Xray Test Management for Jira Cloud: Comprehensive Guide

Including UAT, Manual Creation, and Software Documentation

---

1. Introduction and Overview

Xray is a premier **Test Management application** that integrates seamlessly into Jira Cloud. Unlike external testing tools that require complex synchronization, Xray embeds the entire testing lifecycle directly within Jira, allowing teams to manage requirements, tests, executions, and bugs without leaving their core development environment.

In Jira Cloud, Xray functions by adding specific **Issue Types** (e.g., Test, Test Execution, Test Plan) and **Custom Fields** to your projects. This allows testers to leverage Jira's native features—such as workflows, permissions, agile boards, and search capabilities—for testing purposes. Xray supports both manual exploratory testing and automated testing through robust REST and GraphQL APIs.

---

2. Testing Theory & How It Works

Xray operates on the principle that testing assets should be treated with the same rigor as development tasks. The core logic is built around **Traceability** and **Coverage**.

The Core Workflow (Plan, Design, Execute, Report)

Xray encourages a structured workflow that mirrors standard software development lifecycles:

1. **Plan:** A Test Plan is created to define the scope of testing for a specific version or sprint.
2. **Design:** Tests (Manual, Cucumber, or Generic) are created and often linked directly to User Stories or Requirements. Preconditions can be defined for complex setup steps.
3. **Execute:** Testers create a Test Execution issue, assign it, and run the tests. This can be done manually via the UI or triggered automatically via CI/CD pipelines.
4. **Report:** Results are calculated automatically (e.g., "Pass," "Fail," "TODO"). Xray generates real-time reports on Requirement Coverage, Test Runs, and Trends.

How the Integration Works (API & Automation)

For teams practicing Continuous Integration/Continuous Delivery (CI/CD), Xray Cloud acts as the central reporting hub. Automated tests (e.g., Selenium, JUnit, Cypress) are triggered by code commits or schedules. Upon completion, these tools use Xray's REST API (specifically the `import/execution` endpoints) to send machine-readable results (like JUnit XML or Cucumber JSON). Xray then parses these results, updates the corresponding Test Runs, and calculates the overall impact on the Test Plan and Requirement coverage.

---

3. Types of Testing Issues (Entities)

Xray extends Jira by introducing specific Issue Types to represent testing entities.

Issue Type	Description & Purpose	Usage Example
Test	The core definition of a test case. Can be **Manual** (step-by-step), **Cucumber** (Gherkin language), or **Generic** (unstructured, often used for automation IDs).	A login test with steps: 1. Enter username, 2. Enter password, 3. Click login.
Pre-Condition	Defines the initial state required before a Test can be executed. Reusable across multiple Tests.	"User must be logged out," or "Database must contain product X."
Test Set	A simple, flat list used to group Tests logically (e.g., "Smoke Tests," "Regression Tests"). Static collection.	Grouping all tests related to "Checkout Functionality."
Test Plan	A dynamic, higher-level entity used to organize **Test Executions** for a specific goal (e.g., a Release or Sprint). Provides a consolidated dashboard of progress.	"Sprint 24 Test Plan" containing all Test Executions planned for that sprint.
Test Execution	A task or session where testing is performed. An instance where you assign a tester or automation to run a specific set of Tests against a specific **Environment** and **Revision** (build number).	"Execution of Regression Suite - Build 4.2 on Staging."
Test Run	*Internal Entity.* Not user-created but created automatically whenever a Test is included in a Test Execution. Holds specific results (Pass/Fail) and comments for that single test on that specific day.	The specific result of the "Login Test" run at 10:05 AM on Chrome.

---

4. Using Xray for User Acceptance Testing (UAT)

User Acceptance Testing is the final validation phase where actual end-users or clients verify that the software meets their business needs. Xray is exceptionally well-suited for UAT because it provides structure, traceability, and sign-off capabilities without requiring technical expertise from testers.

4.1 Why Xray Works Well for UAT

UAT Need	How Xray Addresses It
Business-focused tests	Tests can be written in plain language (Manual Test step fields) without code or technical jargon.
Separation from dev tests	You can create a dedicated UAT Test Set or Test Plan filtered only for UAT scenarios.
Sign-off process	UAT Test Executions can require approval workflows or be marked as "Passed" only after business sign-off.
Feedback loops	Failed UAT tests can automatically create Bug issues linked back to the original requirement.
Audit trail	Every UAT test run is timestamped and attributed to a specific UAT participant.

4.2 Recommended UAT Workflow in Xray

Step 1 – Create a UAT Test Plan

- Create a **Test Plan** issue named "Release 2.0 – UAT".
- Set its scope to the specific version or milestone.
- Link it to the Epic or Feature being accepted.

