Dialogue System Support

Contents

Chapter 1: Introduction

This manual describes how to use the Dialogue System for Unity with Quest Machine.

Quest Machine's Dialogue System integration provides these features:

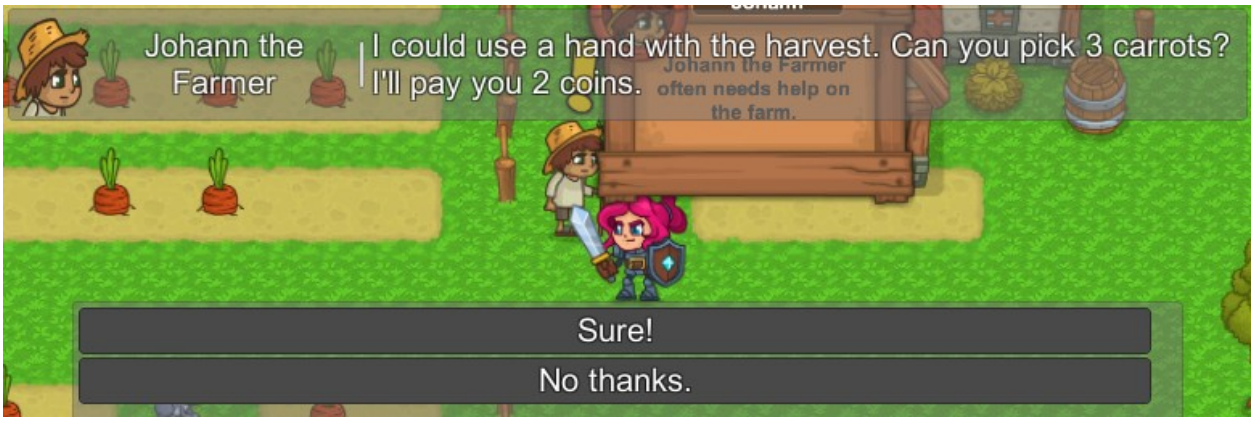
- Configure quests to play Dialogue System conversations.
- Use Quest Machine tags (e.g., counter values) in Dialogue System text.
- Control Quest Machine quests within the Dialogue System.
- Check and control Dialogue System data in quests.

How to Import the Dialogue System Integration

To enable Dialogue System integration, import these two packages:

- **Plugins ► Pixel Crushers ► Common ► Third Party Support ► Dialogue System Support**
- **Plugins ► Pixel Crushers ► Quest Machine ► Third Party Support ► Dialogue System Support**

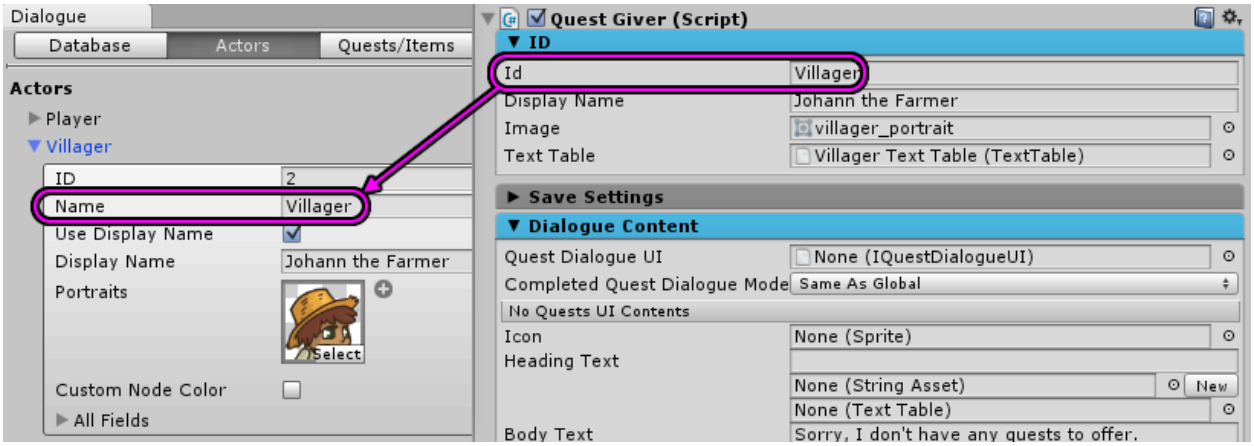
Chapter 2: Quest Machine Dialogue System Demo



The Dialogue System integration package's demo scene contains these quests:

- **Pesky Rabbits**: (Villager) A version of the original Pesky Rabbits quest that plays a Dialogue System conversation instead of using Quest Machine's dialogue UI.
- **Harvest Carrots**: (Villager) A version of the original Harvest Carrots quest that plays a Dialogue System conversation instead of using Quest Machine's dialogue UI.
- *Generated Quests*: (Knight, Pirate) Demonstrates procedurally generated quests with procedurally generated conversation.

