



articy:draft Support

Copyright © Pixel Crushers. All rights reserved.

articy:draft © articy Software GmbH & Co.

Contents

Chapter 1: Introduction.....	4
Overview.....	4
Articy Templates for Quest Machine.....	4
How To Set Up articy:draft Integration.....	9
How to Import Quests.....	9
Runtime Features.....	10
Limitations.....	10
We're Here To Help!.....	10

Chapter 1: Introduction

This manual describes how to import quests from articy:draft to Quest Machine. This integration *requires* Nevigo's [articy:draft plugin for Unity](#).

Overview

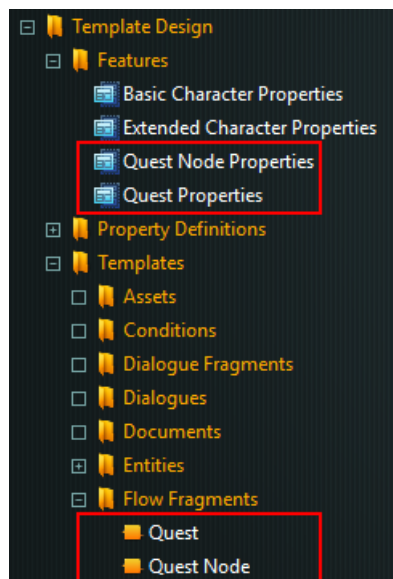
This is the high-level process to import articy:draft into Quest Machine:

1. Create an articy:draft project with specially-defined templates for quests and quest nodes.
2. Import Nevigo's articy:draft plugin for Unity into your Unity project.
3. Export your articy:draft project to Unity.
4. Import Quest Machine's Articy Support.
5. Use a menu item to automatically generate Quest Machine scripts for your articy data.
6. Import your quests from articy data into Quest Machine assets.

The first step is to set up your articy:draft project templates.

Articy Templates for Quest Machine

Quest Machine can import specially-defined flow fragments as quests. Quest Machine requires a specific format for your quests and quest nodes. You will need to set up two property sets: **Quest Properties** & **Quest Node Properties**, and two templates: **Quest** & **Quest Node**:

