drag elements next to each other so that they automatically take up the remaining width of the grid.
* See how your scripts look with a Newport or Lightning theme (4) without switching to Preview.

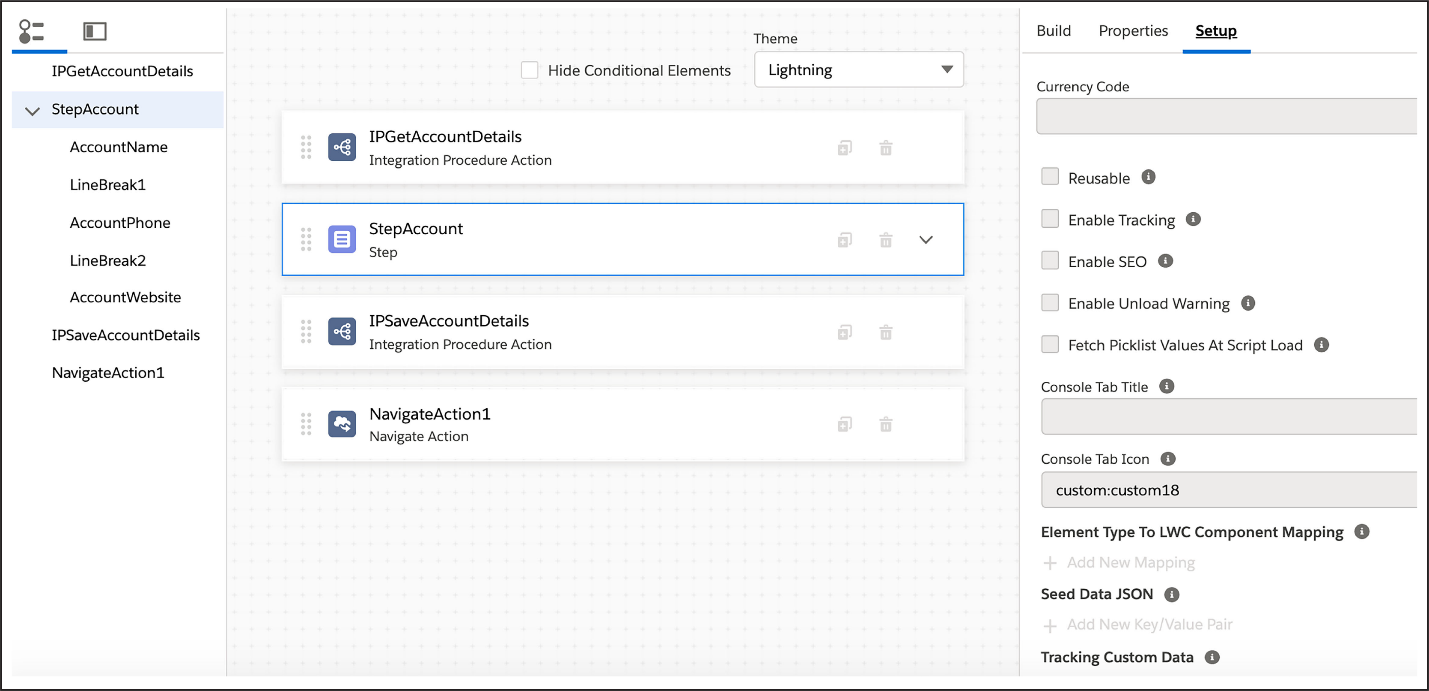
**Properties Panel**

Configure properties for action, display, function, group, input, and embeddable OmniScript elements in the user interface (UI), or edit properties as JavaScript Object Notation (JSON).



