Web Developer Specialization - Sample Exam

Before Starting

This sample exam has 10 questions that will help you get ready for the Web Developer Specialization exam. We recommend that you prepare for a real exam environment as much as possible.

- Find a quiet room just for you.
- Get a stopwatch or set a timer for the (recommended) duration of 30 minutes.

The last page of this document has the correct answers. Don't peek! Use it only after completing your exam, to check how well you did.

During the Sample Exam

To accurately simulate a real exam environment, we suggest that you:

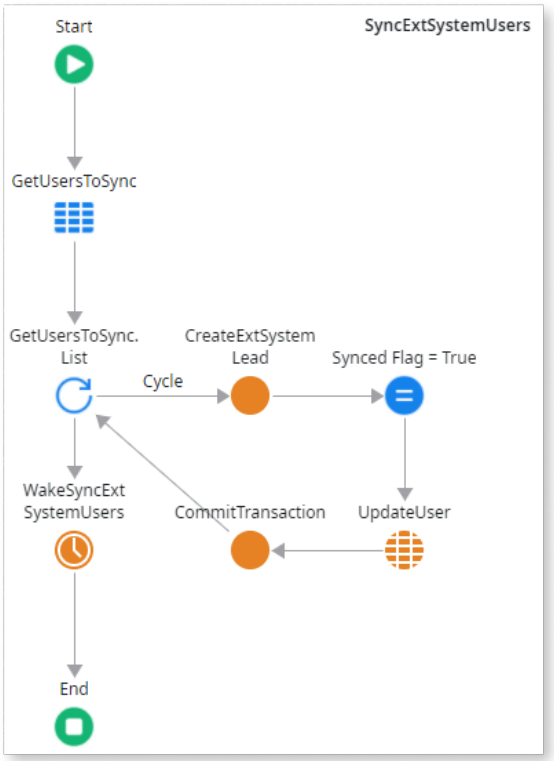
- Read each question and its answers carefully.
- Take your time! Questions may be revisited and your choices can be changed.
- Mark the questions that you want to review at the end.
- Pick only one answer per question, as only one is correct.
- Answer all questions, as there's no benefit in not doing so.
- If possible, turn off all electronic devices during the exam.
- Refrain from using or reading any external materials during the exam.

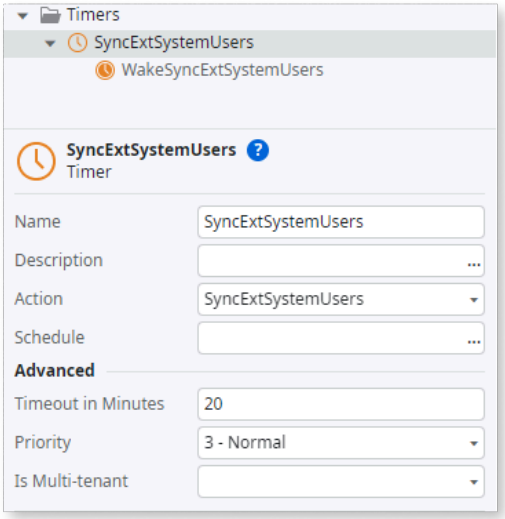
After Completing the Sample Exam

After completing the exam, validate the answers you selected by checking the ones provided on the last page of this document and count the total number of correct answers. Since the passing score is 70% or higher, you should get at least 7 questions right. In case you chose any wrong answers, we suggest you review the study materials where that specific topic is covered.

Sample Exam Questions

1. Consider the Action below triggered by the *SyncExtSystemUsers* Timer, to sync thousands of users to an external system. What is missing, if anything, for this Timer to follow the OutSystems best practices for batch processing?