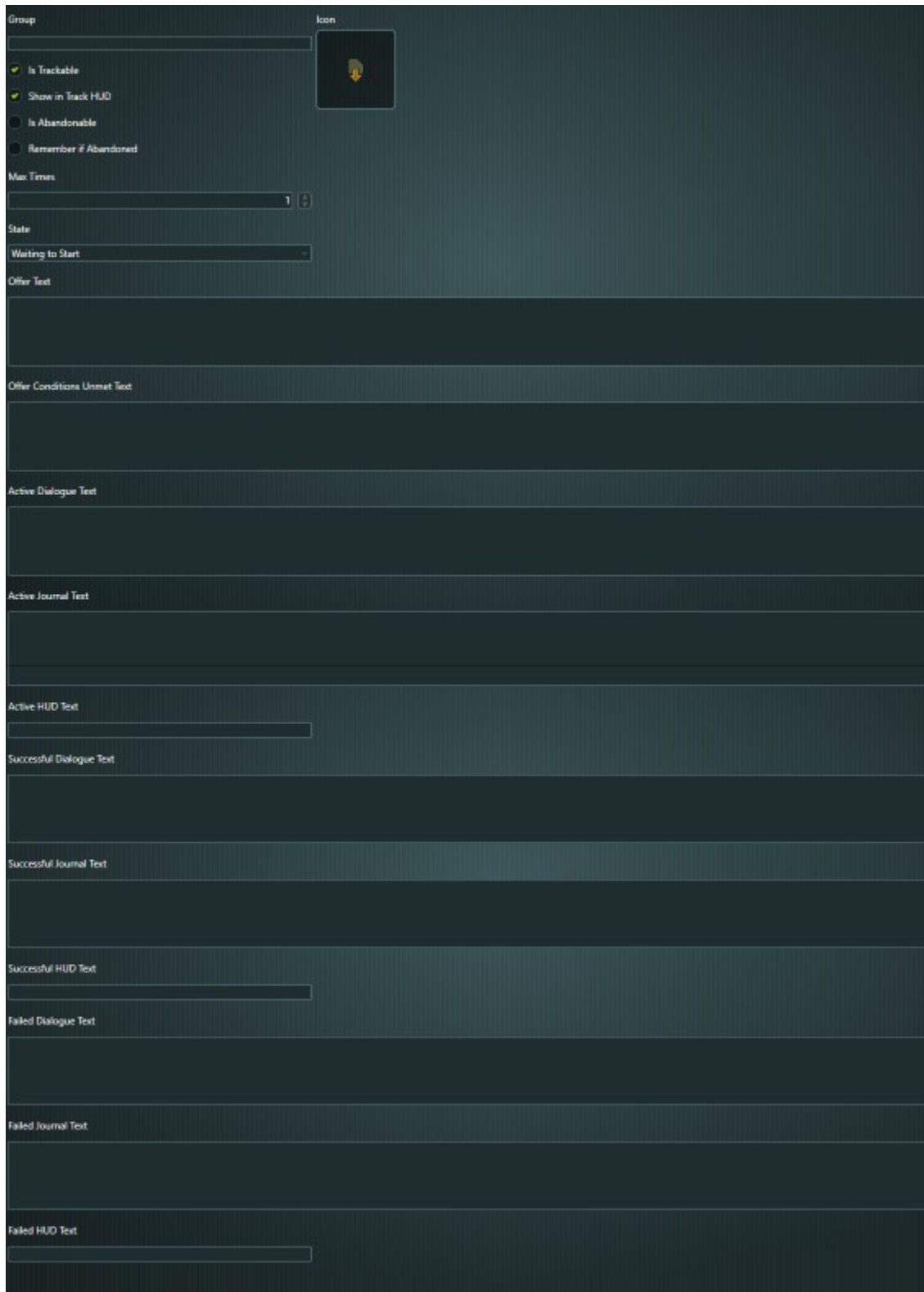


Quest Properties

Quest Machine will import the quest's Display Name as the quest's title and Technical Name as the quest's ID. The remaining quest information will be in a property set called Quest Properties. The Quest Properties' Technical Name must be "**QuestFeature**", with exact spelling and capitalization, and no space between the words. This property set must contain these properties:

Technical Name	Type	Description
group	String	Optional quest group
icon	Slot (Image)	Optional quest icon
isTrackable	Boolean	Player can toggle HUD tracking
showInTrackHUD	Boolean	Show in track HUD
isAbandonable	Boolean	Player can abandon quest
rememberIfAbandoned	Boolean	Keep a record if player abandoned quest
maxTimes	Int (default: 1)	Times the player can do the quest
state	Dropdown with values: <i>Waiting to Start, Active, Successful, Failed, Abandoned, Disabled</i> (default: <i>Waiting to Start</i>)	Initial quest state
offerText	String	Dialogue text to show when offering the quest
offerConditionsUnmetText	String	Dialogue text to show when the player hasn't met the offer conditions
activeDialogueText	String	Dialogue text to show when the quest is active
activeJournalText	String	Text to show in the journal when the quest is active
activeHUDText	String	Text to show in the HUD when the quest is active
successfulDialogueText	String	Dialogue text to show when the quest has been successfully completed
successfulJournalText	String	Text to show in the journal when the quest has been successfully completed
successfulHUDText	String	Text to show in the HUD when the quest has been successfully completed
failedDialogueText	String	Dialogue text to show when the quest has failed
failedJournalText	String	Text to show in the journal when the quest has failed
failedHUDText	String	Text to show in the HUD when the quest has failed.

Below is an example configuration of Quest Properties (QuestFeature):



The image shows a configuration window for Quest Properties (QuestFeature) with a dark theme. The window is divided into several sections:

- Group:** A text input field.
- Icon:** A square icon slot containing a gold skull icon.
- Properties:** A list of checkboxes:
 - Is Trackable
 - Show in Track HUD
 - Is Abandonable
 - Remember if Abandoned
- Max Times:** A numeric input field with the value '1' and a dropdown arrow.
- State:** A dropdown menu with 'Waiting to Start' selected.
- Offer Text:** A large empty text area.
- Offer Conditions Unmet Text:** A large empty text area.
- Active Dialogue Text:** A large empty text area.
- Active Journal Text:** A large empty text area.
- Active HUD Text:** A small text input field.
- Successful Dialogue Text:** A large empty text area.
- Successful Journal Text:** A large empty text area.
- Successful HUD Text:** A small text input field.
- Failed Dialogue Text:** A large empty text area.
- Failed Journal Text:** A large empty text area.
- Failed HUD Text:** A small text input field.

