This document attempts to lay out the work that needs to be done in order to integrate automated AT response collection in [ARIA-AT App](https://github.com/w3c/aria-at-app). It describes the minimum set of user-facing capabilities which will result in a viable system. It also explains what those capabilities imply for the three relevant subsystems: the ARIA-AT App client (the "front-end"), the ARIA-AT App server, and the nascent AT response collector. It proposes a workflow that satisfies the needs. Finally, it identifies functionality which has been intentionally deferred for some future iteration.

## Project goal

ARIA-AT uses automated systems to ensure that humans only run tests when the test and/or the implementation behavior have not previously been reviewed.

## Vocabulary

### Test

See [the ARIA-AT project glossary](https://github.com/w3c/aria-at/wiki/Glossary)

### Test Plan

See [the ARIA-AT project glossary](https://github.com/w3c/aria-at/wiki/Glossary)

### Test Plan Run

See [the ARIA-AT project glossary](https://github.com/w3c/aria-at/wiki/Glossary)

### verdict

a subjective judgment as to whether a given AT response satisfies a given test assertion; verdicts are typically rendered by humans, but ARIA-AT may copy verdicts from [historic test results](#_lplf7acxtptb) to a new [Test Plan Run](#_brg12nh36129) if there is evidence that the test and implementation have not changed

### collection job

The process of collecting AT responses for a given Test in a given Test Plan using a given AT release and a given browser release

### collection job status

One of "queued", "running", "canceled", "error", "complete", or “skipped”; “skipped” collection jobs never transition out of that state; all other collection jobs start as "queued", and typically transition to "running", before finally transitioning to either "error" or "complete." A "queued" or "running" collection job may be canceled, in which case, the state transitions immediately to "canceled."

### approved test results

Test Plan Run results are considered "approved" if one of three conditions are satisfied: if the corresponding Test Plan is in the Draft phase and a Test Admin has explicitly marked the Plan as "approved," if the corresponding Test Plan is in the Candidate phase, or if the corresponding Test Plan is in the Recommended phase.

### historic test results

A test has "historic" results for a given browser and AT if that same test has previously been run in that browser and AT at some point in the past and if those results have been [approved](#_75x5wawsn6yy); any modification to the test (e.g. via a change to the application code or a change to an assertion) disqualifies the previously-reported results from consideration as "historic"

## Minimum-viable set of capabilities

All of these capabilities should be restricted to Test Admins.

* Initiate a collection job for every Test in a Test Plan (including an AT version and browser version from a list of available options)
* View [collection job status](#_gk1bi8xsumgu) within the application
* Verify that a “running” [collection job](#_xpjoqwul93wm) is advancing
* Initiate a [collection job](#_xpjoqwul93wm) for one Test in a Test Plan (specifying AT version and browser version)
* Cancel any [collection job](#_xpjoqwul93wm) with the “queued” or “running” status

Any Tester:

* Assign verdicts to a Test Plan Run who's AT responses were collected by automation

The system:

* Assign a [verdict](#_88n0iwc41v0a) to an AT responses when that AT response matches the AT response in [historic test results](#_lplf7acxtptb)

## Subsystem responsibilities

To write these out, I (Mike) have intentionally made a couple assumptions:

* The ARIA-AT App server should store the status of [collection jobs](#_xpjoqwul93wm) so that it doesn't have to query the AT response collection system to render a page.
* The ARIA-AT App client should not communicate with the AT response collection subsystem. The ARIA-AT App server should act as a facade.

Please feel free to challenge these or any others.

In keeping with the goals of this document, the following responsibilities are implementation-agnostic, avoiding UI design details, database schema changes, communication protocols, etc.