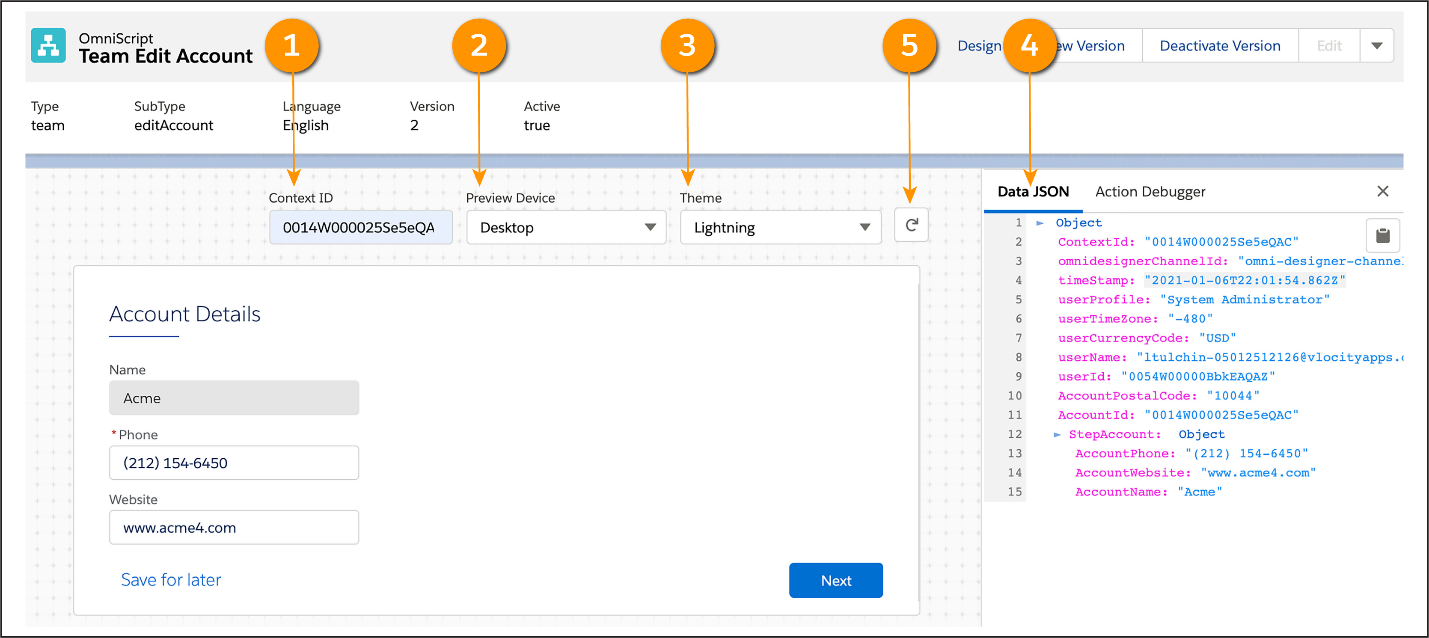
**Setup Panel**

Configure optional script-wide settings in the Setup panel.



Configure basic settings: Step Chart Options, Save Options, Knowledge Options, Error Messages, Messaging Framework, and Lightning Design System Tokens in the UI, or edit properties as JSON.