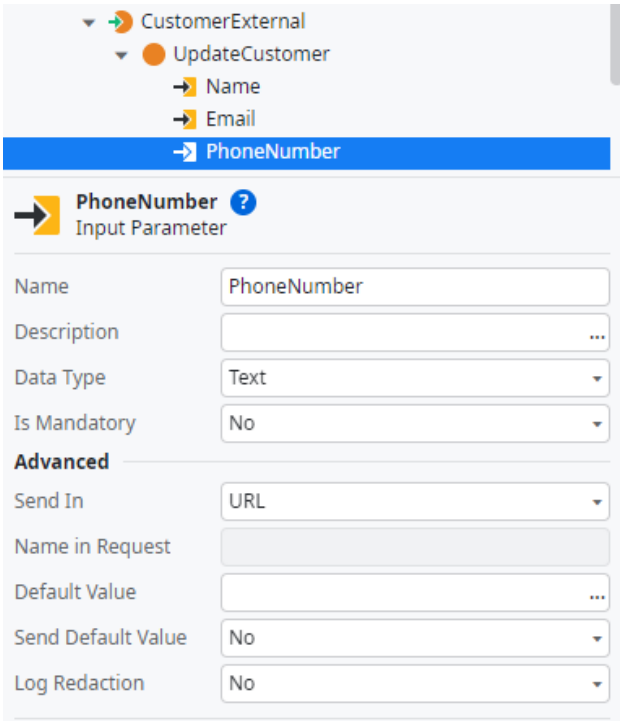
   A. Nothing is missing. The Timer and its Action already follow the OutSystems best practices.

   B. The *Timeout in Minutes* property should be set to a higher value, to guarantee that all records are processed within the Timer run.

   C. The *CommitTransaction* Action is being called too often and should be called right before the *WakeSyncExtSystemUsers* call.

   D. The Action should have some logic to control the timeout using a Local Variable. When we reach the defined timeout limit, the logic re-wakes the Timer.

2. Consider the following scenario with a POST REST API method to update a customer in an external database. The integrations team has asked you to solve an error that occurs in some of these API calls which states: "The *PhoneNumber* attribute must not be null". You confirmed the API is receiving the name and date of birth correctly, and you know that the *PhoneNumber* is not a mandatory field in your app that the end-user must fill. How can you better solve this error?



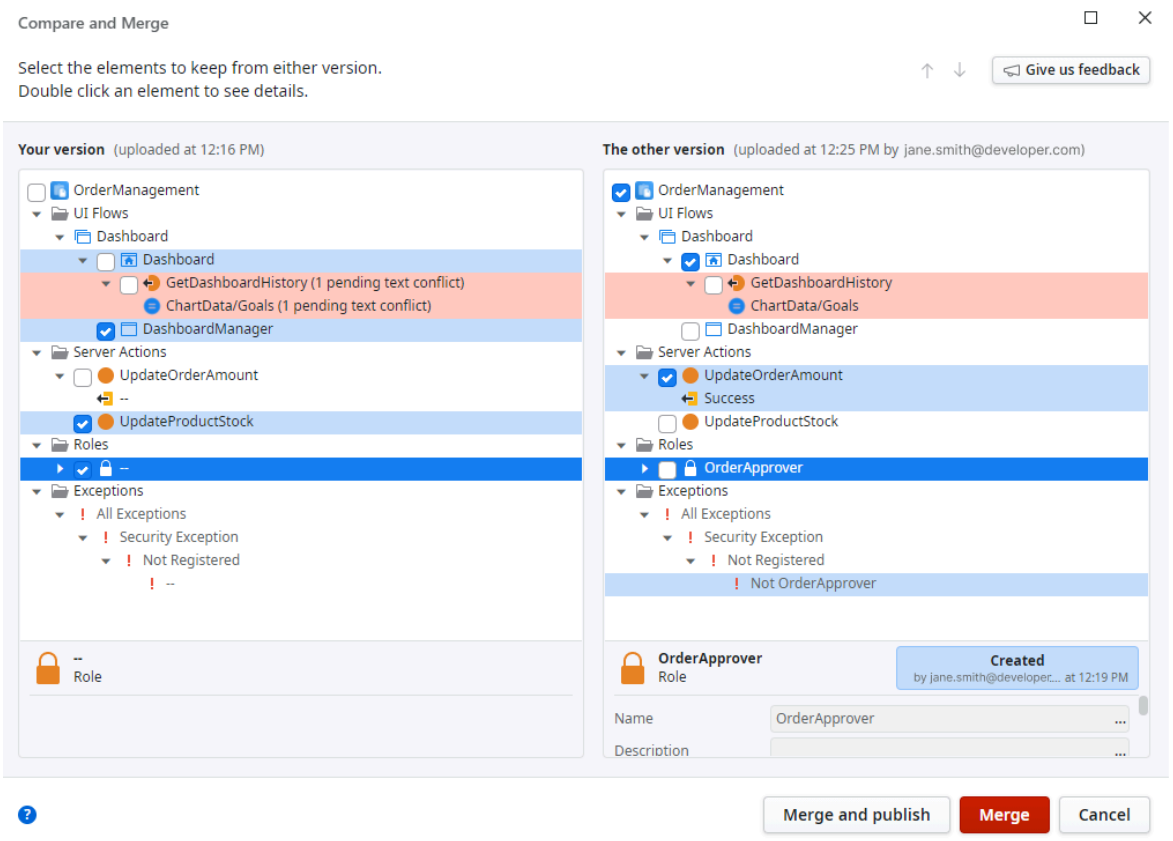
A. Set the **Default Value** property of the *PhoneNumber* input parameter to a value that represents a default phone number, when no phone number is passed to the API.