In the dialogue database, the actors' Names correspond to the Quest Givers' IDs, and the quests' Names correspond to the quests' IDs. Actors and quests also use Display Names:

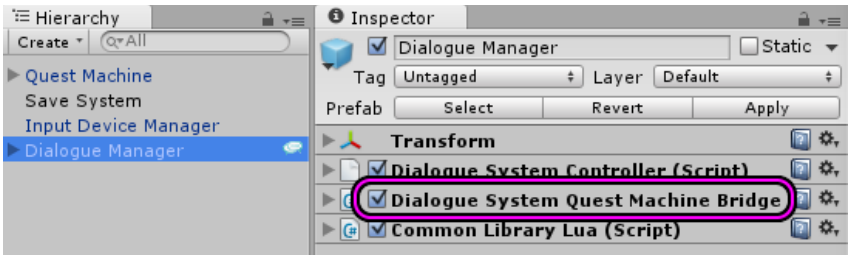


The Villager's quests have been changed so their corresponding conversations update the "Talk to NPC" quest node, instead of the quest node listening for a "Discussed Quest" message from Quest Machine. To do this, the conversations use special Quest Machine Lua functions, covered in Chapter 7.

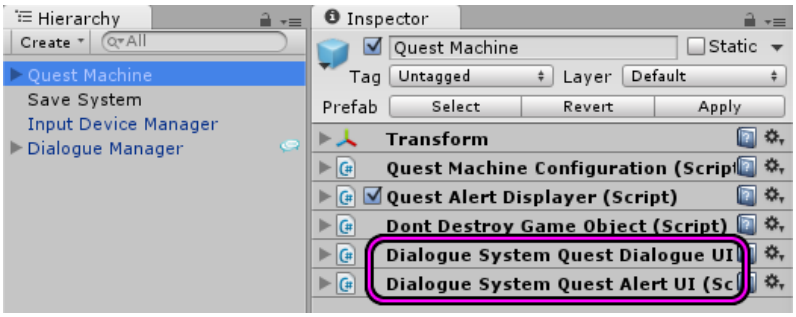
Chapter 3: Set Up Dialogue System Integration

To configure your scene:

1. Add the **Dialogue Manager** prefab. Deactivate or remove the quest window and quest tracker children if you're going to use Quest Machine's quest journal UI and HUD instead.
2. Add a **Dialogue System Quest Machine Bridge** component to the Dialogue Manager.

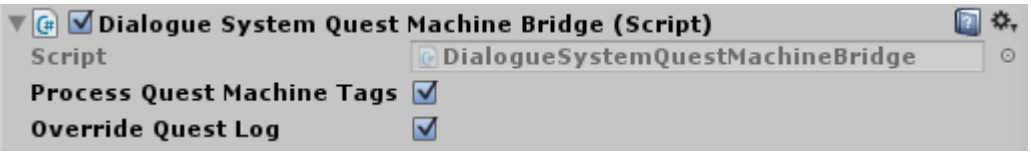


3. Add the **Quest Machine** prefab. (Optional: Add Input Device Manager and/or Save System.)
4. Add a **Dialogue System Quest Dialogue UI** component to the Quest Machine GameObject. By adding the new content type Dialogue System Conversation Quest Content to your quests, you can show Dialogue System conversations instead of Quest Machine's regular quest dialogue UI.
5. If you want to show Quest Machine alerts through your Dialogue System's dialogue UI, add a **Dialogue System Quest Alert UI** component.



6. Quest givers will show Dialogue System conversations for quests if you've added Dialogue System Conversation Quest Content to the quest. If you also want to show a Dialogue System conversation when no quests are available, replace the quest giver's Quest Giver component with a **Dialogue System Quest Giver** component. Then you can select a Dialogue System conversation in the Dialogue Content > No Quests UI Contents section.