**Preview Options**

****

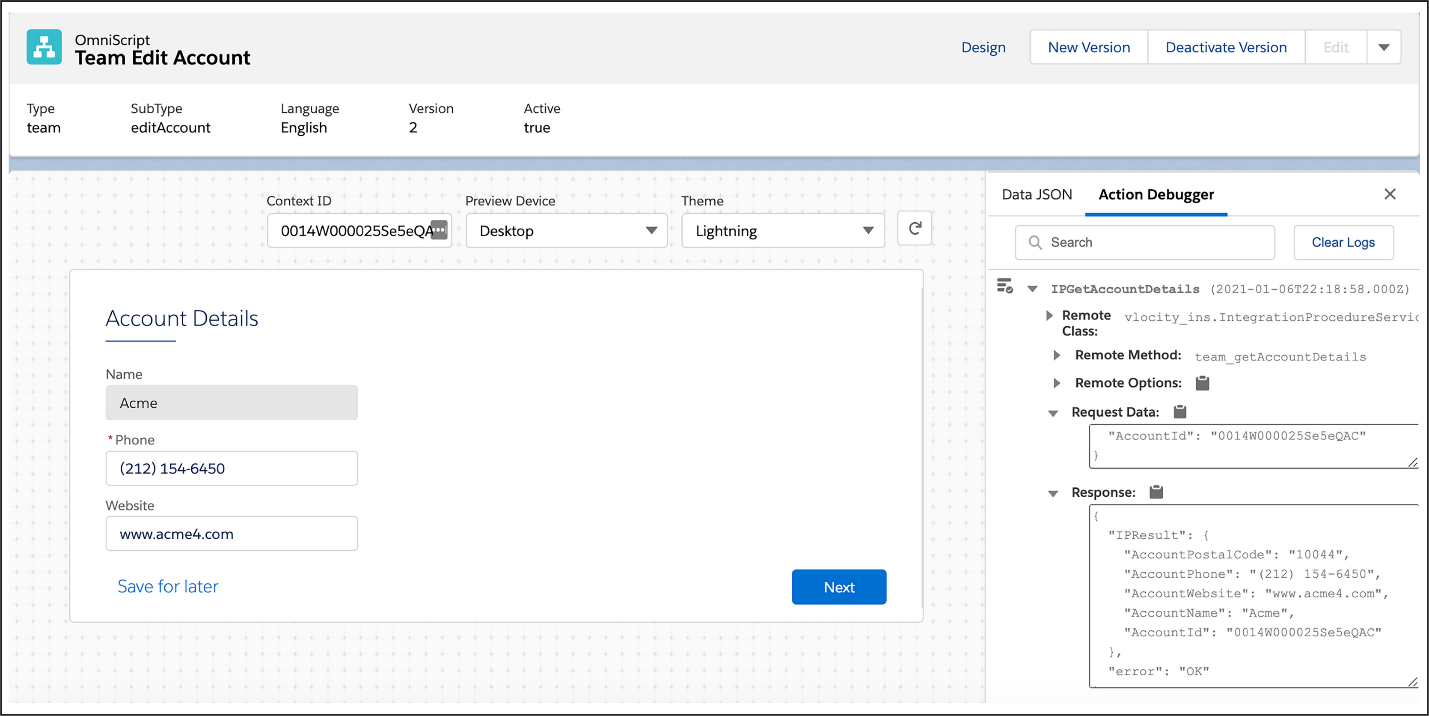
* Preview your script in real-time by entering a RecordId into the Context ID field (1) and refreshing to preview your form with live data. (If you are using stub data, test data will preview automatically.)
* Preview how an OmniScript appears on different devices, such as mobile, desktop, and tablet, with the Preview Device dropdown (2).
* With the Theme dropdown (3), see how your OmniScript looks with a Lightning or Newport theme. If a custom Newport stylesheet is in the org, it overrides the out-of-the-box Newport stylesheet.

**Data JSON**

* The Data JSON (4) provides an easy-to-read JSON format, which updates when you enter values in data fields on the canvas. Copy the entire JSON with just one click if needed.
* Reset Data (5) to reload the canvas and update the Data JSON and the Action Debugger.

