Boost up Your Certification Score

SAP
C_CPI_2601
SAP Certified - Integration Developer

For More Information – Visit link below:

https://www.examsboost.com/

Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

Visit us at: https://www.examsboost.com/test/c-cpi-2601

Latest Version: 4.0
Subjects

Topic: 1
Loch Fyne Kippers Posts Kiln Confirmations Into a Standard SAP Core
Business Context
Loch Fyne Kippers, a herring smokehouse in Khonkaen, Thailand, presses and fires clay roof tiles at a single works and posts each completed firing batch into an SAP application that runs its finance and planning. The line supervisors are adamant the SAP core stay standard so annual upgrades remain painless.
Platform Landscape
A developer is wiring the on-premise firing-line system to that SAP application with SAP Integration Suite so each batch confirmation posts automatically, after a check and top-up of any missing data.
Project Objectives
The release must enrich and validate each firing confirmation before it posts, reach the on-premise firing-line system, present the confirmation in the layout the SAP application expects, and actually run at all — without editing the SAP core.
Current Situation
One engineer proposed putting the validation straight inside the SAP application, which the supervisors rejected on upgrade grounds, and separately the confirmations were not posting despite a clean-looking build.
Governance Context
The team also struggled to reach the firing-line system from the cloud and found its confirmation layout did not match what the SAP application would accept.
Your Assignment
You are acting as the integration developer for Loch Fyne Kippers. Work in the shared development tenant. Where an exact technical setting is not specified, apply the general behavior of the platform rather than assuming a specific named option. Your task is to address the four challenges below for the firing confirmation interface, keeping the SAP application standard throughout
1. **CHALLENGE 1 — Keeping the Production-Confirmation Enhancement Off the SAP Application Core**
• Confirm each firing confirmation needs validation and enrichment before it posts to the SAP application.
• Decide where the custom logic should live so the core stays standard and upgradable.
• Reject core modification, copying core code, or removing standard checks.
CHECKPOINTS
• The enrichment runs in transit through the integration layer, with the SAP application core unmodified.
• Future upgrades to the SAP application require no rework of the custom logic.
2. **CHALLENGE 2 — Connecting the Production-Confirmation Flow to the On-Premise SAP Application**
• Contrast the working cloud-to-cloud step with the failing on-premise call to locate the gap.
• Identify the secure connectivity option that bridges the cloud tenant to the on-premise network.
• Do not treat logging, timeouts, or pre-copying data as the fix.
CHECKPOINTS
• The on-premise step completes successfully through the secure link.
• The cloud-to-cloud step is confirmed unaffected, isolating the fault to connectivity.

Visit us at: https://www.examsboost.com/test/c-cpi-2601

3. **CHALLENGE 3 — Transforming Each Production-Confirmation into the Structure the SAP Application Expects**
• Confirm the SAP application rejects each firing confirmation because the structure, not the data, is wrong.
• Identify the step that converts the sender's structure into the receiver's structure.
• Do not treat timeouts or logging as a substitute for transformation.
CHECKPOINTS
• The SAP application accepts a message after it is transformed into the expected structure.
• The transformation preserves the data while changing the shape and field names.
4. **CHALLENGE 4 — Bringing the Production-Confirmation Flow Into Operation and Seeing Its Messages**
• Confirm no firing confirmation messages appear in monitoring though the flow is built and saved.
• Identify that a flow processes messages only once deployed to the runtime.
• Identify that payload visibility depends on the flow's log level.
CHECKPOINTS
• Messages appear in monitoring after the flow is deployed.
• Payloads are viewable once the log level captures message content.

Question: 1

CHALLENGE 1 — Keeping the Production-Confirmation Enhancement Off the SAP Application Core
The team is committed to keeping the SAP application standard, and each firing confirmation needs validation and enrichment before it posts.
Which action aligns with the clean core approach?

