CHERI Tag Bit Transport Overlay Standard

CHERI SoC Working Group

Version	0.9
Date	08/09/2026

1 Introduction

1.1 Purpose

This standard defines a transport overlay for carrying CHERI tag bits alongside ordinary read and write transactions. It specifies how tag bits are transported with read and write data, and how components preserve the association between tag bits and data.

This standard does not define CHERI capability formats, capability translation, software ABI rules, memory-management policy, or cross-format capability sharing.

1.2 Scope

This standard applies to system fabrics that carry data for capability-aware memory or peripherals.

This standard defines mandatory mappings for AXI and CHI. A TileLink mapping is included as work in progress, but is not currently part of this standard.

1.3 Normative Keywords

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in BCP 14, RFC 2119, and RFC 8174 when, and only when, they appear in all capitals.

Lowercase uses of these words are descriptive and do not express a normative requirement.

1.4 Content

2 Terminology

Bus manager: An initiator capable of issuing memory transactions into a system fabric.

Capability-aware memory or peripheral: A bus target, such as memory or a peripheral, that maintains tag bit state for capability data granules. A bus target may contain both capability-aware and non-capability-aware address regions.

Capability data granule: A naturally aligned `YLEN`-bit region of data storage that can hold the data representation of one capability. Each capability data granule has one associated tag bit. The tag bit is not part of the capability data granule and is not included in `YLEN`.

Clear: A bit state in which the bit value is `0`.

Coherency group: A set of bus managers, caches, interconnect components, and memory-facing components that participate in a coherent view of memory.

Complete-granule write: A write that updates every byte of the data portion of one or more capability data granules. For each written capability data granule, the transaction also carries the corresponding write tag bit, or all corresponding partial write tag bits when the capability data granule is split across multiple data beats.

Data beat: One transfer of data on a protocol data channel. A data beat contains `DATA_WIDTH` bits of data.

I/O coherent bus manager: A bus manager whose transactions are coherent with a coherency group, but whose local cache is not snooped by other coherent agents.

Inverse-bit encoding: A resilient encoding in which each protected bit is accompanied by a second bit carrying the logical inverse of the protected bit.

Non-tag-aware bus manager: A bus manager that is not trusted to issue transactions that carry tag bit state.

Partial-granule write: A write that updates at least one byte, but not all bytes, of a capability data granule.

Partial tag bit: A transported tag bit associated with a data beat that contains only part of a capability data granule. Partial tag bits are used when `DATA_WIDTH` is less than `YLEN`. They allow tag bit state for a capability data granule to be carried across the data beats that contain that granule.

Read tag bits: Tag bits supplied with read data by a tag-aware target for the capability data granules returned by a read transaction.

Set: A bit state in which the bit value is `1`.

Tag bit: A non-addressable validity bit associated with a naturally aligned capability data granule. The tag bit is not part of the capability data granule.

Tag-aware bus manager: A bus manager that is trusted to issue transactions that carry tag bit state and to provide valid tag bit state for complete-granule writes.

Tag-aware target: A target, manager, subordinate, bridge, memory controller, or other memory-facing component that preserves tag bit state according to this standard.

Write tag bits: Tag bits supplied by a tag-aware bus manager for the capability data granules written by a write transaction.

YLEN: The bit length of a capability, excluding the tag bit, from the perspective of the bus manager that initiates a memory transaction. Commonly, but not definitively, `YLEN` is double the address size, for example `YLEN` = 128 for 64-bit addresses, or `YLEN` = 64 for 32-bit addresses. `YLEN` is equivalent to `CLEN`, which is used in some texts and specifications relating to CHERI.

3 General Requirements

[REQ-GEN-001] A capability-aware memory or peripheral MUST associate exactly one tag bit with each naturally aligned capability data granule.

[REQ-GEN-002] Tag bits MUST NOT be addressable through ordinary load or store transactions.

[REQ-GEN-003] For every capability data granule, the associated data and tag bit MUST be observed and updated atomically.

[REQ-GEN-004] A state in which a tag bit is set ('1') MUST result only from a tag-aware write that supplies a set ('1') write tag bit for a complete capability data granule, or from a trusted initialization mechanism defined by the implementation.

[REQ-GEN-005] A tag-aware target MUST NOT autonomously change a tag bit to set ('1') for data that was not received with valid tag bit state or produced by a trusted initialization mechanism.

[REQ-GEN-006] A non-tag-aware bus manager MUST NOT issue a transaction that transports or sets valid tag bit state.