Chapter 4: Dialogue System Quest Machine Bridge



The Dialogue System Quest Machine Bridge should be added to your Dialogue Manager. It performs three functions:

1. Replaces Quest Machine tags, such as {#*counter*}, in dialogue text.
2. Redirects the Dialogue System's QuestLog class to refer to Quest Machine quests.
3. Adds Lua functions to the Dialogue System that you can use to control Quest Machine quests.

To enable the replacement of Quest Machine tags in dialogue text, tick **Process Quest Machine Tags**. You can use Quest Machine tags in the Quest Giver's Dialogue Content and in Dialogue System conversations.

To redirect the methods in the Dialogue System's QuestLog class to Quest Machine, tick **Override Quest Log**. This will also redirect the Lua functions CurrentQuestState(), CurrentQuestEntryState(), SetQuestState(), and SetQuestEntryState() to Quest Machine. Note: CurrentQuestEntryState() and SetQuestEntryState() operate on quest entry numbers, which will correspond to the order of the Quest Machine quest's Node Order For UIs. You can see Node Order For UIs at the bottom of the quest's main info.

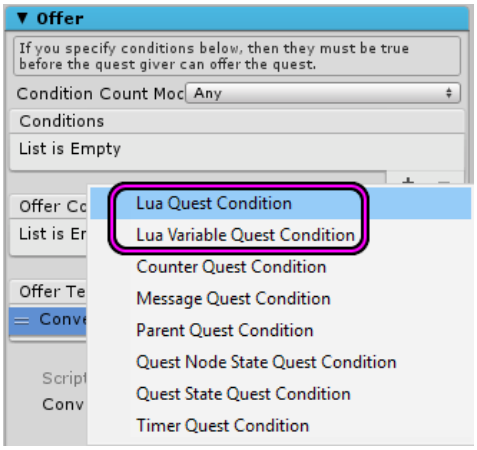
If **Override Quest Log** is *not* ticked, these Lua functions will not be overridden; instead, it will add unique function names GetQuestState() and SetQMQuestState() to access Quest Machine quests. These Lua functions are described in greater details in Chapter 7.

When the Dialogue System receives a message to update its quest tracker HUD, the bridge will forward the message to Quest Machine to refresh its UIs, too.

Chapter 5: Access Dialogue System in Quests

Dialogue System Quest Conditions

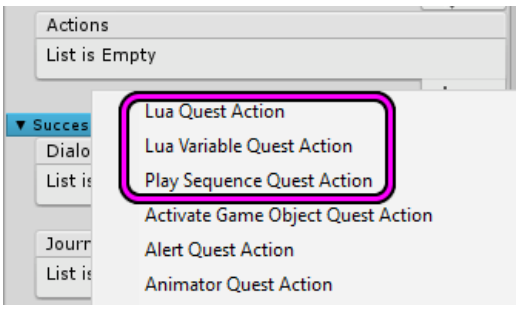
The integration package adds two quest condition types: **Lua Quest Condition** and **Lua Variable Quest Condition**.



Lua Quest Condition checks an arbitrary Lua condition, while Lua Variable Quest Condition checks the value of a Dialogue System Lua variable. Both check on a frequency that you specify (e.g., every three seconds).

Dialogue System Quest Actions

The integration package adds three quest action types: **Lua Quest Action**, **Lua Variable Quest Action**, and **Play Sequence Quest Action**.



- **Lua** Quest Action runs arbitrary Lua code
- **Lua Variable** Quest Action sets a Dialogue System Lua variable.
- **Play Sequence** Quest Action plays a Dialogue System sequence.
- **Bark** Quest Action plays a Dialogue System bark. (not pictured)

