Search documentation Search

Appian

Messaging Best Practices

Appian 7.7

NOTE: This site does not include documentation on the latest release of Appian. Please upgrade to benefit from our newest features. For more information on the latest release of Appian, please see the Appian 7.8 documentation

Use the following recommendations as a guide when designing process models that use Receive Message and/or Send Message events.

Toggle Na

1. Improving Messaging Efficiency
    - Use Conditional Filters
    - Target Processes by ID
    - Allow Public Events
    - Avoid Cascading Messages
    - Avoid Message Loops
    - Minimize the Number of Listeners
    - Pass Minimal Amounts of Data
    - Avoid Enabling the Activate Message Event
2. Starting Processes with Messages
3. Monitoring Messaging Performance

Improving Messaging Efficiency

Use Conditional Filters

There are two simple ways to program whether a given Receive Message event should process a message: by condition or by expression. Conditions a always the most efficient way to process messages. Appian recommends using conditions over expressions in all process designs.

Conditions are combined and evaluated all at once.

- For example, if two Receive Message events are both changed to use three filters each (from a single filter), the following results are expected:
    - When the filters use expressions, the evaluation time approximately triples.
    - When the filters use conditions, the evaluation time remains approximately the same as with a single condition.

Expressions for message filtering are evaluated one-by-one. The use of expression filtering leads to a linear increase in processing time as the number of messages increase.

Target Processes by ID

Targeting specific process instances for messages by their process IDs is much more efficient than sending messages to multiple processes or a process model because only the Receive Message Events for the process ID are scanned, rather than all events in the other processes.

- As of Appian 6.7, you can configure this through the Send Message Event properties.
- Prior to this version, this functionality must be configured within Appian as a customization.

Allow Public Events

Rights-management for events may not be required if your organization has other safeguards in place to prevent unauthorized JMS messages from being sent to or retrieved from Appian.

If additional security is not required, select the **Public Events** checkbox in the Process Model Properties to make the event public.

- The Public Events option disables security checking of events sent from outside the process model.

Avoid Cascading Messages

Synchronization issues can arise if design assumptions are made regarding the order in which messages are received, as messages at a lower tier of the cascade may, on occasion, be sent and delivered earlier than those in an assumed-to-be higher tier.

Where possible, avoid architectures where messages cascade, such as when the sending and subsequent receipt of a message causes multiple other messages to be sent that may trigger other message events.

These process designs create risks which, unless carefully controlled, can cause an exponential growth in the number of messages in the queue and slo the delivery of messages.

Avoid Message Loops

Appian does not recommend passing messages between different processes to send data back and forth in a way that mimics a synchronous protocol.

Messaging is not designed as a synchronous transport component, and there are many race conditions as well as performance implications to using messages in this way.

Best practice dictates that when designing a process, messaging loops should be studiously avoided. In some cases, such designs may be practical only when implemented by an expert in this area, as even extensive testing is unlikely to uncover all possible race conditions and performance issues.

Minimize the Number of Listeners

Once a flow reaches a Receive Message Event, the event is activated. Any time an untargeted message is sent, all active Receive Message Events are scanned. Generally, the more Receive Message Events there are listening for messages, the higher the performance penalty.

If your architecture presents more and more listeners as sub-processes are created, consider passing information to sub-processes by reference.

- This is done by selecting the **Pass as Reference** checkbox for the Input Variable configurations in the Sub-Process Activity of the parent proces

See also: Sub-Process Activity

Where high numbers of message listeners are unavoidable, ensure that you apply as many of the other best practices listed in this document as possibl

Pass Minimal Amounts of Data

The amount of data passed within a message can have significant impact on the amount of time taken to process it. This effect becomes more noticeable, as the number of messages increases.

- Large text strings in particular should be avoided. Keeping messages as small as possible is the best practice.
- It is difficult to provide adequate controls to ensure that future messages remain small, especially when some of the data is input by users.
- As a rule of thumb, messages over 100KB are likely to cause significant performance issues.

Avoid Enabling the Activate Message Event

The Activate Message Event checkbox in the Receive Message Event Setup tab allows your process to listen for events for the life of a process. This car negatively impact messaging performance.

If a Receive Message event needs to listen for more than one message, but doesn't need to listen for the entire life of the process, place the Receive Message Event inside a sub-process.

- Use an exception flow to terminate the sub-process activity when the parent process no longer needs to listen for messages.

Starting Processes with Messages

A common use for messaging is to configure a Receive Message Event on a Start Event to automatically start a process.

When a large number of processes must be started, you can avoid a heavy load on the messaging architecture by using multiple Sub-Process activities to launch multiple asynchronous processes when a single message is received.

- Keep in mind that Sub-Process instances start on the same execution engine as the parent.

When you have three (or more) sub-processes to start, use at least three message-listening processes. Distribute the launching duties among the listening processes as evenly as possible, taking care to also divide up any processes that are expected to remain active for long periods of time. This avoids a potential load imbalance between the three (standard) execution engines.

Processes do not have to be related to each other, in order to be grouped together by a parent launch process.

Avoid using sub-process activities to start processes based on query rules when each new item in a data store starts a new process. Use process-to-process messaging instead.

Monitoring Messaging Performance

Review the Process Execution performance log files to monitor messaging performance.

See also: Monitoring Performance and Usage.

User Guides by Role

- Designer
- Developer
- Web Admin
- Server Admin (On-Premise Only)

Tutorials

- Records
- Interfaces
- Process

Release Information

Release Notes

Installation

Migration

System Requirements

Hotfixes

Release History

Other

STAR Methodology

Best Practices

Glossary

APIs