[REQ-GEN-007] A partial-granule write to a capability-aware memory or peripheral MUST clear ('0') the tag bit associated with each affected capability data granule.

[REQ-GEN-008] A write from a non-tag-aware bus manager to a capability-aware memory or peripheral MUST clear ('0') the tag bit associated with each affected capability data granule.

[REQ-GEN-009] The preserve-or-clear rule is that, if a component cannot preserve the correct tag bit value for a capability data granule, it MUST clear ('0') the associated tag bit or return an error response.

4 Common Transport Requirements

[REQ-COM-001] Each supported protocol mapping MUST provide:

- read tag bits; and
- write tag bits.

[REQ-COM-002] Each protocol mapping MUST provide tag bit transport for both of the following cases:

- one or more complete capability data granules in each data beat; and
- one capability data granule split across multiple data beats of the same transaction.

For a data beat of width `DATA_WIDTH`, the number of returned read tag bits or write tag bits per data beat is:

```
N = DATA_WIDTH / YLEN,  when DATA_WIDTH >= YLEN

N = 1,                  when DATA_WIDTH < YLEN
```

[REQ-COM-003] For a conforming tag-aware interface, either `DATA_WIDTH` MUST be an integer multiple of `YLEN`, or `YLEN` MUST be an integer multiple of `DATA_WIDTH`.

[REQ-COM-004] When `DATA_WIDTH` is greater than or equal to `YLEN`, tag bit `i` MUST correspond to capability data granule `i` in the associated data beat, ordered from the least significant byte lane at `i`=0 to the most significant byte lane.

[REQ-COM-005] When `DATA_WIDTH` is less than `YLEN`, each data beat that carries tag bit state MUST have an associated partial tag bit for the capability data granule containing that data beat.

[REQ-COM-006] All partial tag bits associated with the same capability data granule MUST have the same value.

[REQ-COM-007] If partial tag bits associated with the same capability data granule are inconsistent, the receiver MUST NOT infer a set ('1') architectural tag bit for that granule. For a write transaction, a tag-aware target MUST clear ('0') the affected tag bit or return an error response. For a read transaction, a tag-aware bus manager MUST infer the architectural tag bit for the affected granule as clear ('0').

[REQ-COM-008] A tag-aware component that handles partial tag bits MUST preserve tag bit/data atomicity for the affected capability data granule. If the component cannot preserve that atomicity, it MUST clear ('0') the affected tag bit or return an error response.

[REQ-COM-009] For a complete-granule write, a clear ('0') write tag bit associated with a capability data granule MUST cause the tag-aware target to clear ('0') the tag bit associated with that capability data granule.

[REQ-COM-010] If a write transaction updates only part of a capability data granule, the tag-aware target MUST clear ('0') the tag bit for that granule, regardless of the value of the associated write tag bit.

[REQ-COM-011] An implementation MAY provide resilient encoding (redundancy) for the tag transport signals defined by this standard.

[REQ-COM-012] A conforming resilient implementation of this standard MUST use inverse-bit encoding. In inverse-bit encoding, each read tag bit and write tag bit is accompanied by a second bit carrying the logical inverse of the primary bit.

[REQ-COM-013] An implementation MAY use an alternative resilient encoding as an implementation-specific extension. Such an encoding may limit interoperability and MUST be documented by the implementation.

> **Note:** This standard defines resilient encoding only for the tag transport signals defined by this standard. It does not define, require, or restrict the use of parity, ECC, CRC, or other protection mechanisms for the data path or for other protocol metadata.

[REQ-COM-014] A component that receives inverse-bit encoded tag transport signals MUST check that each inverse bit is the logical inverse of its corresponding primary bit before using the associated value.

[REQ-COM-015] If a component detects a resilient encoding mismatch, it MUST NOT use the affected value as a valid set ('1') tag bit. The component MUST either return an error response supported by the base protocol, or treat the affected value as clear ('0').

5 AXI Mapping

5.1 AXI User Signaling

[REQ-AXI-001] An AXI implementation of this standard MUST use AXI user-defined signaling for tag bit transport.

[REQ-AXI-002] The AXI user signaling REQUIRED by this standard, using the value of `N` defined in Section 4, is:

- `N` bits of `RUSER` for read tag bits; and
- `N` bits of `WUSER` for write tag bits.