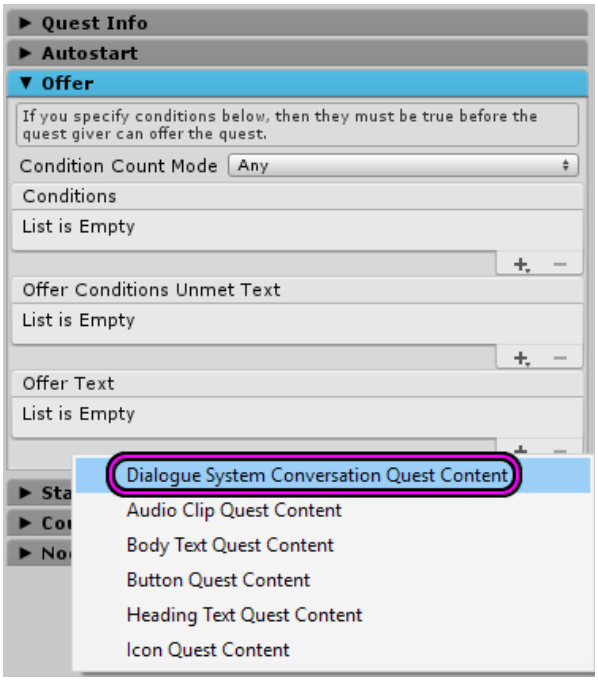
Dialogue System in Quest Content

If you want to use Dialogue System markup tags such as [var=*varName*], use **Dialogue System Text Quest Content** in place of **Body Text Quest Content**.

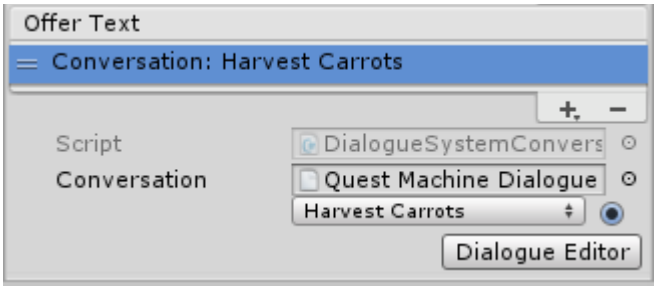
Chapter 6: Quest Conversations

Dialogue System Conversation Quest Content

To configure a quest to run a Dialogue System conversation, add a **Dialogue System Conversation Quest Content**:



You'll typically add this to the quest's Offer Text section as well as the Dialogue Text sections of the main quest's Active and Successful states (and other states if your conversation contains content for them). This content type lets you specify a Dialogue System conversation to play. You can also click the Dialogue Editor button to open the Dialogue Editor window on that conversation's START node.



If a state's content contains a Dialogue System Conversation Quest Content, any other content (such as

Heading Text, Body Text, etc.) will not be shown. If there is no Dialogue System Conversation Quest Content, Quest Machine will use the default UI configured on the Quest Machine Configuration component.

Quest Machine Tags

Conversations will recognize Quest Machine tags. For example, in the following Dialogue Text:

> "I'm {QUESTGIVER}. Please harvest {#>carrots} carrots."

The tag {QUESTGIVER} will be replaced by the quester giver's display name (e.g., "Johann the Farmer") and {#>carrots} will be replaced by the maximum value of the quest's carrots counter.

When the Dialogue System Conversation Quest Content starts a conversation, it will attempt to set two additional tags: QuestID and QuesterID. If you prefer, you can set these manually by adding them as custom fields in your conversation's All Fields section. They should match the IDs of the quest and the quester (player), respectively.

If you don't want conversations to process Quest Machine tags, inspect the Dialogue System Quest Machine Bridge component and untick Process Quest Machine Tags.

Quest Machine Dialogue System Variables

When Quest Machine starts a Dialogue System conversation, it will set these Dialogue System variables:

- Variable["QUESTID"]: Quest Machine quest ID.
- Variable["QUESTGIVER"]: Quest Giver Display Name.
- Variable["QUESTGIVERID"]: Quest Giver ID.
- Variable["QUESTER"]: Quester (e.g, player) Display Name as set in its Quest Journal.
- Variable["QUESTERID"]: Quester ID as set in its Quest Journal.

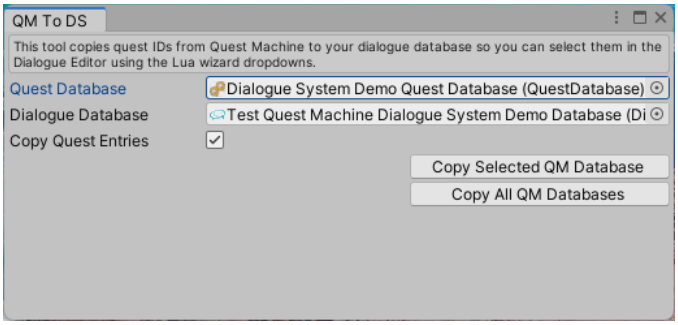
Dialogue System Quest Functions Control Quest Machine

The Dialogue System Quest Machine Bridge's **Override Quest Log** checkbox will redirect the Dialogue System's quest functions to control Quest Machine quests instead of the Dialogue System's own quest subsystem. This includes the Dialogue Editor's Lua wizards / Lua functions as well as the QuestLog script class. If you don't want to redirect quest functions to Quest Machine, untick Override Quest Log.