Step 2 – Write UAT Tests in Business Language

- Create **Test** issues with the "Manual" type.
- Use business-readable steps:
  - **Step:** "Log in as a Procurement Manager"
  - **Expected Result:** "The Purchase Order dashboard shows pending approvals"

Step 3 – Organize UAT Tests into a Test Set

- Create a **Test Set** called "UAT – Procurement Module."
- Add all relevant UAT tests.

Step 4 – Assign UAT Execution to Business Users

- Create a **Test Execution** issue.
- Assign it to the business stakeholder (e.g., John from Finance).
- In the Test Execution, add a custom field for "UAT Sign-off Date."

Step 5 – Business User Runs Tests

- The business user logs into Jira Cloud.
- They open the assigned Test Execution.
- For each test, they click "Run Test" and select:
  - **Pass** (if the feature works as required)
  - **Fail** (if it does not meet business needs)
  - **TODO** (if they cannot complete due to environment issues)

Step 6 – Automatic Bug Creation on Failure

- When a UAT user marks a test as **Fail**, Xray can be configured to automatically prompt:
  - "Create a Bug from this failure?"
  - The bug is pre-linked to the original requirement and the failed test.

Step 7 – UAT Completion and Sign-off Report

- Once all tests in the UAT Test Execution are passed, Xray generates a **UAT Completion Report**.
- This report can be exported as PDF and attached to the release ticket as formal sign-off.

4.3 UAT-Specific Customizations in Xray Cloud

Customization	Purpose
Add a "Business Approver" user group	Restrict UAT Test Execution assignments to only business users.
Create a "UAT Environment" dropdown	Track whether testing was on Staging, Demo, or Production-like environment.
Add a "Business Criticality" field to Tests	Prioritize which UAT tests are mandatory for release.
Use Dashboards for UAT Progress	Build a Jira dashboard showing "UAT Tests Passed vs Failed by Module."

---

5. Creating a UAT Manual / Test Manual / New Software Manual Using Xray

One of the most powerful and overlooked features of Xray is its ability to **generate human-readable testing and user manuals directly from your test cases**. This means you can create a UAT manual, an internal test manual, or even an end-user software manual without duplicating effort.

5.1 The Concept: Tests as Documentation

In Xray, a well-written **Manual Test** case contains:

- Preconditions
- Step-by-step actions
- Expected results

If you write these clearly, you have already written 80% of:

- A **UAT manual** (for business users)
- A **Test manual** (for QA team onboarding)
- A **New software manual** (for end-users learning the system)

5.2 How to Structure Xray Tests for Documentation

Documentation Type	How to Structure the Xray Test
UAT Manual	Use business roles (e.g., "As a Sales Rep"), business outcomes ("Should create a quote"), and screenshots in the Test step description.
Test Manual (Internal QA)	Include test data requirements, edge cases, and expected system logs or API responses in the step results.
End-User Software Manual	Write tests as tutorials: "Step 1: Click the blue 'New Order' button. Step 2: Enter customer email. Expected: The system validates email format."

5.3 Exporting Manuals from Xray (Built-in and Custom)

Method	Output Format	Best For
Export Test to PDF (individual test)	PDF with test steps, preconditions, and expected results	Quick reference guides

Method	Output Format	Best For
Export Test Set to PDF	Combines multiple tests into one document	Module-level UAT manuals
Export Test Plan with Details	Includes test descriptions, owners, and links to requirements	Compliance documentation
Xray JSON API → Custom Doc Generator	JSON → Markdown → HTML/PDF (use Pandoc or custom script)	Branded, professional manuals

5.4 Step-by-Step: Creating a UAT Manual from Xray

Step 1 – Write UAT Tests as Narratives

- In your Test issue, under "Manual Test Steps," write each step as a complete instruction.
- **Example:**
  - **Step 1:** "Navigate to the Purchase Order screen from the main dashboard."
  - **Expected Result 1:** "The system displays a list of open purchase orders."
  - **Step 2:** "Click the blue 'Approve' button next to PO-12345."
  - **Expected Result 2:** "PO-12345 moves to 'Approved' status, and an email notification is sent."

Step 2 – Add Screenshots to Test Steps

- Xray allows image attachments per test step (via rich text editor).
- Attach annotated screenshots showing exactly where to click.

Step 3 – Organize Tests into a Logical Flow

- Create a **Test Set** called "UAT Manual – Release 2.0."
- Order the tests to match the user journey, not random priority.

Step 4 – Export to PDF with Custom Settings

- Go to the Test Set → More actions → Export to PDF.
- Choose "Include Step Details" and "Include Attachments."