The default AXI bit allocation is:

Signal	Bit or Bits	Meaning
`RUSER[N-1:0]`	`N` bits	Read tag bits
`WUSER[N-1:0]`	`N` bits	Write tag bits

If AXI resilient encoding (redundancy) is implemented, the default AXI resilient bit allocation is:

Signal	Bit or Bits	Meaning
`RUSER[N-1:0]`	`N` bits	Read tag bits
`RUSER[2N-1:N]`	`N` bits	Inverse of read tag bits
`WUSER[N-1:0]`	`N` bits	Write tag bits
`WUSER[2N-1:N]`	`N` bits	Inverse of write tag bits

For AXI, `DATA_WIDTH` is `AXI_DATA_WIDTH`.

When `AXI_DATA_WIDTH` is greater than or equal to `YLEN`, tag bit `i` in `RUSER[N-1:0]` or `WUSER[N-1:0]` corresponds to capability data granule `i` in the associated data beat.

When `AXI_DATA_WIDTH` is less than `YLEN`, `RUSER[0]` or `WUSER[0]` is one of the partial tag bits for the capability data granule containing the associated data beat.

5.2 AXI Read Requirements

[REQ-AXI-003] If the addressed memory or peripheral is capability-aware, the tag-aware target MUST return the associated tag bits in `RUSER[N-1:0]`.

[REQ-AXI-004] If the addressed memory or peripheral is not capability-aware, the target MUST return `RUSER[N-1:0]` as clear ('0').

An AXI component that handles `RUSER` tag bits preserves their association with the corresponding `RDATA`, as required by **[REQ-GEN-003]** and **[REQ-COM-008]**.

5.3 AXI Write Requirements

[REQ-AXI-005] For an AXI write transaction that carries tag bit state, the tag-aware bus manager MUST provide the associated write tag bits in `WUSER[N-1:0]`.

If `AXI_DATA_WIDTH` is greater than or equal to `YLEN` and `WUSER[i]` is set ('1'), the write transaction is a complete-granule write for capability data granule `i`. If the write transaction does not write every byte of that granule, the partial-granule write rules in **[REQ-COM-010]** apply.

If `AXI_DATA_WIDTH` is less than `YLEN` and the partial `WUSER[0]` tag bits for a capability data granule are set ('1'), the write transaction is a complete-granule write for that capability data granule. If the write transaction does not write every byte of that granule, the partial-granule write rules in **[REQ-COM-010]** apply.

If a write transaction updates only part of a capability data granule, the associated tag bit is cleared ('0') as required by **[REQ-COM-010]**, regardless of the value of `WUSER[i]`.

Any AXI component that handles `WUSER` tag bits preserves the association between each tag bit and its corresponding `WDATA` capability data granule, or between each partial tag bit and its corresponding `WDATA` beat, as required by **[REQ-GEN-003]** and **[REQ-COM-008]**.

6 CHI Mapping

6.1 CHI RSVDC Signaling

[REQ-CHI-001] A CHI implementation of this standard MUST use CHI RSVDC signaling for tag bit transport.

[REQ-CHI-002] The CHI implementation MUST reserve the selected CHI RSVDC bits for DAT_RSVDC and preserve their values end-to-end across the CHI interconnect.

Note: A CHI-compliant interconnect is not necessarily required to preserve RSVDC bit values. Implementers should check that the selected CHI interconnect, fabric, or NoC preserves the selected RSVDC bits before using them for this overlay.

[REQ-CHI-003] The CHI RSVDC signaling REQUIRED by this standard, using the value of N defined in Section 4, is:

- N data-channel RSVDC bits for read tag bits; and
- N data-channel RSVDC bits for write tag bits.

The default CHI bit allocation is:

CHI overlay field	Bit or Bits	Meaning
DAT_RSVDC[N-1:0] on read data	N bits	Read tag bits
DAT_RSVDC[N-1:0] on write data	N bits	Write tag bits

If CHI resilient encoding (redundancy) is implemented, the default CHI resilient bit allocation is:

CHI overlay field	Bit or Bits	Meaning
DAT_RSVDC[N-1:0] on read data	N bits	Read tag bits
DAT_RSVDC[2N-1:N] on read data	N bits	Inverse of read tag bits
DAT_RSVDC[N-1:0] on write data	N bits	Write tag bits
DAT_RSVDC[2N-1:N] on write data	N bits	Inverse of write tag bits

For CHI, DATA_WIDTH is the width of the CHI data flit or data beat carrying the associated read or write data.

This mapping requires that the CHI DATA_WIDTH is greater than or equal to YLEN, as the minimum data width for a CHI interface is 128 bits.