The Lua wizards' dropdowns will only show quest names that are defined in your dialogue database, so you'll need to use the QM To DS Tool described below to set up the Lua wizards.

QM To DS Tool

Use the QM To DS Tool to populate your dialogue database with a quest database's quest IDs. This will allow you to access them in the Dialogue Editor's Lua wizard. To open the tool, select menu item **Tools → Pixel Crushers → Quest Machine → Third Party → Dialogue System → Quest DB To Dialogue DB**.



Click **Copy Selected QM Database** to copy the quest IDs in the selected quest database into the dialogue database, or click **Copy All QM Databases** to copy all quest IDs in all quest databases assigned to the current scene's Quest Machine Configuration component.

If you tick Copy Quest Entries, it will copy each quest's quest nodes to corresponding quest entries in the dialogue database. This will make it easier to select them in the "…" dropdowns for Quest Entry > CurrentQuestEntryState and Quest Entry > SetQuestEntryState.

Accessing Quest Machine in Conversations

In your conversations, you can use a number of Quest Machine-specific Lua functions and sequencer commands. The Lua functions are described in Chapter 7 below. The sequencer commands are described in Chapter 8.

Chapter 7: Lua Functions

The Dialogue System Quest Machine Bridge adds the Lua functions below. If your game has only one player/quester, you can use the simpler **Quests On Default Player** functions. If your game has multiple questers, use the **Quests on Any Quester** functions.

Lua Function List

Function	Description
Quests on Default Player	
CanOfferQuest(giverID, questID)	Checks if the quest giver can offer a quest to the player.
GiveQuest(giverID, questID)	Tells the quest giver with *giverID* to give the quest with *questID* to the player.
GiveAllQuests(giverID)	Tells the quest giver with *giverID* to give all of its quests to the player.
HasQuest(questID)	Returns true if the player has a quest with *questID*.
CurrentQuestState(questID)	Returns the state* of a quest in the player's journal
SetQuestState(questID, state)	Sets the state* of a quest in the player's journal
GetQuestNodeState(questID, nodeID)	Gets the state* of a quest node in the player's journal
SetQuestNodeState(questID, nodeID, state)	Sets the state* of a quest node in the player's journal
GetQuestNodeNumberState(questID, nodeNumber)	Gets the state* of a quest node (specifies the node *number*, not its ID)
SetQuestNodeNumberState(questID, nodeNumber, state)	Gets the state* of a quest node (specifies the node *number*, not its ID)
GetQuestCounterValue(questID, counterName)	Gets the value of a quest counter
SetQuestCounterValue(questID, counterName, value)	Sets the value of a quest counter
Quests on Any Quester	
HasOfferableOrActiveQuest(giverID, questerID)	Checks if quest giver has a quest to offer or update.
CanOfferQuestToQuester(giverID, questID, questerID)	Checks if the quest giver can give the quest to a quester.
GiveQuestToQuester(giverID, questID, questerID)	Tells a quest giver to give the quest to a quester.
GiveAllQuestsToQuester(giverID, questerID)	Tells a quest giver to give all of its quests to a quester.
QuesterHasQuest(questID, questerID)	Returns true if a quester has a quest.
GetQuesterQuestState(questID, questerID)	Returns the state* of a quest
SetQuesterQuestState(questID, state, questerID)	Sets the state* of a quest
GetQuesterQuestNodeState(questID, nodeID, questerID)	Gets the state* of a quest node
SetQuesterQuestNodeState(questID, nodeID, state, questerID)	Sets the state* of a quest node
GetQuesterQuestNodeNumberState(questID, nodeNumber, questerID)	Gets the state* of a quest node (specifies the node *number*, not its ID)
SetQuesterQuestNodeNumberState(questID, nodeNumber, state, questerID)	Gets the state* of a quest node (specifies the node *number*, not its ID)
GetQuesterQuestCounterValue(questID, counterName,	Gets the value of a quest counter