**Action Debugger**

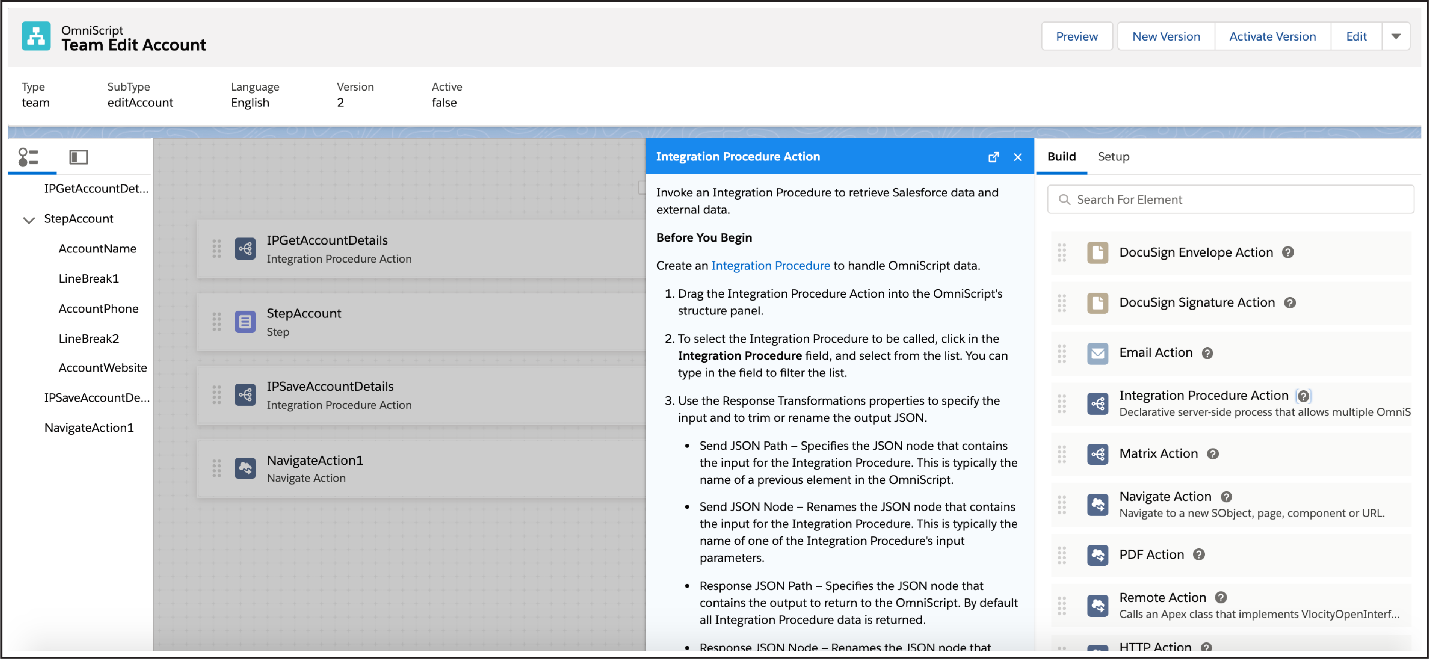
The Action Debugger lets you debug action requests and response data. Search for actions, copy specific nodes in one click, and clear the logs.



**In-Product Help**

In the Build, Properties, and Setup panels, use the in-product help feature to view contextual information and instruction about elements and properties without leaving your script.

* Access in-line information about properties with tooltips.
* View detailed documentation about element functionality with slide-out help panels.



**Examine OmniScript Elements**

OmniScripts are built by moving elements into the canvas and defining their properties. These elements are placed in a specific sequence for the workflow. There are elements for:

* User interaction on the user interface (UI).
* Getting, saving, and manipulating data behind the scenes.

OmniScript elements are grouped by type, based on the kind of functionality they perform, and they each have settings that must be configured appropriately.

Here are the different types of elements and how they can be used.

| **Element Type** | **What Is It For?** |
| --- | --- |
| Actions | For calling on other tools to perform various actions: getting or saving data, calculating, sending an email, and so on |
| Display | For displaying text and images on the screen to enhance the usability of the UI |
| Functions | For performing calculations within the OmniScript, showing conditional messages, and providing geolocation |
| Group | For grouping elements together on the UI |
| Inputs | For system or user input or selection |
| OmniScripts | Reusable OmniScripts to insert and use |

**Actions elements**

Action elements are used to:

* Get data from one or more Salesforce records (DataRaptor Extract Action).
* Update the data in one or more Salesforce records (DataRaptor Post Action).
* Call a series of actions (Integration Procedure Action).
* Get, save, or delete data through a web application programming interface (API) (HTTP Action).
* Populate and send an email as part of an interaction (Email Action).
* Send DocuSign emails for signature or sign DocuSign emails (DocuSign Envelope Action and DocuSign Signature Action).
* Send the user back to a previous page after the interaction is complete (Navigate Action).

**Display Elements**

Display elements let you provide important context within the workflow or help you with the layout of the page.