Note: A CHI implementation with 256-bit capabilities and a 128-bit data width will require a new mapping.

Tag bit i in DAT_RSVDC[N-1:0] corresponds to capability data granule i in the associated data beat.

6.2 CHI Read Requirements

[REQ-CHI-004] If the addressed memory or peripheral is capability-aware, the tag-aware target MUST return the associated tag bits in read-data `DAT_RSVDC[N-1:0]`.

[REQ-CHI-005] If the addressed memory or peripheral is not capability-aware, the target MUST return read-data `DAT_RSVDC[N-1:0]` as clear ('0').

A CHI component that handles read-data `DAT_RSVDC` tag bits preserves their association with the corresponding read data, as required by **[REQ-GEN-003]**.

6.3 CHI Write Requirements

[REQ-CHI-006] For a CHI write transaction that carries tag bit state, the tag-aware bus manager MUST provide the associated write tag bits in write-data `DAT_RSVDC[N-1:0]`.

If write-data `DAT_RSVDC[i]` is set ('1'), the write transaction is a complete-granule write for capability data granule `i`. If the write transaction does not write every byte of that granule, the partial-granule write rules in **[REQ-COM-010]** apply.

If a CHI write transaction updates only part of a capability data granule, the associated tag bit is cleared ('0') as required by **[REQ-COM-010]**, regardless of the value of `DAT_RSVDC[i]`.

Any CHI component that handles write-data `DAT_RSVDC` tag bits preserves the association between each tag bit and its corresponding write-data capability data granule, as required by **[REQ-GEN-003]**.

7 TileLink Mapping

Note: The TileLink mapping in this section is work in progress. It is included to show the expected direction for future TileLink support, but is not currently part of this standard.

7.1 TileLink User Signaling

A future TileLink implementation of this standard is expected to use TileLink user-defined signaling for tag bit transport.

The expected TileLink user signaling, using the value of `N` defined in Section 4, is:

- `N` D-channel user bits for returned read tag bits; and
- `N` A-channel data user bits for write tag bits.

The default TileLink bit allocation is:

TileLink overlay field	Bit or Bits	Meaning
`d_user[N-1:0]`	`N` bits	Read tag bits
`a_data_user[N-1:0]`	`N` bits	Write tag bits

If TileLink resilient encoding (redundancy) is implemented, the default TileLink resilient bit allocation is:

TileLink overlay field	Bit or Bits	Meaning
`d_user[N-1:0]`	`N` bits	Read tag bits
`d_user[2N-1:N]`	`N` bits	Inverse of read tag bits
`a_data_user[N-1:0]`	`N` bits	Write tag bits
`a_data_user[2N-1:N]`	`N` bits	Inverse of write tag bits

For TileLink, `DATA_WIDTH` is the width of the TileLink data beat carrying the associated read or write data.

When `DATA_WIDTH` is greater than or equal to `YLEN`, tag bit `i` in `d_user[N-1:0]` or `a_data_user[N-1:0]` corresponds to capability data granule `i` in the associated data beat.

When `DATA_WIDTH` is less than `YLEN`, `d_user[0]` or `a_data_user[0]` is one of the partial tag bits for the capability data granule containing the associated data beat.

The TileLink implementation is expected to bind `d_user` and `a_data_user` to concrete TileLink user fields, bundle members, or diplomatic parameters.

7.2 TileLink Read Behavior

If the addressed memory or peripheral is capability-aware, the tag-aware target is expected to return the associated tag bits in `d_user[N-1:0]`.

If the addressed memory or peripheral is not capability-aware, the target is expected to return `d_user[N-1:0]` as clear ('0').

Any TileLink component that handles d_user tag bits preserves the association between each returned tag bit and its corresponding read-data capability data granule, or between each returned partial tag bit and its corresponding read-data beat.

7.3 TileLink Write Behavior

If DATA_WIDTH is greater than or equal to YLEN, each a_data_user[i] bit provides the tag bit value for capability data granule i in the associated write-data beat.

If DATA_WIDTH is less than YLEN, a_data_user[0] provides one of the partial tag bits for the capability data granule containing the associated write-data beat.

If DATA_WIDTH is greater than or equal to YLEN and a_data_user[i] is set ('1'), the write transaction is a complete-granule write for capability data granule i. If the write transaction does not write every byte of that granule, the partial-granule write rules in Section 4 apply.

