Derek Barnett – 11 October 2011

Porting to BamTools 2.x

This document describes the process of porting applications from BamTools 1.x (BT-1x) to BamTools 2.x (BT-2x). If you are migrating older code from BamTools 0.x (BT-0x), please see the BamTools-1x porting guide first, available here. All changes in this document will be referencing BT-1x, so please make those updates first, then come back here to move to BT-2x.

Attention!

The primary reason for moving to BT-2x so soon (after recently moving to BT-1x in the spring), was the discovery of discrepancies in the coordinate system used by BamTools.

Basically I goofed when initially writing the code that handles regions and intervals. The SAM/BAM specification describes the coordinates used throughout BAM as being 0-based, and the bin-calculating pseudocode provided in that document describes both begin & end positions as 0-based as well. I read that explanation as meaning that intervals were 0-based, **CLOSED** – overlooking the notation that those binning intervals are intended to be 0-based, **HALF-OPEN**.

This was brought to my attention recently, and I've pushed out this update (along with some other features that I had already begun work out) as quickly as possible to fix this critical issue.

What does this mean for you?

All the intervals in BamTools, both the API and toolkit, were originally implemented (improperly) using the closed interval system for easy comptability (with my faulty understanding of) the binning coorindates strategy. With BT-2x, both the API and toolkit now move to the half-open interval. In short, this means that it affects any API call or command-line option that takes a region parameter, as well as the default return value for BamAlignment::GetEndPosition().

If you wrote C++ code or command-line scripts under the assumption that BamTools was using half-open intervals, your code should be fine moving forward. You may want to check old results, however, as your regions-of-interest would have been off by one at right boundaries.

If you wrote C++ code or command-line scripts based on the old, closed interval – then you'll need to update your logic to work as expected, which may be as simple as adding 1 to any region's right bound. Please check your code's logic; there is basically no way for BamTools to distinguish which interval you are intending to provide.

For both groups, you should be aware that potential discrepancies (though likely rare) may have occurred in reported alignments near the index bin boundaries. So it might be worth inspecting old results.

All apologies

I realize that this is a potentially huge headache for some (including myself) and offer my humblest & sincerest apologies for missing this in earlier releases. I hope that this porting guide

will help ease the transition, on this and other topics covered by the update. If you have any questions about the new coordinate system, please feel free to contact me directly; if you notice any odd behavior, please post an “Issue” to the GitHub site.

Introduction

Our development philosophy with BamTools so far has been to allow periodic breaks in binary compatibility while avoiding source incompatibilities as much as reasonably possible. (Changes in my codebase shouldn't force you to change your code.) However, there are occasions when this must be done: parts of the API have gotten overly complex or muddy (see the move to 1.x) or when there are major issues that need to be addressed and solved (as noted above).

BT-2x and beyond will be neither binary- nor source-compatible with the earlier BT-0x or BT-1x versions. Your programs compiled using those classes will not work with the new libraries.

BT-2x adds some additional features and removes some methods. Any removed methods will be outlined in this document, so you'll know what they are, the rationale behind the removal, and, where applicable, alternative approaches to achieve the same behavior.

Please note that, for the sake of simplicity, return values may be ignored in the following code examples. I strongly advise checking them in production code, especially combined with the new GetErrorString() function that many of the API objects now provide.

BamAlignment

The biggest change to be aware of with BamAlignment is the GetEndPosition() function. In BT-1x, this function returned a 0-based, **closed** interval end position by default. In BT-2x, the default behavior is to provide the 0-based, **half-open** end position. The parameter name has changed to better reflect the intent. Setting *closedInterval* to true allows you to calculate the old, closed-interval position.

BT-1x

```
int GetEndPosition(bool usePadded = false, bool zeroBased = true) const;
```

BT-2x

```
int GetEndPosition(bool usePadded = false, bool closedInterval = false) const;
```

Removed methods

In addition, some of the methods that were previously marked as deprecated in BT-1x have been removed. They are listed below, along with the equivalent BT-2x code:

Alignment Flaq Setters

BT-1x (removed)	BT-2x
`void SetIsMateUnmapped(bool ok)`	`void SetIsMateMapped(bool ok)`
`void SetIsSecondaryAlignment(bool ok)`	`void SetIsPrimaryAlignment(bool ok)`
`void SetIsUnmapped(bool ok)`	`void SetIsMapped(bool ok)`

In these cases, the removed method is simply a complement of the preferred method. So instead of:

```
BamAlignment al;
al.SetIsUnmapped(true);
```

You should use:

```
BamAlignment al;
al.SetIsMapped(false);
```

And so on...

