

## Minutes of ARG Meeting 63I

28 May 2025

### Attendees:

### Voting Members:

Steve Baird, Randy Brukardt, Jeff Cousins, Gary Dismukes, Bob Duff, Ed Fish, Niklas Holsti, Brad Moore, Alejandro Mosteo, Jean-Pierre Rosen, Justin Squirek, Tucker Taft, Tullio Vardanega, Richard Wai.

### Non-voting Members:

None.

### Observers:

Robin Leroy (Unicode liaison) (leaves 12:00), Ethan Luis McDonough (US)

## Meeting Summary

The meeting convened on Wednesday, 28 May 2025 at 10:43 hours EDT and adjourned at 13:33 hours EDT. The meeting was held using Zoom. The meeting covered many of the new and updated AIs on the agenda.

### AI Summary

The following AIs were approved:

AI22-0133-1/03 Make 10646 non-normative (14-0-0)

The following AIs were approved with editorial changes:

AI22-0022-1/05 Difficult example issues from WG 9 review (14-0-0)

AI22-0124-1/05 Aggregates and capacity of bounded containers (14-0-0)

AI22-0134-1/01 Allow both Add\_Named and Add\_Unnamed in aggregates (13-0-1)

AI22-0136-1/02 Allow indexing aspect on untagged types (13-0-1)

The intentions of the following AIs were approved but they require a rewrite:

AI22-0075-1/05 Explicitly aliased results (13-0-1)

The following AIs were discussed and assigned to an editor:

AI22-0129-1/03 Usage example for protected type with Dynamic\_Predicate

AI22-0135-1/02 Coextensions, functions calls, and storage pools

The following Github issues were discussed and assigned to an editor to create an AI:

#12 Iterator over vectors should preserve unconstrained state of elements of discriminated type

## Detailed Minutes

### Welcome

Steve welcomes everyone to this meeting.

### Apologies

Randy recalls that Christoph Grein is on vacation for a whole month and thus is missing the meeting.

## **Previous Meeting Minutes**

There were no comments on the minutes: Approve minutes: 14-0-0.

## **Date and Venue of the Next Meeting**

Our next electronic meeting is proposed for Wednesday, August 6. Gary says he is on vacation on the 6th. Tucker proposes a week later. Jeff says he will be away on the 13<sup>th</sup>. We look a week earlier, there are fewer conflicts that week. Wednesday, July 30<sup>th</sup> is adopted, with the usual time (10:30-13:30 EDT [-5 UTC]) and method (Zoom).

## **Format Updates on older AIs**

Randy reports that he has finished permanently reformatting the AIs to HTML that were originally formatted only in plain text. He still has a number of AIs that were started in plain text and later converted to Google Docs to reformat and repack the archives.

He notes that this project should reduce the number of broken links in the indexes. He has been maintaining a set of temporary HTML-formatted AIs in order that the Corrigendum creation program only needed to read HTML AIs. However, he had to remove those before creating an index (lest they get picked up and make links to non-existent CVS archives). That was often forgotten, leading to frequent broken links in the indexes. That should now be a thing of the past.

## **Progress report on the Corrigendum**

Randy has posted RM versions of the Corrigendum and Defect Reports (the latter are generated from the appropriate AIs). He's also provided an ISO formatted set to WG 9; those won't be posted publicly as they look like (but are not, at this point) official ISO documents.

## **Unfinished Action Items**

Randy asks Tucker about the update of AI22-0119-1 (canonical order). At a minimum, the issues with Image should be removed or extensively revised, as they are handled in AI22-0132-1. Tucker did not do this, he will try to do so next time.

## **Current Action Items**

The combined unfinished old action items and new action items from the meeting are shown below.