Quest Template

The Quest template's Technical Name must be "**QuestTemplate**" with exact spelling and capitalization. It should use the Quest Properties set described above.

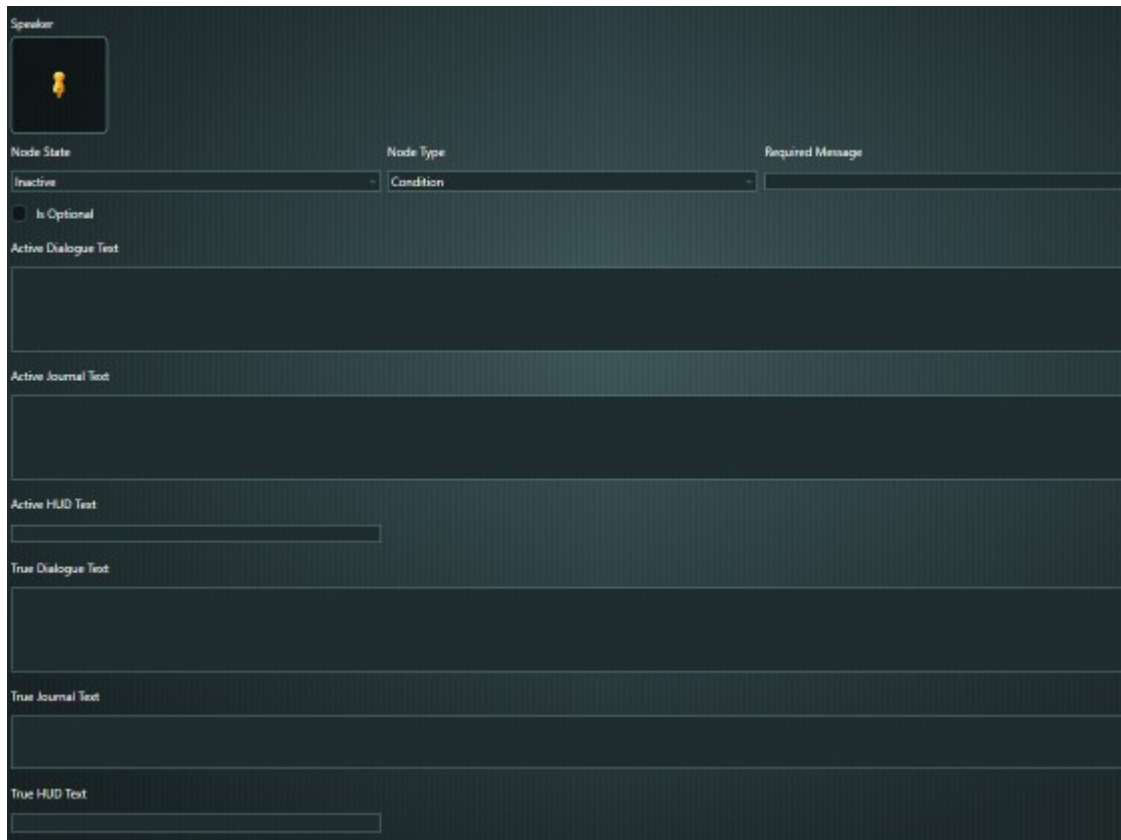
Quest Node Properties

Each Quest Machine quest node will be represented by a flow fragment inside the main quest flow fragment. Quest Machine will import the Display Name as the quest node's internal name and Technical Name as the quest node's ID. The remaining quest information will be in a property set called Quest Node Properties. The Quest Node Properties' Technical Name must be "**QuestNodeFeature**", with exact spelling and capitalization, and no space between the words. This property set must contain these properties:

Technical Name	Type	Description
speaker	Slot (Entity)	Optional speaker for this node's dialogue text
nodeState	Dropdown with values: <i>Inactive</i> , <i>Active</i> , <i>True</i> (default: <i>Inactive</i>)	Current node state
nodeType	Dropdown with values: Condition, Passthrough, Success, Failure	Indicates the node type
requiredMessage	String	Optional; if set, the Message System message that sets this node true. If a parameter is required, use colon (:) to separate the parameter from the message. Example: "Discuss Quest:KillRats"
isOptional	Boolean	Getting this node into the true state is optional to progress the quest
activeDialogueText	String	Dialogue text to show when the node is active
activeJournalText	String	Text to show in the journal when the node is active
activeHUDText	String	Text to show in the HUD when the node is active
trueDialogueText	String	Dialogue text to show when the node's state is true
trueJournalText	String	Text to show in the journal when the node's state is true
trueHUDText	String	Text to show in the HUD when the node's state is true

Conditions on input pins will be imported into Quest Machine as a condition on the preceding node. Expressions on output pins will be imported into Quest Machine as actions that are run when the node becomes true.