B. Set the **Send Default Value** property of the REST API to *Yes*.

C. Use the *OnBeforeRequest* callback to populate the *PhoneNumber* input parameter with a value that represents a default phone number, when no phone number is passed to the API.

D. Set the **IsMandatory** property of the *PhoneNumber* input parameter to *Yes*.

3. Considering the scenario in the following screenshot, what will happen if we select the *Merge and Publish* option?



A. The selected elements from both versions and the conflicting changes in the local version of the module will be merged, and the module will be published.

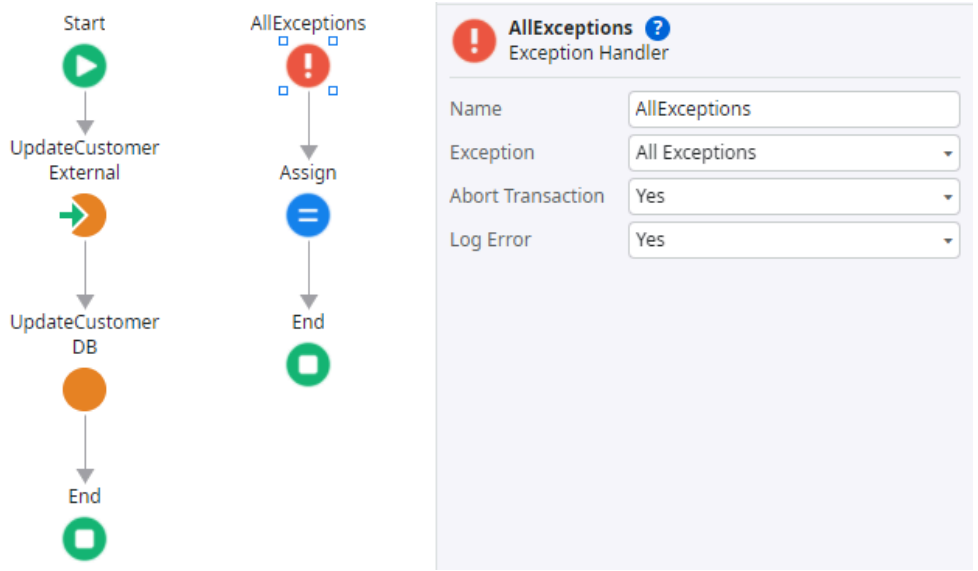
B. The selected elements from both versions and the conflicting changes in the other (already published) version of the module will be merged, and the module will be published.

C. The selected elements from both versions will be merged, the conflicting changes performed in both versions will be discarded, and the module will be published.

D. We get a message from Service Studio to solve the conflict, since the merge cannot be performed while the conflict persists.

---

4. If an error occurs in the *UpdateCustomerDB* Action in the following Action flow, what happens to the transaction?



A. The changes made by the *UpdateCustomerExternal* and *UpdateCustomerDB* are rolled back.

B. The changes made by the *UpdateCustomerExternal* and *UpdateCustomerDB* are committed.

C. The changes made by the *UpdateCustomerDB* are rolled back, but the changes made by the *UpdateCustomerExternal* are not rolled back.

D. The changes made by the *UpdateCustomerExternal* are rolled back, but the changes made by the *UpdateCustomerDB* are committed.

---

5. Consider an application to manage *Orders*. Each *Order* has several fields, including a *Description*. This *Description* is a text field that can be longer than 2000 characters. What is the best way to model this data?

A. The *Description* should be inside the *Order* Entity as a *Text* attribute.

B. The *Description* should be in an extension Entity, with a 1-1 relationship to the *Order* Entity, as a *Text* attribute.

C. The *Description* should be inside of the *Order* Entity as a *Binary Data* attribute.

D. The *Description* should be in an extension Entity, with a 1-many relationship with the *Order* Entity, as a *Binary Data* attribute.