A. Modify the SAP application core so the validation runs inline at the posting step. It treats a symptom rather than the actual cause.
B. Copy the standard posting code and adjust the copy. It targets the wrong layer for the fault described here. Nothing about this situation makes it the needed fix.
C. Remove the standard posting checks to make room for the new logic. It leaves the underlying gap exactly where it was.
D. Build the validation as integration logic in SAP Integration Suite, outside the core, so the SAP application stays standard and upgradable.

Answer: D

Explanation:
Clean core keeps enhancements off the core; building the check in the integration layer enriches the firing confirmation in transit while leaving the core standard.

Question: 2

CHALLENGE 1 — Keeping the Production-Confirmation Enhancement Off the SAP Application Core
The team wants to understand why keeping the logic off the core matters.
Which statement best explains the clean core rationale?

Visit us at: https://www.examsboost.com/test/c-cpi-2601

A. Custom logic in the core must be carried forward and retested at every upgrade, so keeping it off the core keeps upgrades cheap and low-risk.
B. Keeping logic off the core makes the firing confirmation process run faster. The evidence points elsewhere than this action.
C. The core cannot run validation logic at all. It is a plausible reflex that does not hold up here. It addresses a different concern than the one at hand.
D. Off-core logic removes the need for any validation. It changes the wrong thing and resolves nothing.

Answer: A

Explanation:
A standard core upgrades cleanly; embedded custom logic couples the enhancement to core internals and raises the cost and risk of every future update.

Question: 3

CHALLENGE 1 — Keeping the Production-Confirmation Enhancement Off the SAP Application Core
A teammate argues the check should sit in the core because that is closest to where the firing confirmation is posted.
What is the best response?

A. Agree, because logic nearest the posting step is always best. It acts beside the layer that actually governs this.
B. Proximity does not justify modifying the core; the enrichment can run in transit in the integration layer while the core stays standard.
C. Agree, because the integration layer cannot enrich a record before posting. It would not survive the validation the team applies.
D. Disagree, and remove the posting step entirely. It reads well but misdiagnoses where the break lies. The scenario's own facts rule this out as the fix.

Answer: B

Explanation:
Closeness to the posting step is not worth the upgrade burden of a core change; the integration layer enriches the record before it posts without touching the core.

Question: 4

CHALLENGE 2 — Connecting the Production-Confirmation Flow to the On-Premise SAP Application
The flow's cloud-to-cloud step works every run, but the step calling the on-premise SAP application always fails with a connection error; credentials are correct.
Why does only the on-premise step fail?

A. The credentials for the on-premise SAP application must be wrong. It treats a symptom rather than the actual cause. It changes the wrong thing and resolves nothing.

B. The flow was never deployed. It targets the wrong layer for the fault described here. Nothing about this situation makes it the needed fix.
C. A cloud flow cannot reach an on-premise system without a secure connectivity bridge, which this step lacks and the cloud-to-cloud step does not need.
D. The log level is too low on that step. It leaves the underlying gap exactly where it was. It is a plausible reflex that does not hold up here.

Answer: C

Explanation:
The on-premise call needs a secure link into the private network; the cloud-to-cloud step works precisely because it needs no such bridge.

Question: 5

CHALLENGE 2 — Connecting the Production-Confirmation Flow to the On-Premise SAP Application
The developer needs the flow to reach the on-premise SAP application.
Which action establishes the path?

A. Raise the log level on the failing step. The evidence points elsewhere than this action. It acts beside the layer that actually governs this. It targets the wrong layer for the fault described here.
B. Add a longer timeout so the step waits for the system. It reads well but misdiagnoses where the break lies. The scenario's own facts rule this out as the fix.
C. Copy the on-premise data into the cloud in advance. It addresses a different concern than the one at hand. It would not survive the validation the team applies.
D. Configure the secure connectivity option that bridges the cloud tenant to the on-premise network, because a cloud flow reaches on-premise systems only through that link.