* ARIA-AT App client
  + Provide an affordance to initiate a [collection job](#_xpjoqwul93wm) for a given Test Plan, AT version, and browser version
  + Document the [status](#_gk1bi8xsumgu) for each [collection job](#_xpjoqwul93wm)
  + Provide an affordance to view the progress of the [collection job](#_xpjoqwul93wm)
  + Provide an affordance to initiate a [collection job](#_xpjoqwul93wm) for a given test, AT version, and browser version (only for [status](#_gk1bi8xsumgu) of "canceled", "error", or "complete")
  + Provide an affordance to cancel a [collection job](#_xpjoqwul93wm) for a given Test (only for [status](#_gk1bi8xsumgu) of "queued" or "running")
  + Document the AT responses retrieved from a [collection job](#_xpjoqwul93wm)
  + Provide an affordance to assign verdicts to AT responses retrieved from a [collection job](#_xpjoqwul93wm)
* ARIA-AT App server
  + Maintain a list of available test environments
  + Store the ID and status of [collection jobs](#_xpjoqwul93wm)
  + Handle requests (from the AT response collection system) to change the [status](#_gk1bi8xsumgu) of any given [collection job](#_xpjoqwul93wm)
  + Forward requests to start any given [collection job](#_xpjoqwul93wm) (from the ARIA-AT App client to the AT response collection system)
  + Forward requests to stop any given [collection job](#_xpjoqwul93wm) (from the ARIA-AT App client to the AT response collection system)
  + Assign a [verdict](#_88n0iwc41v0a) to an AT response when that AT response matches the AT response in [historic test results](#_lplf7acxtptb)
* AT response collection
  + Handle requests for the list of available test environments
  + Handle requests to create [collection jobs](#_xpjoqwul93wm)
  + Report changes to [collection job status](#_gk1bi8xsumgu) to the app
  + Handle requests to cancel [collection jobs](#_xpjoqwul93wm)
  + Handle requests to view the progress of any given [collection job](#_xpjoqwul93wm)
  + Retain the logs for all [collection jobs](#_xpjoqwul93wm) indefinitely
  + Handle requests to view a list of all [collection jobs](#_xpjoqwul93wm) (for forensics)

## Workflow proposal

### Proposal description

* Test Admins may assign a new Test Plan Run to a user named "Response Collector Bot." This user represents the automation system. The Test Admin must also specify a web browser and AT that the collection system is to use (the system will initially support only one version of each, and it will display the available version at this moment, as the Test Admin is drafting the Test Plan).
* When the Test Admin creates a Test Plan Run that is assigned to the "Response Collector Bot," the collection system will immediately create a new collection job for each Test in the Test Plan using the specified web browser, web browser version, AT, and AT version. The system immediately assigns the “skipped” [status](#_gk1bi8xsumgu) to any collection jobs for which the underlying Test has no [historic test results](#_lplf7acxtptb)–this state is final.
* Test Plans assigned to the "Response Collector Bot" are rendered slightly differently than other Test Plans: each Test in such Test Plans also includes a [collection job status](#_gk1bi8xsumgu), a link to real-time logging information, and (depending on the collection job status) a button to cancel or retry the collection job.
* When an AT response collection job transitions to “done,” the system compares that AT response with that of the [historic test results](#_lplf7acxtptb). If the AT responses match, the system copies the corresponding [verdicts](#_88n0iwc41v0a) into the results for the new Test Plan Run.
* At any time, a Test Admin may reassign the Test Plan to a human user. When they do this, the Test Plan retains any AT responses from "completed" collection jobs, and any "queued" or "running" collection jobs are automatically canceled. Collection jobs with the status "complete," "error," or "canceled," or “skipped” retain their status.

### Proposal tradeoffs

#### Tradeoff: the Test is the lowest level of granularity for collection jobs

##### Drawbacks

* Limited insight from the UI (ARIA-AT App will not report on the progress of the automation system as it proceeds through the Commands in a Test; it will only report that collection is "In Progress" for the entire duration.)
* Inefficiency in failure recovery (if the system fails to collect a response for a single Command in a Test, but successfully collects responses for the others, "retrying" the collection will involve discarding the valid responses and collecting them again)

##### Alternative design

The "Command" could be the lowest level of granularity for collection jobs.

##### Reasons we recommend this design

* We have no reason to believe that collection failure will be particularly common.
* More precise status information will be available (albeit in a less user-friendly and less central location) via the logging data that the automation system makes available on the web.
* Additional granularity will complicate the protocol (increasing development time and surface area for bugs) and the user interface (increasing development time and the cognitive load on users).

#### Tradeoff: Automation has only a temporary effect on Test Plans

##### Drawbacks

Once a Test Plan is assigned to a human, it will not be possible to determine if any of the AT responses it contains were collected by automation.

##### Alternative design

We could extend the Test Plans to retain information about the provenance of the AT responses.

##### Reasons we recommend this design

* Test Admins will be responsible for approving the AT responses collected in automation, and following that approval, it's not clear who would benefit from understanding the provenance of the data.
* It most strongly reflects the philosophy that automation is simply a tool for executing the Working Mode (rather than an integral aspect of the process).
* Additional tracking will complicate the data model and the user interface.

## Future work

Because this design describes a minimum-viable product, it intentionally omits capabilities that the community group nonetheless expects in the long term. These omitted features include (at a minimum):

* selecting operating system version
* selecting old browser versions
* re-running a specific command within a test
* visualizing discrepancies between AT responses collected by the automated system and AT responses that have previously been approved for equivalent test runs
* explicitly designating [historic test results](#_lplf7acxtptb) (e.g. to signal that a change to a Test did not impact the expected behavior and therefore did not disqualify test results from consideration for reuse in future automated Test Plan Runs)