Display elements include:

* **Line Breaks element:** Allows you to create a line break to control the spacing on the page.
* **Text Block element:** Allows you to add instructions, images, and hyperlinks.

**Functions Elements**

Functions elements are commonly used to:

* Perform average/sum calculations (Aggregate).
* Perform formula functions and algebraic calculations (Formula).
* Display warnings, errors, and other feedback to users based on conditions (Messaging).

**Group Elements**

Use these elements to group items together, such as the step/page view. For example, Step elements group interaction items onto one view or page. They are the UI canvas for a step in the flow.

Another type of Group element, Type Ahead Blocks, includes a combination of functionality that provides autocomplete and search. For example:

1. The user begins to enter information.
2. The system searches for matches and displays a list of the matching items.
3. The user then chooses the appropriate item.

Other commonly used Group elements include:

* **Block:** Group elements together within the page/view.
* **Edit Block:** Add, edit, or delete records in real time.
* **Radio Group:** Group radio buttons such as when a user has common choices on a questionnaire.

**Inputs Elements**

Inputs elements enable user input. With these element types, a user:

* Enters information, such as payment method (1), the payment date (2), and the payment amount (3).
* Selects one or multiple options (4).
* Selects a checkbox.
* Enters or views text, a phone number, an email address, or a website.

An OmniScripts element (Child OmniScript) calls an OmniScript to reuse it within another one. This allows you to create smaller “child” OmniScripts for common functionality, such as showing specific contact or account fields.

## What Makes an OmniScript Unique?

An OmniScript’s Type, Subtype, and Language gives an OmniScript its unique identity. Only one active OmniScript may have the same Type, SubType, and Language at any time. An OmniScript’s Type must start with a lowercase letter.

Only one version of an OmniScript may be active at a time. If you need to make a change to an active OmniScript, create a new version. The version of the OmniScript already active and in production remains in place while you work on the new version.

## Add an OmniScript Step and Inputs

Steps are how you organize a page. Once you create your OmniScript, you then configure the Step.

1. From the Build panel, drag a Step element onto the canvas.
2. Drag the following Inputs elements into the Step creating fields to display and allow the end user to edit data: Text (1), Phone (2), and Website (3).

**Configure Element Properties**

Many elements have the same types of properties. For example, each element must have a Name.

Element names must be unique in an OmniScript. Here are some other guidelines to follow.

* Use Pascal Case and no spaces for element names (for example, StepAccount).
* Use object and field names for clarity (for example, AccountName).
* Specify the object for Ids (for example, AccountId).

For Display or Inputs elements, element labels are what the end users see on the screen. For Actions elements, element labels display in the Action Debugger. Labels do not need to be unique.

How is this accomplished?

| **Property** | **How is it enabled?** | **What does it do?** |
| --- | --- | --- |
| Read Only | Select a checkbox. | Restricts the user from making changes to a field. |
| Required? | Select a checkbox. | Requires the end user enter data in the field before moving on. |
| Width Settings | On the Canvas, drag the responsive control width grid. | Adjusts the size of a field using HTML web standards (dynamic control). |

When you finish configuring your Inputs elements, you’re ready to add Actions elements. Before you do that, though, let’s take a minute to learn about the way data flows into and out of an OmniScript.

## How Data Flows into and out of an OmniScript

There are several data input options for an OmniScript. Using Integration Procedures is a best practice. We explain in detail why in the OmniStudio Integration Procedures module, but Integration Procedures allow you to separate data configuration from OmniScript configuration.