---

6. Regarding Site Properties, which of the following options is **correct**?

   A. Changing the value of a Site Property requires republishing the apps.
   B. Changing the value of a Site Property invalidates the application's cache.
   C. Site Properties can be used to store Entity Records for faster access.
   D. Site Properties values reset when the user logs out.

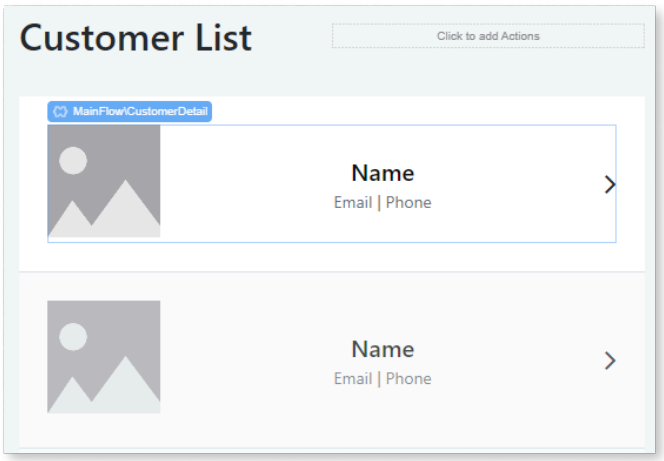
---

7. Consider an Entity that holds the annual bonus information of employees with thousands of records. For audit reasons, it's important to save the annual bonus information for 5 years, but after this period, the information is not needed anymore. What is the best option to manage and disregard these unnecessary records, knowing this should be done every month?

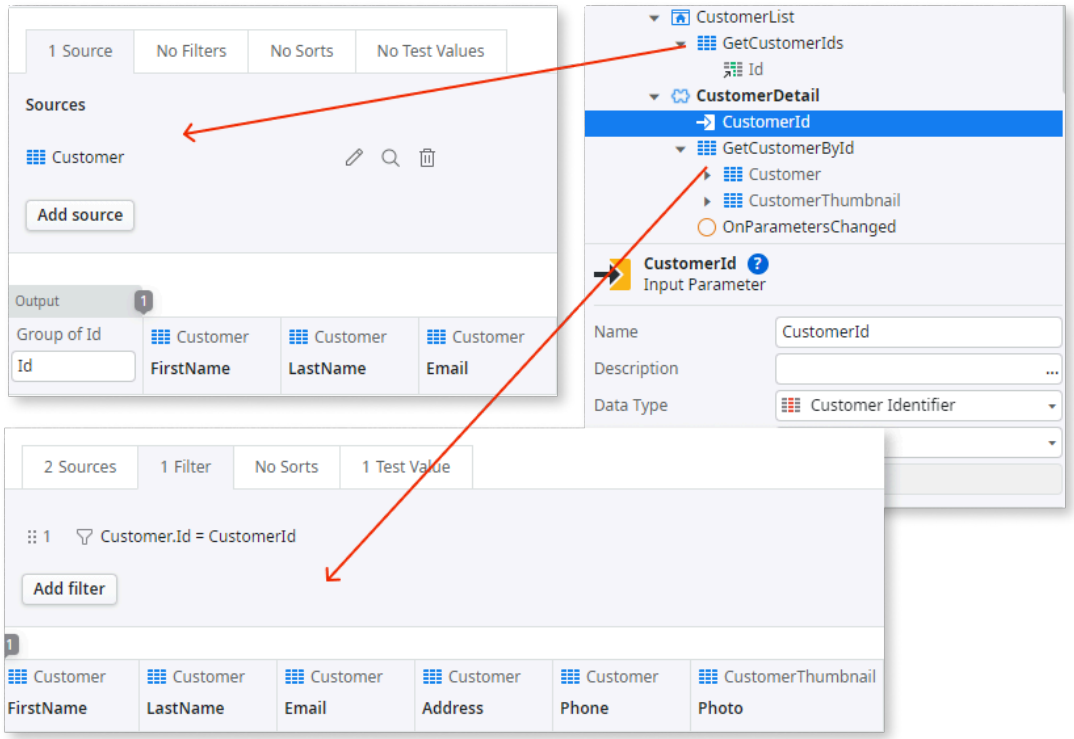
   A. Create a scheduled Timer that calls an Action that performs a bulk delete using an Advanced SQL element.
   B. Add an attribute to the Entity, *IsActive*, to manage if the records are active or not. Create a scheduled Timer that calls an Action that fetches the records that are not needed and performs a bulk update using an Advanced SQL element changing the *IsActive* attribute to *False*.
   C. Create a scheduled Timer that calls an Action that fetches the records that are not needed and loops through them, deleting the Entity's records one by one.
   D. Create a new *Archive* Entity and a scheduled Timer that calls an Action that fetches the records that are not needed and performs a bulk insert using an Advanced SQL element on the *Archive* Entity.