Step 5 – Brand the Output (Optional via API)

- Use Xray's REST API to fetch test data as JSON.
- Run a script (Python, Node.js) to convert JSON → Markdown → PDF with your company logo and cover page.

5.5 Example: From Test Case to End-User Manual

Raw Xray Test Case (Manual Test)

> **Precondition:** User is logged in as Inventory Manager.
> **Step 1:** Click "Reports" → "Stock Levels".
> **Expected:** Grid shows current stock.
> **Step 2:** Search for "SKU-123".
> **Expected:** Row highlights in yellow.

Exported as End-User Manual (automatically generated)

> **2.1 Checking Stock Levels**
> *Before you begin:* Ensure you are logged in as an Inventory Manager.
>
> 1. From the main menu, click **Reports**, then select **Stock Levels**.
> 2. The system will display a grid of current stock across all warehouses.
> 3. In the search box, enter the SKU (e.g., "SKU-123").
> 4. The matching row will be highlighted in yellow.
>
> ✅ **You have successfully checked stock for a specific product.**

5.6 Automation Script for Custom Manual Generation (Pseudo-code)

If you need fully branded manuals, use Xray's API:

```
# Python pseudo-code for generating a manual from Xray
import requests
from markdown import markdown
import pdfkit

# Fetch all tests from a Test Set
test_set_id = "TS-101"
response = requests.get(
    f"https://your-
instance.atlassian.net/rest/raven/1.0/api/testset/{test_set_id}/test",
    headers={"Authorization": "Bearer YOUR_TOKEN"}
)
tests = response.json()

manual_content = "# User Acceptance Test Manual - Release 2.0\n\n"
for test in tests:
    manual_content += f"## {test['key']}: {test['summary']}\n"
    manual_content += f"**Precondition:** {test['precondition']}\n\n"
    for step in test['manualSteps']:
        manual_content += f"**Step:** {step['step']}\n"
        manual_content += f"**Expected:** {step['expected']}\n\n"
```

```
# Convert to PDF
pdfkit.from_string(manual_content, 'UAT_Manual.pdf')
```

6. Best Practices for UAT & Manual Creation in Xray

Practice	Why It Matters
Write tests in plain language, no technical jargon	Business users and manual readers must understand without QA training.
Use consistent naming conventions	`[UAT] Login - Forgot Password` makes filtering and export easy.
Attach screenshots or mockups directly to test steps	Visuals reduce misinterpretation in UAT and manuals.
Version your UAT Test Plan for each release	You can reuse previous UAT manuals as starting points.
Assign UAT Test Executions to real business roles	"Purchase Officer" group, not just "John Smith" (handles turnover).
Export PDFs at each release milestone	Gives legal/compliance a static record of UAT sign-off.

---

7. Limitations & Workarounds for Manual Creation

Limitation in Xray Cloud	Workaround
No native "Merge Tests to One Professional Document" button	Use the REST API or add-ons like "Better Excel Exporter for Jira."
Exported PDFs show Jira metadata (e.g., issue key, status)	Customize the export template (requires Xray Enterprise or API).
Screenshots in steps export at low resolution	Upload high-res images; test before final export.
No built-in table of contents in PDF exports	Post-process the PDF using a tool like `pdftk` or `DocRaptor`.

---

8. Summary Table: Xray for UAT vs. Test Manual vs. Software Manual

Use Case	Primary Xray Issue Type	Key Feature Used	Output Delivered
User Acceptance Testing	Test Execution (assigned to business users)	Manual Test step execution, Pass/Fail, Bug auto-creation	UAT sign-off report
UAT Manual	Test Set of manual tests	Export Test Set to PDF with step details	UAT instruction booklet
Internal Test Manual	Test (Manual type with edge cases)	Preconditions + expected results	QA onboarding guide
New Software Manual	Test (written as tutorials)	Step-by-step + screenshots + API export to branded PDF	End-user documentation

---

9. Final Recommendations

1. **For UAT:** Create a dedicated "UAT" Test Plan per release. Assign UAT Test Executions to a separate Jira group. Automate bug creation on failure.
2. **For Manuals:** Start writing every Manual Test as if it will be read by a new hire or a customer. You will gain both test coverage and documentation for free.
3. **For Automation:** Use the Xray API to generate branded, professional PDF manuals directly from your Test Sets on every release.
4. **For Traceability:** Always link your UAT tests back to the original Requirement or User Story. This gives you release readiness at a glance.

By using Xray not just as a testing tool but as a **documentation engine** and **UAT collaboration hub**, you save dozens of hours of manual writing and ensure that your test assets deliver value beyond bug detection.

End of Document