Below is an example configuration of Quest Node Properties (QuestNodeFeature):



Quest Node Templates

The Quest Node template's Technical Name must be “**QuestNodeTemplate**” with exact spelling and capitalization. It should use the Quest Node Properties set described above.

Variables

To track values such as kill counts, define global variables. You may prefer to define a variable set for each quest, as in the example below:

Variable set	Name	Description	Type	Default value
Kill3Orcs	canDoOrcQuest	Set true when the player can be offered the Orc quest.	Boolean	False
Kill3Orcs	isAnOrcKiller	Quest node's output pin sets this true.	Boolean	False
Kill3Orcs	orcsKilled	Counts # orcs killed for Kill 3 Orcs quest.	Integer	0
Kill3Orcs	returnedToNPC	True when the player has returned after kill 3 Orcs.	Boolean	False
StopSelfDestruct	clearedDestructCommand	Set true when the player has cleared the self-destruct.	Boolean	False
StopSelfDestruct	returnedToCaptain	Set true when player has returned; watched by input pin.	Boolean	False

General Notes

When defining text fields in property sets, tick the **Multi-language** checkbox.

When exporting to Unity, you *must* export localization support (xlsx files).

How To Set Up articy:draft Integration

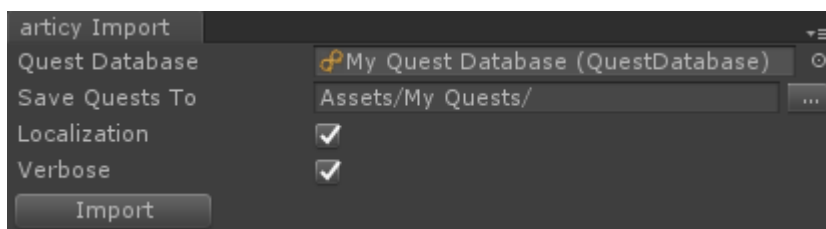
To enable articy:draft / Quest Machine integration in your Unity project:

1. Import Nevigo's articy:draft plugin for Unity and import your articy project into Unity.
2. Import this package:
 - **Plugins ▶ Pixel Crushers ▶ Quest Machine ▶ Third Party Support ▶ Articy Support**
3. Select menu item **Tools → Pixel Crushers → Quest Machine → Import → articy:draft → Generate Articy Project-Specific Quest Machine Scripts**. This menu item will only be available if Unity is able to compile your project's source code. If there are compiler errors, resolve them first.

This will generate scripts in the folder **Pixel Crushers ▶ Quest Machine ▶ Third Party Support ▶ Articy Support ▶ Generated**. If you change the Technical Name of your articy project, re-run this menu item.

How to Import Quests

To create Quest Machine quest assets, select menu item **Tools → Pixel Crushers → Quest Machine → Import → articy:draft → Import Articy Quests to Quest Machine....** This menu item will only be available after you run the Generic Articy Project-Specific Quest Machine Scripts menu item. It will open the **articy Import** window:



1. Optionally assign a quest database. Imported quest assets will automatically be added to this database.
2. Specify the folder where quest assets will be created.
3. If your project uses more than one language tick **Localization**. (See [Localization](#).)
4. For verbose logs of the import process, tick **Verbose**.
5. Click **Import** to begin the import process.

Runtime Features

Quest Machine's articy:draft support runs on top of Nevigo's articy:draft plugin for Unity. If you change articy data at runtime, such as setting an articy global variable value, call the static function `QuestMachineArticy.ArticyStateChanged()`. This will inform quests to check their current conditions.

Limitations

Quest group names are currently not localized. (This is on the roadmap.)

Dialogues and dialogue fragments are currently imported as passthrough nodes.

We're Here To Help!

We're here to help! If you get stuck or have any questions, please contact us any time at support@pixelcrushers.com or visit <http://pixelcrushers.com>.

We do our very best to reply to all emails within 24 hours. If you haven't received a reply within 24 hours, please check your spam folder.