---

8. Consider that you are creating a Screen to display a large List of customers. Each List element will display the customer information (*Customer* Entity) and their photo (*CustomerThumbnail* Entity), with a Block to design the UI for each customer. What is the **best way** to fetch the required information and pass it to the Block?

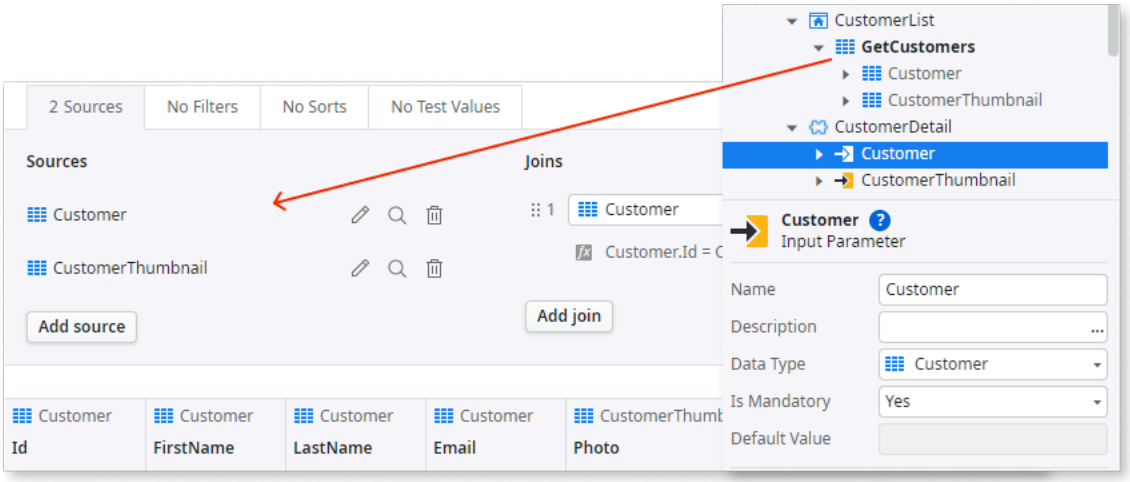


A.