Alignment Tag Access

Some of the BAM tags previously had dedicated methods in the API. Retaining these tag-specific methods simply adds unnecessary clutter to the interface since we are not going to implement a method for every possible tag. They were marked as deprecated in BT-1x, and have been removed with the current release.

BT-1x (removed)

```
bool GetEditDistance(uint32_t& editDistance) const
bool GetReadGroup(string& readGroup) const
```

BT-2x

```
BamAlignment al;
al.GetTag("NM", editDistance);
al.GetTag("RG", readGroup);
```

BamIndex

The IndexCacheMode has been removed from BamIndex. Each index now provides a reasonable hybrid cache & paging strategy that keeps both disk access and RAM usage low. Since these strategies were implemented, any calls to set the cache mode have been essentially ignored. If a use case requires that such cache behavior be available, I am to open to re-instate it. However, as it stands, the cache mode functions are irrelevant and “crufty”, hence their removal.

BamMultiReader

Please be aware that the coordinate issue will affect any code that uses the multi-reader's SetRegion() functions. In addition, a few methods have been removed:

IsIndexLoaded

This method was deprecated in BT-1x, and has been removed for BT-2x. Use HasIndexes() instead.

BT-1x (removed)	BT-2x
`bool IsIndexLoaded(void) const;`	`bool HasIndexes(void) const;`

PrintFilenames

This method was deprecated in BT-1x, and has been removed for BT-2x.

BT-1x (removed)	BT-2x
`void PrintFilenames(void) const;`	(removed)

SetIndexCacheMode

With the elimination of BamIndex::IndexCacheMode (see above), this has been removed for BT-2x.

BT-1x (removed)	BT-2x
`void SetIndexCacheMode(const BamIndex::IndexCacheMode& mode);`	(removed)

SetSortOrder

The SetSortOrder() function has been removed. BamMultiReader now automatically determines the BAM files' sort order based on the SAM header's sort order tag (@HD SO:<x>). As long as your BAM file generating code sets this flag in the header appropriately, you should not miss this. Please note, though, that this requires all BAM files that use a shared multi-reader must use the same sort order.

BT-1x (removed)	BT-2x
`void SetSortOrder(const SortOrder& order);`	(removed)

BamReader

As with the multi-reader, please be aware that the coordinate issue will affect any code that uses the BamReader::SetRegion() functions.

SetIndexCacheMode

With the elimination of BamIndex::IndexCacheMode (see above), this has been removed for BT-2x.

BT-1x (removed)	BT-2x
`void SetIndexCacheMode(const BamIndex::IndexCacheMode& mode);`	(removed)

BamRegion

This is more of a continued note regarding the coordinate issue than a porting example. Any existing code that uses a BamRegion will be syntactically compatible with BT-2x, but may be

logically incorrect now. Please be aware that the region is treated differently as of this updated release.

At the risk of sounding repetitive: all regions in BT-2x are 0-based, **HALF-OPEN**, no longer 0-based, **CLOSED**. Please inspect the code that uses any sort of BamRegion to be sure you have accounted for this behavior.

General Error Handling

Error messages should no longer pollute STDERR. Instead of printing a message (and sometimes rudely killing the application), these error messages will be availbe on many of the API's objects via a GetErrorString() function. When you encounter such an error, you may decide how to display that message to the user (if at all), how to recover from the error (if you can), etc.

Example:

```
BamReader reader;
if ( !reader.Open("does_not_exist.bam") ) {
    std::string message = reader.GetErrorString();
    // do something with the message, prompt user for different file,
    // terminate, or whatever makes sense for your application
}
```

Conclusion

I hope this guide proves helpful in porting over to BT-2x. There have been some other features added to this release as well. I hope to update the API tutorial soon to reflect these new capabilities.

If I've missed anything important, or you have any suggestions to improve this guide, please feel free to contact me with your feedback.

Thanks and happy hacking!

Derek Barnett
Marth Lab, Boston College
derekwbarnett@gmail.com