Steve Baird:

- AI for derived private types that happen to be an array (see discussion of AI22-0136-1)

Randy Brukardt:

- Github Issue #12

Editorial changes only:

- AI22-0022-1
- AI22-0124-1
- AI22-0134-1
- AI22-0136-1

Christoph Grein:

- AI22-0129-1 (with assistance from Tucker Taft)

Tucker Taft:

- AI22-0063-1 (Split work with Richard Wai)
- AI22-0071-2
- AI22-0075-1

- Revise AI22-0119-1 (see Unfinished Action Items above)
- AI22-0129-1 (assist Christoph Grein)
- Add examples of aggregates to AI22-0134-1
- AI22-0135-1

Richard Wai:

- AI22-0063-1 (Split work with Tucker Taft)

## ***Detailed Review***

The minutes cover detailed review of Ada 2022 AIs (AI22s). The AI22s are presented in numeric order, which is not necessarily the order in which they were discussed. Votes are recorded as “for”-“against”-“abstentions”. For instance, a vote of 6-1-2 would have had six votes for, one vote against, and two abstentions.

If a paragraph number is identified as coming from the working Ada 202y AARM, the number refers to the text in draft 3 of the Ada 202y AARM. Paragraph numbers in other drafts may vary. Other paragraph numbers come from the final Ada 2022 AARM; again the paragraph numbers in the many drafts may vary.

## ***Detailed Review of Ada 2022 AIs***

### **AI22-0022-1/05 Difficult example issues from WG 9 review**

Jeff is asked about these. We decide that all of the original questions have been answered.

We decide to remove the example from 9.5.3, since what it references does not exist.

We'll leave the 3.9.4 and 3.10 declarations.

Niklas notices a number of mistakes in the examples, and those are fixed.

Approve AI with changes: 14-0-0.

### **AI22-0075-1/05 Explicitly aliased results**

Tucker explains the AI. He says that **aliased** is similar to **ref** in some other languages.

Ed is concerned that this is complex. Randy suggests that this is another attempt to get this right. Ada 95 had return by reference (which had accessibility problems). Ada 2005 replaced that by anonymous access, which is too expensive at runtime (because of the need to pass in storage pools and finalization chains).

Steve asks whether this would be used to change the existing containers libraries. We could do that, but there are some issues with compatibility, tampering, and ambiguity. Tucker is looking at ways to make that work, but that is a separate AI (AI22-0131-1).

Bob would like some examples of calling the functions. Gary agrees, suggesting that all of these samples should include calls (in particular showing them used on the LHS of assignments).

Approve intent: 13-0-1. Ed abstains.

### **AI22-0124-1/05 Aggregates and capacity of bounded containers**

Randy explains the AI. He added the actual wording since we last discussed it. He moved and generalized the existing “applicable index constraint” rules.

We look at a number of issues, and make a few minor changes.

Approve AI with changes: 14-0-0.

### **AI22-0129-1/03 Usage example for protected type with Dynamic\_Predicate**

Christoph didn't change anything significant, and Tucker never worked on it. He wants to change this further. So we table this. It goes back to Christoph and Tucker (with Tucker needing to do something next).

### **AI22-0133-1/03 Make 10646 non-normative**

Robin explains that we don't need to reference 10646 now that we reference Unicode. Since we have to reference Unicode, there isn't any important reason to reference 10646 at all. He notes that he reordered some of the text to read better, because just inserting "The Unicode Standard" caused some places to say things like "the The Unicode Standard".

Robin had noted that we should consider changing the Unicode version to 16.0.0, since it has an official HTML version to link to. Randy thought we should do that separately (and near the end of a revision cycle), as we would need to change quite a few places not mentioned in this AI (as well as carefully check each reference). We agree that is the best plan.

Randy is asked if this is part of the Corrigendum. He says that it originally was intended to be, but he didn't get to it in time, so it will not be.

Approve AI: 14-0-0.

### **AI22-0134-1/01 Allow both Add\_Named and Add\_Unnamed in aggregates**

Tucker explained that he was trying to implement JSON. And he needed this combination.

There is very little conflict between the rules. The rules are intended to ensure that a specific aggregate only uses one of the routines Add\_Named or Add\_Unnamed.

Gary asks that the example ought to have some aggregates in it. Tucker is given an action item to do that.

Approve AI with changes: 13-0-1. Jeff abstains.

### **AI22-0135-1/02 Coextensions, functions calls, and storage pools**

Tucker explains that this is clarifying that an allocator inside a function returning an anonymous access acts as if it is at the point of the function call.