questerID)	
SetQuesterQuestCounterValue(questID, counterName, value, questerID)	Sets the value of a quest counter
Message System	
SendMessageSystem(message, parameter)	Sends a message to the Message System
SendMessageSystemString(message, parameter, string)	Sends a message with a string value
SendMessageSystemInt(message, parameter, int)	Sends a message with an integer value
Misc	
SetQuestIndicator(questID, entityID, state)	Sets an entity's quest indicator for a specified quest**
StartQuestListConversation(questGiverID)	Starts generated conversation showing list of quests available to talk about. Blank string forquestGiverID uses current conversant.
StartQuestConversation(questGiverID, questID, questerID)	Starts a specific quest conversation. If questGiverID is blank, uses the current conversant. If questerID is blank, uses the default Quest Journal. If questID is blank, uses the first applicable quest.
ShowQuestAlert(message)	Shows a message through the quest alert UI. If you've added a DialogueSystemQuestAlertUI to the Quest Machine GameObject, it will look for a UnityUIQuestAlertUI in the Quest Machine's hierarchy to use a Quest Machine alert UI instead of Dialogue System.
Procedural Generation	
HasGeneratedQuest(questGiverID, questerID)	Checks if a quest giver has an offerable or active procedurally-generated quest. If questGiverID is a blank string, uses the current conversant. If questerID is blank, uses the default Quest Journal.
ShowGeneratedQuestConversation(questGiverID, questerID)	Switches dialogue to a conversation in the quest giver's procedurally-generated quest. If questGiverID is a blank string, uses the current conversant. If questerID is blank, uses the default Quest Journal.

* Dialogue System quest state strings:

Dialogue System Quest State String	Description
"unassigned"	Quest is in waiting to start or disabled state, or node is in inactive state
"active"	Quest/node is in active state
"success"	Quest is in successful state, or node is true
"failure"	Quest is in failed state
"abandoned"	Quest is in abandoned state

** Quest indicator state strings:

Indicator State String (any case, no spaces)
“None”
“OfferDisabled”
“TalkDisabled”
“InteractDisabled”
“Offer”
“Talk”
“Interact”
“Custom0” - “Custom9”

Override Quest Log Unticked

If the Bridge component’s Override Quest Log checkbox is unticked, the Quest Machine Lua function to get and set a quest’s state will be GetQuestState(), and SetQMQuestState() so as not to conflict with the Dialogue System’s quest Lua functions.

Lua Function Examples

```
GiveQuest("Villager", "harvestCarrots")
SendMessageSystemInt("Give", "Coin", 5)
```

Quest ID Variable

Quest Machine conversations will set a Dialogue System variable named “QUESTID”. You can use this variable in Lua functions, such as:

```
GetQuestState(Variable["QUESTID"])
```

Chapter 8: Sequencer Commands

The Dialogue System integration adds the following sequencer command:

Sequencer Command List

Sequencer Command	Description
Spawner(subject, operation)	Controls a Spawner. • subject: A GameObject with a Spawner. Default: speaker. • Operation: start | stop | despawn

Sequencer Command Example

```
Spawner(WolfSpawner, start)
```

Chapter 9: Save System

To tie the Dialogue System to Quest Machine's Save System, add a **Dialogue System Saver** component to the Dialogue Manager. In the future, the Dialogue System will replace its own save subsystem with the Pixel Crushers Common Library Save System, so this integration package currently doesn't provide a component to go the other way, saving Quest Machine using the Dialogue System's save subsystem.

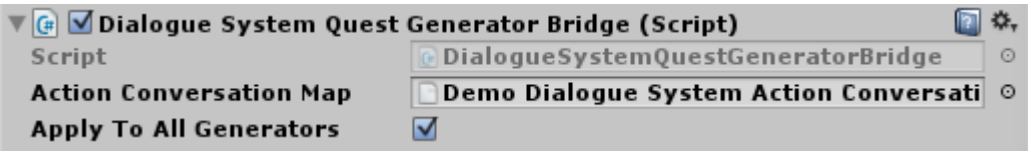
Chapter 10: Quest Generation

To set up a quest generator NPC to use Dialogue System conversations:

1. Configure the quest generator entity as normal in Quest Machine:
   - Add a Quest Giver component.
   - Add a Quest Generator Entity component.
   - Add reward systems.
2. Optional: Add an **Override Actor Name**, and tick **Use Localized Name In Database**. Set **Override Name** to the name of an actor in your dialogue database.
3. Add a **Dialogue System Quest Generator Bridge** component. You can add this individually to NPCs, or add one component (e.g., to the Dialogue Manager) and tick **Apply To All Generators** to make it apply to all quest generator NPCs.