Here are the scenarios for using the other data input options.

| **For This Data Input Source** | **Use This OmniScript Action Element** |
| --- | --- |
| Salesforce Org—One field only | Lookup |
| Salesforce Org—Many fields | DataRaptor Extract Action |
| API | HTTP Action |
| Anything that Apex can access | Remote Action |
| User | Any Inputs Element |

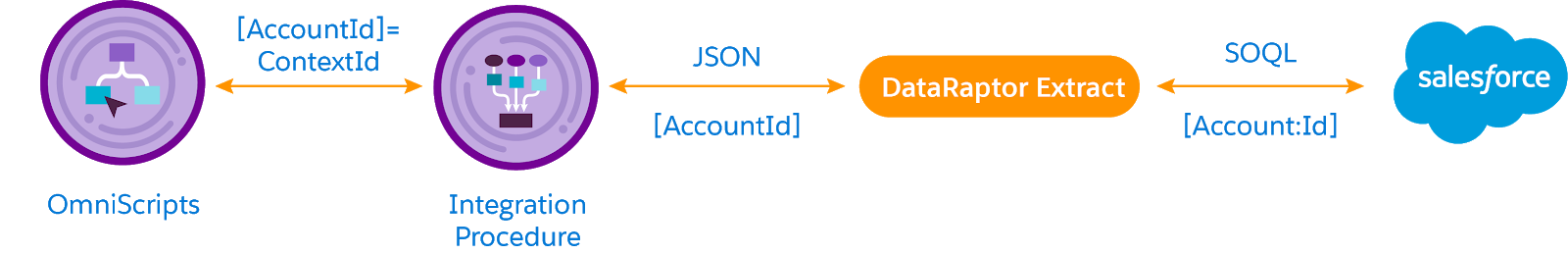
So, how do you move data from Salesforce to an OmniScript using an Integration Procedure? When an OmniScript is launched via an Action on a FlexCard, the RecordId is passed along the JSON in a node (in this example, it’s called AccountId). The RecordId is saved in the OmniScript in a variable called ContextId.



Once the variable is in the OmniScript, pass the RecordId via a variable called AccountId to the data tool you plan to use. Following best practices, choose an Integration Procedure to pass the AccountId to a DataRaptor Extract. AccountId is an arbitrary name, but it must be the same in each of the elements in order for data to flow properly between them and Salesforce.

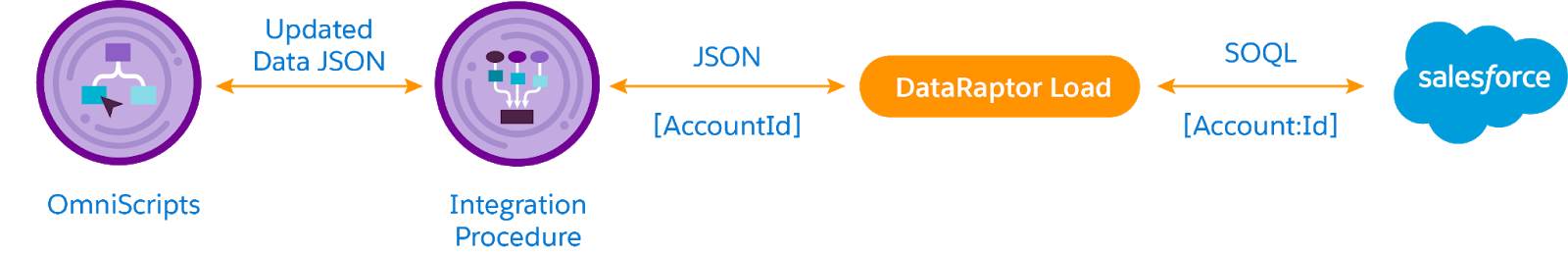
The DataRaptor Extract uses the AccountId in a Salesforce Object Query Language (SOQL) query to retrieve data from Salesforce. This includes the AccountId and data from the Account record. The DataRaptor Extract maps them to a data JSON, which is sent to the Integration Procedure and then to the OmniScript. The OmniScript uses a parser to match the JSON that comes from the Integration Procedure to the inputs in the OmniScript based on element names. If the JSON doesn’t match the element name, the fields will appear empty.

The data flow looks like this:



The user and other OmniScript actions manipulate the data while the OmniScript runs.