Tucker argues that the RM already says what happens here.

Randy complains that the question is not really answered. He agrees that Tucker's other references seem to conflict with the wording "for an allocator used to define the discriminant of an object" (which clearly does not apply to any other construct used in place of an allocator), but he has done nothing to resolve the wording conflict.

Randy argues that this should be declared a pathology and not answered. He is concerned about adding implementation cost to a feature that was supposed to be cheap (function calls returning anonymous access). Some implementations will have to materialize this "collection" at the call site, potentially for all such calls (coextension or otherwise).

Randy notes that this supposedly cheap operation has to pass in a storage pool and a finalization location. We wanted a cheap reference return, not all of this complication. (He thinks that allocators should have been banned for

anonymous access types, not given some meaning that no one asked for and we've had to work around ever since. But probably too late to fix this for Ada.)

Approve AI: Oppose: Randy; Abstain: Jean-Pierre; Niklas, Brad, Ed, Alejandro, Jeff, Steve. 6-1-7 which does not pass.

Steve offers to change his vote to make this pass at 7-1-6.

Editor's note: Unfortunately, 7-1-6 does not pass either (I gave incorrect information during the meeting). The procedures say "The vote for an action succeeds (and is considered a consensus for the action) if a majority of voting members present at an ARG meeting vote in favor of the action and less than 17% of the voting members present at the meeting vote against the action." We had 14 voters at the meeting, and a majority is more than 50%, so since there are no fractions of a vote, we would have needed 8 in favor for this to pass.

### **AI22-0136-1/02 Allow indexing aspect on untagged types**

Tucker explains the AI. These are defined in terms of prefixed notation, and that only was defined for tagged types. But that's no longer the case, and other issues are covered (hopefully) by the non-overrideable rules, so there no longer is any reason for limiting these aspects to tagged types.

Randy had suggested that the completion rule be written separately. We agree to do that. We need to check if the generic boilerplate is needed (Randy thinks so).

We never want user-defined indexing on an array type, because that would lead to confusion between the predefined indexing and the user-defined versions. Thus we make it illegal even when hidden. We wonder why the access type is different. That follows from the way prefixed views work (they can be used on private types completed by access types). Tucker makes the claim that private types are often completed by access types, but almost never by array types. Randy is dubious that a private type can be usefully completed with an access type; he's always ended up needing other data and having to wrap the access in a record. Others agree with Tucker. In any case, indexing won't work (just like prefixed notation won't work) when the full (access) type is visible. But it still will work for clients. The notion that more visibility is subtracting something is uncomfortable, but we already agreed to that for prefixed views and this is just sugar on top of that.

Steve asks if this rule covers a derivation of a private type with a full type that is an array. Randy says that the existing rule for containers does not seem to cover that case, so there seems to be a problem. He thinks it would be best to handle that in a separate AI (since it seems to come up in at least two cases). Steve is given an action item to create an AI to explore this question.

[Editor's note: If the full type (in this case, the array-ness) of the parent type is never visible for the derived type, then there is no problem – there can be no conflict between the predefined operations (which are not available for the derived type anywhere) and the user-defined ones. But if there is a place where the full type becomes visible (as can happen in child units), then a late check would be required. Note that such a check doesn't break privacy, as it would only be applied in cases where the full type is visible anyway. Not sure if such a check already occurs in other cases in the language (that seems likely); if so we'd like to use similar wording. Steve will have to figure that out.]

Approve AI with changes: 13-0-1. Jeff abstains.

### ***Detailed Review of Github Issues***

#### **Github Issue #12 Iterator over vectors should preserve unconstrained state of elements of discriminated type**

Gary tries to explain the possible solutions. He had proposed an aspect for access types and discriminants.

Randy suggested a more general proposal, having an aspect that applied to subtypes of (mutable) discriminated types. He notes that we have a long-standing similar issue with subprogram parameters. One has to test whether an

object is unconstrained at runtime, as the actual might be constrained. This provides a “tripping hazard” (as Bob might say) if the intent is that the actual is unconstrained.

We agree to pursue the more general proposal.

Randy will write an initial version of the AI for this.