To create your own quests through code, use QuestBuilderWithDialogueSystem, which is a subclass of QuestBuilder that adds methods for Dialogue System content. See this section for more information.

Dialogue System Quest Generator Bridge



The Dialogue System Quest Generator Bridge allows quest content to refer to Dialogue System conversations. It relies on an asset called a *Dialogue System Action Conversation Map*, described below.

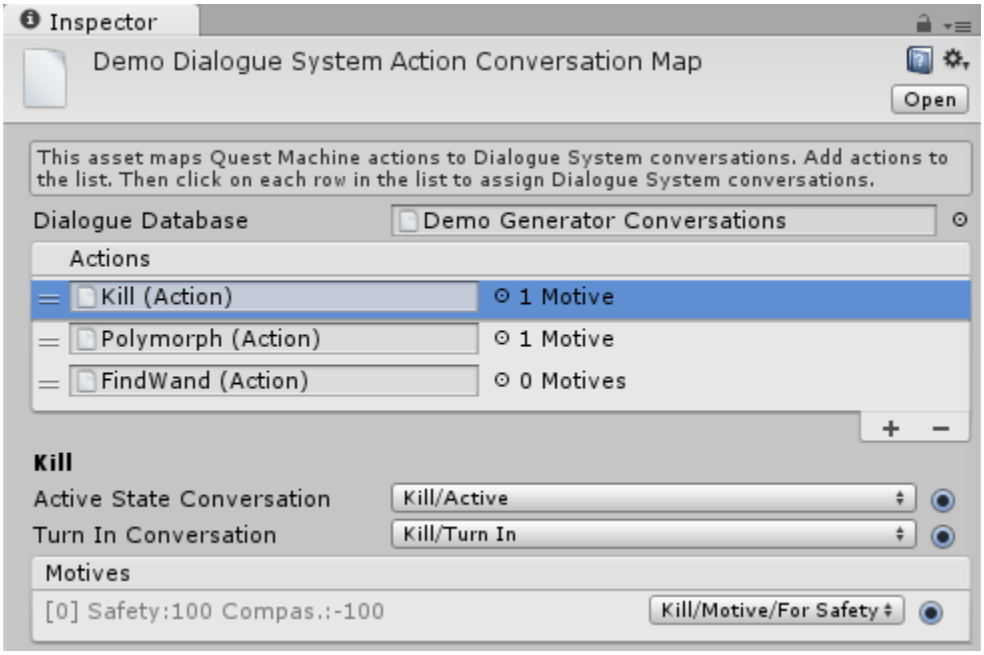
Dialogue System Asset Conversation Map

The Dialogue System Quest Generator Bridge assembles dialogue content from multiple conversations. Take for example a procedurally-generated kill quest. This quest will have at least three conversation states:

1. Offering the quest to the player,
2. Talking to the player while the quest is still active (i.e., kill count not met yet), and
3. Turning in the quest once the player has met the required kill count.

Each of these will be a separate conversation. This allows Quest Machine to efficiently assemble dialogue even when quests involve multiple steps.

The *Dialogue System Asset Conversation Map* specifies how Actions relate to Dialogue System conversations. To create a Dialogue System Conversation Map, select **Assets → Create → Pixel Crushers → Quest Machine → Generator → Dialogue System Action Conversation Map**.