Answer: D

Explanation:
A secure bridge is the only way a cloud tenant reaches a system inside a private network; configuring it gives the failing step the path it lacks.

Question: 6

CHALLENGE 2 — Connecting the Production-Confirmation Flow to the On-Premise SAP Application
After configuring connectivity, the developer wants to confirm the fix is real and isolated.
How should the flow be validated?

A. Confirm the on-premise step now completes successfully while the cloud-to-cloud step remains unaffected.
B. Confirm the cloud-to-cloud step still works, since that proves the on-premise step does too.
C. Confirm the log now shows the connection error in more detail. It treats a symptom rather than the actual cause.
D. Confirm a copy of the data exists in the cloud. It leaves the underlying gap exactly where it was.

Answer: A

Explanation:
A successful on-premise call through the link, with the cloud step still working, proves the connectivity gap was the fault and is now closed.

Question: 7

CHALLENGE 3 — Transforming Each Production-Confirmation into the Structure the SAP Application Expects
The SAP application rejects every firing confirmation because the structure does not match, though all needed fields are present in the sender's format.
Why are the messages rejected?

A. The receiver's timeout is too short to read the message. It changes the wrong thing and resolves nothing.
B. The sender's structure differs from the structure the SAP application expects, so the message is not understood even though the data is complete.
C. The flow was never deployed. Nothing about this situation makes it the needed fix. The evidence points elsewhere than this action.
D. The log level is too low to process the message. It is a plausible reflex that does not hold up here. It addresses a different concern than the one at hand.

Answer: B

Explanation:
Matching data is not matching structure; the receiver needs the fields arranged and named its way, which the sender's format does not provide.

Question: 8

CHALLENGE 3 — Transforming Each Production-Confirmation into the Structure the SAP Application Expects
The developer must convert each firing confirmation into the structure the SAP application expects.
Which action does this?

A. Increase the receiver's timeout so it accepts the sender's structure. It would not survive the validation the team applies.
B. Raise the flow's log level to record the mismatched fields. It acts beside the layer that actually governs this.
C. Add a message-mapping step that maps the sender's fields onto the receiver's structure before the message is sent.
D. Deploy the flow again to align the structures. It reads well but misdiagnoses where the break lies.

Answer: C

Explanation:
A mapping restructures and renames fields to the receiver's format, converting each message into the exact shape it expects.

Question: 9

CHALLENGE 4 — Bringing the Production-Confirmation Flow Into Operation and Seeing Its Messages
The firing confirmation flow is built and saved with clean design checks, but no messages appear anywhere in monitoring for it, while other flows in the tenant run normally.
Why is nothing processed?

A. The flow's log level is too low to process messages. It leaves the underlying gap exactly where it was.
B. The sender and receiver addresses are wrong. It targets the wrong layer for the fault described here.
C. The flow needs a second receiver to run. The scenario's own facts rule this out as the fix. It treats a symptom rather than the actual cause.
D. The flow was never deployed to the runtime, so it exists only in design time and processes no messages.

Answer: D

Explanation:
A saved flow runs only after deployment; with no messages at all, it was never deployed, which is why nothing is processed.

Question: 10

CHALLENGE 4 — Bringing the Production-Confirmation Flow Into Operation and Seeing Its Messages
The developer must get the firing confirmation flow processing live messages.
Which action achieves this?

A. Deploy the flow to the runtime, because a flow processes messages only once deployed, not while it exists solely in design time.
B. Save the flow a second time in the design workspace. It changes the wrong thing and resolves nothing.
C. Raise the flow's log level so it begins processing. It is a plausible reflex that does not hold up here.
D. Add a second receiver to trigger processing. Nothing about this situation makes it the needed fix. The evidence points elsewhere than this action.

Answer: A

Explanation:
Deployment moves the flow from design time to the runtime, where it actually processes messages, which saving alone never does.

Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

15 USD Discount Coupon Code:

G74JA8UF

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer