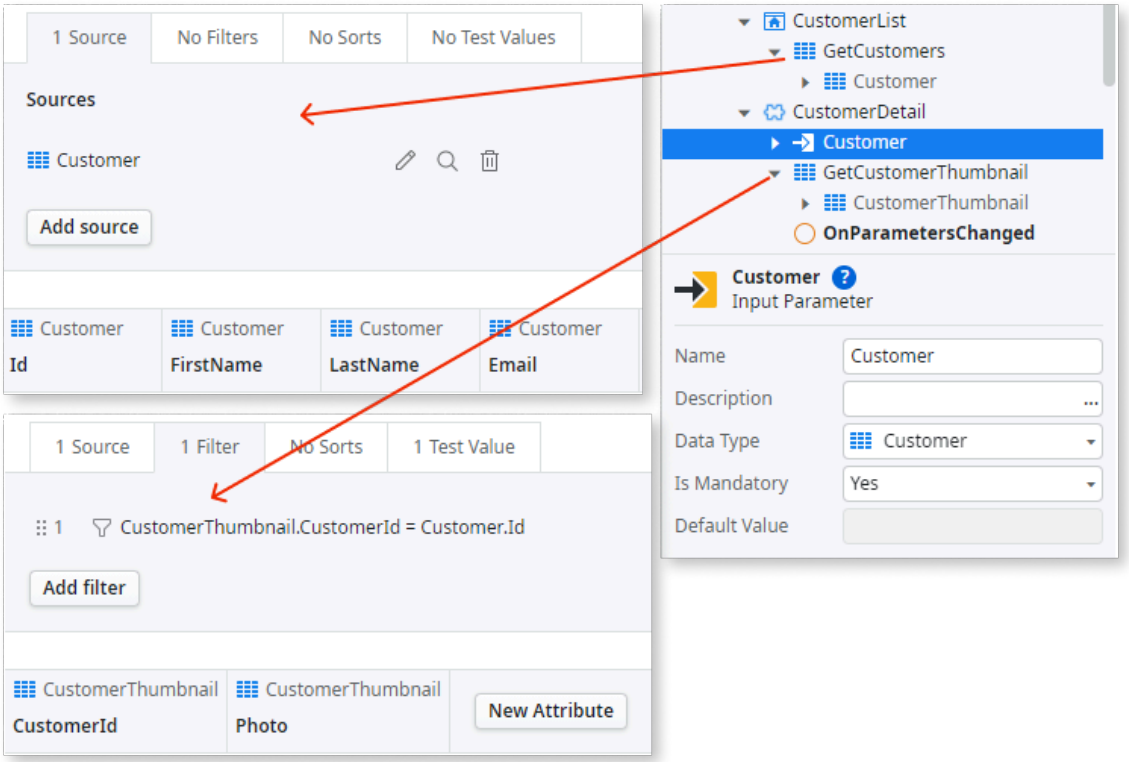


(continues on the next page)

B.

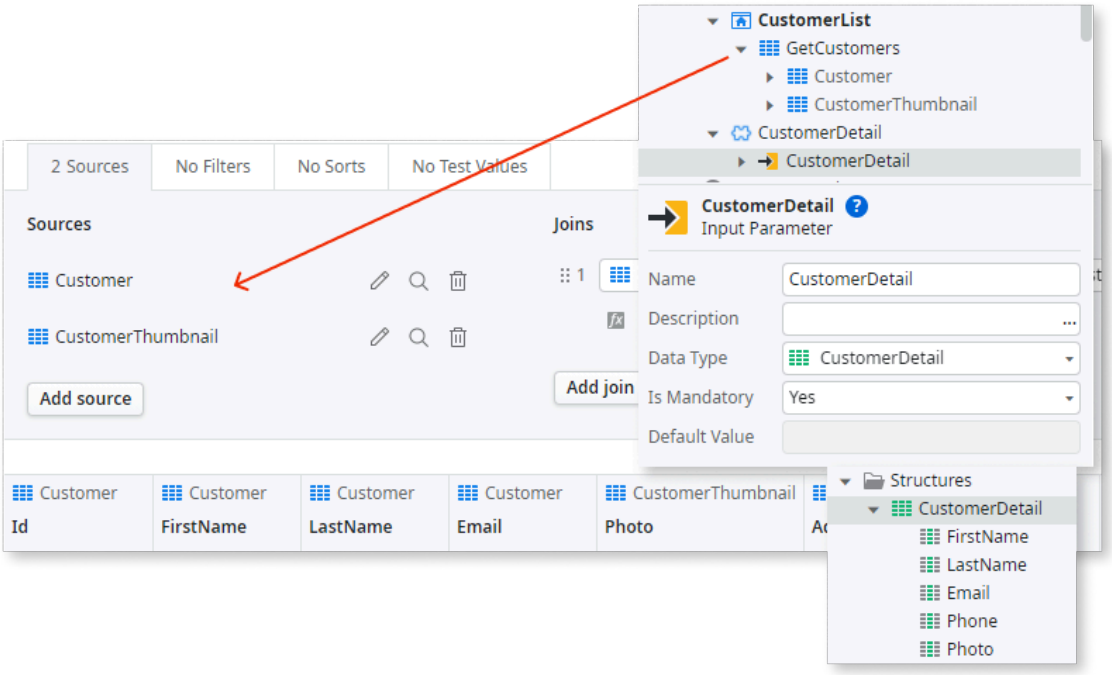


C.



(continues on the next page)

D.