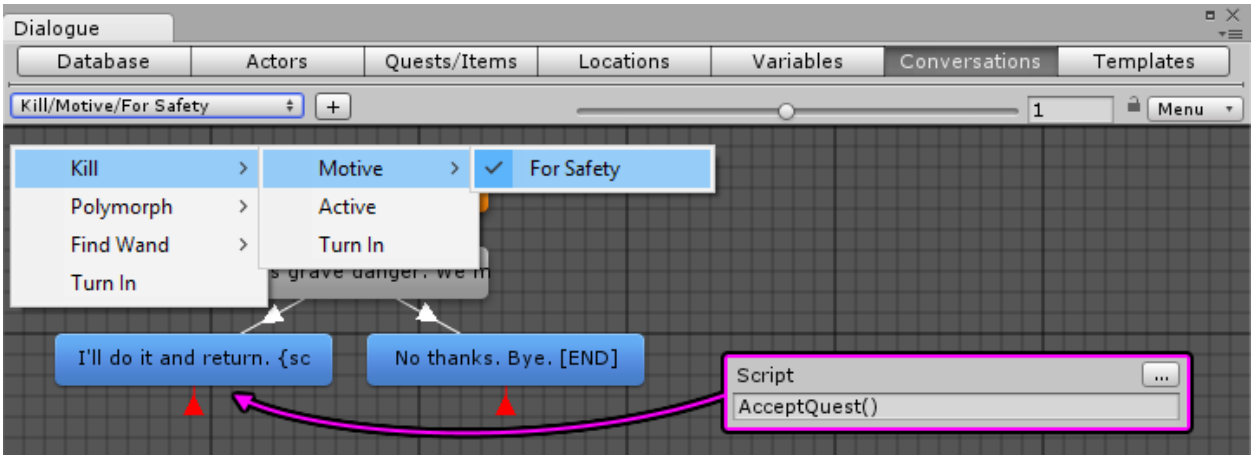


Inspect the asset, and assign actions to the **Actions** list. Each row in the list will report the number of motives defined for the action. If the row has an asterisk (*), this means some conversations still need to be specified for that action.

Click on a row to assign conversations to the action.

- In the **Motives** list, specify the conversation to play when offering the quest to the player with each motive. For example, the Kill action might have different motives for safety, revenge, and prestige. The conversations to offer the quest to the player may be vastly different for each motive.
  - Conversations can use a number of special tags that are listed later in this chapter.
  - On the dialogue entry node in which the player accepts the quest, set the **Script** field to `AcceptQuest()`. This tells Quest Machine to add the quest to the player's journal.
- Set the **Active State Conversation**. This conversation plays when the quest node representing this action is active. You can use it to remind the player what to do, or urge them to complete the task faster.
- Set the **Turn In Conversation**. This conversation plays when the quest's "return to NPC" node is active. This is typically a thank-you conversation.

The screenshot below shows the demo's "For Safety" motive/offer conversation for kill quests:



In the demo, these conversations are in an extra database named "Demo Generator Conversations". The demo scene loads it using an Extra Databases component. The demo's main database and this database have been run through the Dialogue System's Unique ID Tool to ensure that their internal ID numbers don't conflict when both databases are loaded.

Special Tags Available In Generated Quest Conversations

You can use the following tags in conversations assigned to the conversation map:

Tag	Example	Description
{DOMAIN}	The Forest	Location of the action's target
{ACTION}	Kill	Action's display name
{TARGET}	Orc	Target type's display name
{TARGETS}	Orcs	Target type's plural name
{TARGETDESCRIPTOR}	5 Orcs	Action counter goal and target
{COUNTERGOAL}	5	Counter goal
{REWARD}	I'll give you: 10 coins.	Complete reward text, or blank if no reward.

Conversation Tag Example

The "Kill / Motive / For Safety" conversation uses several of these tags:

"There's grave danger. We must step forward for the safety of our people. Enter {DOMAIN} and {kill} {TARGETDESCRIPTOR}. {REWARD}"

QuestBuilderWithDialogueSystem Class

QuestBuilderWithDialogueSystem is a subclass of QuestBuilder that adds two methods:

- **CreateConversationContent**: Creates a DialogueSystemConversationQuestContent object that can be added to the quest using AddContent.
- **CreateDialogueSystemTextContent**: Creates a DialogueSystemTextQuestContent object that can be added to the quest using AddContent.

You can use this class to create your own quests in code if you like.

Quest Dialogue From Other Conversations

To jump to a procedurally-generated quest's conversation from another conversation, use the `ShowGeneratedQuestDialogue()` Lua function in the other conversation's Script field. To check if there is a procedurally-generated conversation to jump to, you can use the `HasGeneratedQuest()` Lua function in the Conditions field.

Final Advice

Pick apart the demo scene to see how it works. If you have any questions, we're here to help! Please visit the forum or email support@pixelcrushers.com any time!