If DATA_WIDTH is less than YLEN and the partial a_data_user[0] tag bits for a capability data granule are set ('1'), the write transaction is a complete-granule write for that capability data granule. If the write transaction does not write every byte of that granule, the partial-granule write rules in Section 4 apply.

PutPartialData is a partial-granule write unless the transaction is defined by the implementation as a capability-preserving complete-granule write. For a partial-granule write, the tag-clearing rules in Section 4 apply.

Any TileLink component that handles a_data_user tag bits preserves the association between each tag bit and its corresponding write-data capability data granule, or between each partial tag bit and its corresponding write-data beat.

8 Width Conversion and Resizing

[REQ-WID-001] Any component that changes the data width of a tag-aware interface MUST also resize the associated read and write tag bit fields.

[REQ-WID-002] When widening a data path, the component MUST combine the tag bits from the input beats so that each output tag bit remains associated with the same capability data granule as the corresponding output data.

[REQ-WID-003] When narrowing a data path, the component MUST split the tag bits so that each output tag bit remains associated with the same capability data granule as the corresponding output data.

[REQ-WID-004] If a width converter produces a tag-aware interface with `DATA_WIDTH` less than `YLEN`, it MUST produce partial tag bits according to Section 4.

[REQ-WID-005] If a width converter consumes a tag-aware interface with `DATA_WIDTH` less than `YLEN`, it MUST combine the partial tag bits associated with each capability data granule before producing an output tag bit for that granule.

If a width converter receives inconsistent partial tag bits for a capability data granule, the inconsistent partial-tag rules in Section 4 apply.

For any output capability data granule where the width converter cannot determine the correct tag bit value, the preserve-or-clear rule **[REQ-GEN-009]** applies.

9 Caches and Buffered Storage

[REQ-CCH-001] A cache and its subcomponents MUST preserve the association between each tag bit and its corresponding capability data granule whenever they store, observe, or forward tag bit state.

[REQ-CCH-002] If a cache stores data for a capability data granule, it MUST either store the associated tag bit or be located behind a tag-aware component that preserves equivalent tag bit behavior.

[REQ-CCH-003] Cache operations MUST preserve tag bit/data atomicity.

A cache or buffered storage component treats a non-tag-aware write or partial-granule write as described in Section 3.

If a cache or buffered storage component cannot determine the correct tag bit value for a capability data granule, the preserve-or-clear rule **[REQ-GEN-009]** applies.

10 Coherency Requirements

[REQ-COH-001] All tag-aware bus managers and tag-aware targets in a coherency group MUST have a consistent view of which address regions are capability-aware.

[REQ-COH-002] If a cache in a coherency group can hold data from capability-aware memory or peripherals, it MUST preserve the tag bit associated with each cached capability data granule.

[REQ-COH-003] Any coherent operation that transfers data for a capability data granule MUST transfer the associated tag bit state atomically with that data.

A cache in a coherency group preserves tag bit state for cached data from capability-aware memory or peripherals. If the cache cannot preserve the correct tag bit state, the preserve-or-clear rule **[REQ-GEN-009]** applies.

An I/O coherent bus manager may have a local cache that is not snooped by other coherent agents. Because that local cache cannot supply data to other agents through a coherent snoop, it is not required to preserve tag bit state. The tag bit transport and tag-clearing requirements still apply to the transactions it issues.

11 Error Handling

[REQ-ERR-001] If a tag-aware target receives an unsupported transaction that carries tag bit state, it MUST either:

- complete the transaction with all affected tag bits clear ('0'); or
- return an error response supported by the base protocol.

[REQ-ERR-002] If an implementation provides tag bit-specific error reporting, the mapping of that error reporting MUST be documented by the implementation.

12 Conformance

A component conforms to this standard if it:

- implements the AXI protocol mapping defined in Section 5 when it implements this standard for AXI;
- implements the CHI protocol mapping defined in Section 6 when it implements this standard for CHI;
- implements the common transport requirements in Section 4;
- implements the width conversion requirements in Section 8 when it changes data width;
- implements the cache and buffered storage requirements in Section 9 when it stores, buffers, or forwards capability data granules;
- implements the coherency requirements in Section 10 when it participates in a coherency group; and
- preserves tag bit/data atomicity for all transactions to capability-aware memory or peripherals that it handles.

References

- RFC 2119, "Key words for use in RFCs to Indicate Requirement Levels"
- RFC 8174, "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words"
- CHERI Alliance SoC System Working Group, "Transport, Caching and Storage of Capabilities"
- AMBA AXI specification
- AMBA CHI specification
- TileLink specification