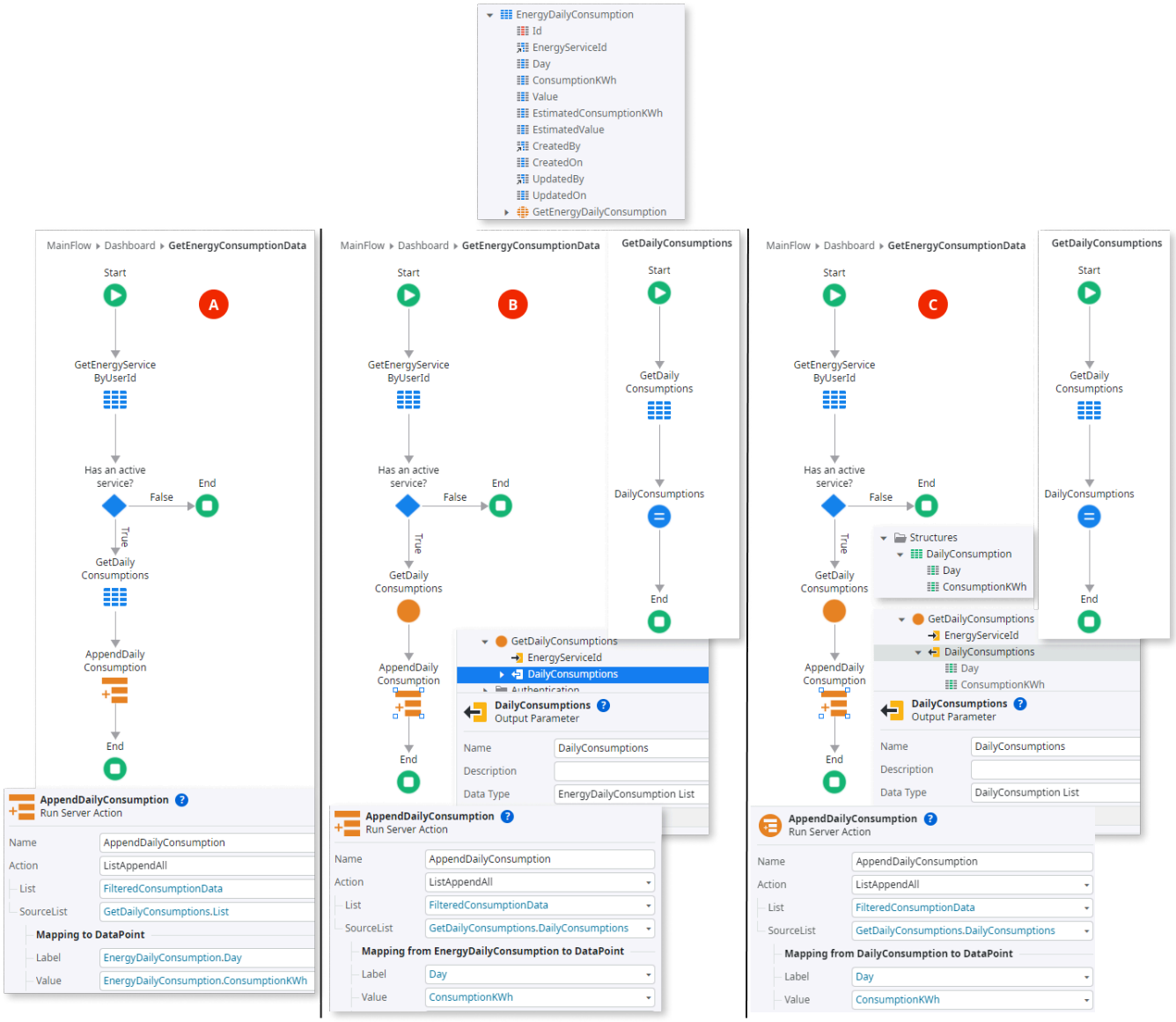


---

9. Regarding Service Center logs, which of the following options is **true**?
   - **A.** Custom logs created via the *LogMessage* Server Action appear exclusively under the *General* logs.
   - **B.** The *Default* logging level for consumed/exposed REST APIs guarantees that all requests/responses are logged.
   - **C.** Logs concerning errors in requests to consumed REST APIs appear exclusively under the *Integration* logs.
   - **D.** Service Center displays logs for the current week, plus the past 9 weeks, per type of log.

---

10. Consider a scenario where we need to fetch the daily energy consumption and display it in a chart on a *Dashboard* Screen. The daily energy consumption is saved in the *EnergyDailyConsumption* Entity in the next image. The relevant information for the chart is the day (*Day* attribute) and the consumption in kWh (*ConsumptionKWh* attribute), and this calculation is going to be needed in other places in the app. Which of the following approaches, if not all of them, follow the recommended best practices to support this scenario?



A. Solution A

B. Solution B

C. Solution C

D. All solutions follow the recommended best practices.

Answers

1. D
2. B
3. D
4. C
5. B
6. B
7. A
8. D
9. A
10. C