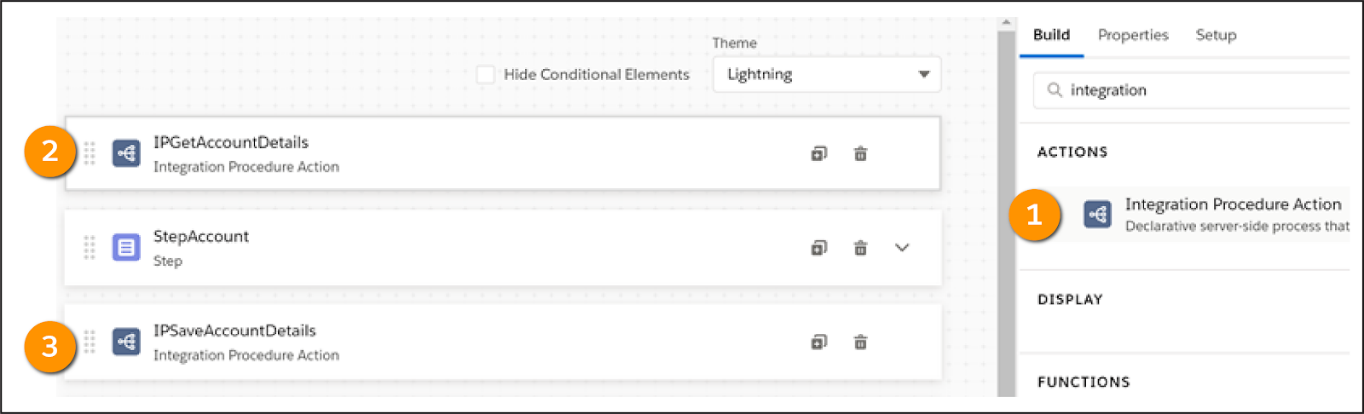
An updated data JSON, including the AccountId, is then passed to an Integration Procedure, which passes the JSON with AccountId to a DataRaptor Load. The DataRaptor Load uses the AccountId to identify the original account record and updates the data in Salesforce.



Your job is to configure the Action elements to ensure the data flows in correctly from the FlexCard and out of the OmniScript back to Salesforce.

**Configure OmniScript Action Elements**

To configure OmniScript Action elements, return to the canvas. Let’s take a look at this example, which configures two Integration Procedure Actions, to understand how this works.



1. From the Build panel, drag an Integration Procedure Action (1) onto the canvas. This calls an Integration Procedure that gets the data. Place the Action (2) above the Step element.
2. From the Build panel, drag an Integration Procedure Action (3) onto the canvas to call the Integration Procedure that saves the data. Place it below the Step element.

When placed outside a step, Actions execute automatically in the order they appear. When placed inside a step, an Action appears as a button that must be clicked before it executes.

Actions have their own naming conventions. Use a prefix for the element type, such as DR for DataRaptor or IP for Integration Procedure. Then use VerbObjectDetail. In the Edit Account OmniScript, IPGetAccountDetails is the name of the Integration Procedure Action (2) above the Step, and IPSaveAccountDetails is the name of the Integration Procedure Action (3) below the Step.

Each Integration Procedure Action has a dropdown list from which you select the Integration Procedure you want to call from the OmniScript. When you link an Integration Procedure to an OmniScript, the OmniScript sends the entire JSON to the Integration Procedure by default.

When the JSON is small, as it is in the Edit Account OmniScript, there’s no need to trim or redefine it. However, you do need to confirm what the ContextId is in that JSON. For the Edit Account OmniScript, for example, you need to tell the Integration Procedure that the ContextId is the AccountId.

But isn’t AccountId an arbitrary name? Yes, it is! Still, it’s important to follow the best practice of being explicit about what type of RecordId you are sending to an Integration Procedure.

When an end user completes the OmniScript, the Navigate Action tells the OmniScript where to send the end user. If an agent opens the OmniScript from a FlexCard on a console, you can configure the Navigate Action to return the agent to the console.

Once you have added the Navigate Action and previewed the OmniScript, don’t forget to activate it! And there you are. You have not only learned about the power of OmniScripts to guide end users through a process, you’ve learned how to design and create a simple one-step OmniScript.