App Connect Enterprise V13
組み込みグローバル・キャッシュ利用ガイド

2025/07/18

日本アイ・ビー・エム株式会社
日本アイ・ビー・エム システムズ・エンジニアリング株式会社

ご注意

この資料に含まれる情報は可能な限り正確を期しておりますが、日本アイ・ビー・エム株式会社および日本アイ・ビー・エム システムズ・エンジニアリング株式会社の正式なレビューを受けておらず、当資料に記載された内容に関して当コンファレンスの主催者である日本アイ・ビー・エム システムズ・エンジニアリング株式会社は何ら保証するものではありません。従って、この情報の利用またはこれらの技法の実施はひとえに使用者の責任において為されるものであり、資料の内容によって受けたいかなる被害に関しても一切の補償をするものではありません。

はじめに

- 当資料は、App Connect Enterprise V13.0.3.0より新たに追加された組み込みグローバル・キャッシュの利用方法について解説したガイドです。
- 記載内容は、ACE V13.0.3.0を使用した検証結果を元に執筆しています。
  - 一部、V13.0.4.0での検証結果を含む

目次

- 内容
  - グローバル・キャッシュとは
  - Java Computeノードを使用したグローバル・キャッシュの利用方法
  - Mappingノードを使用したグローバル・キャッシュの利用方法
  - 複数統合サーバ間でのグローバル・キャッシュ共有
  - グローバル・キャッシュの運用管理

グローバル・キャッシュとは

グローバル・キャッシュについて

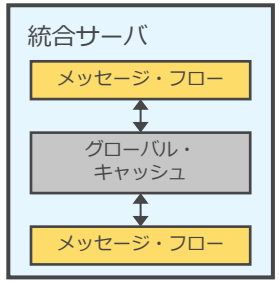
- グローバル・キャッシュとは、再利用したいデータをメモリ上に保存して共有する機能の総称
  - 統合サーバ内や複数統合サーバ間でのデータ共有が可能で、データベースのような代替手段が不要
  - 共有するデータは、グローバル・マップと呼ばれる実体に格納
  - グローバル・マップとのやり取りはフローのトランザクション外で実行され即座にコミットされるため、後続のフロー処理が失敗しても、マップに対する操作はロールバックされない
- グローバル・キャッシュのタイプ
  - 要件に応じて、以下のグローバル・キャッシュから選択可能
    - 当資料では、組み込みグローバル・キャッシュについて解説する

キャッシュ・タイプ	アクセス方法	最大保持期間	最大スコープ	管理方法	制限事項
組み込みグローバル・キャッシュ ★ (Embedded global cache)	Java Computeノード /Mappingノード	キャッシュを共有する すべての統合サーバが 同時に停止するまで	統合サーバ間	• server.conf.yaml • ibmint display cacheコマンド • ibmint clear cacheコマンド	V13.0.3.0以降、利用可能
外部Redisグローバル・キャッシュ (External Redis global cache)		外部Redisの設定に依存	ACEの外部	• 外部Redisの管理 • server.conf.yaml	V13.0.3.0以降、利用可能
組み込みWXSグリッド (Embedded WebSphere eXtreme Scale grid) ※非推奨		キャッシュを共有する すべての統合サーバが 同時に停止するまで	統合サーバ間	• server.conf.yaml • mqsicacheadminコマンド	• Java 8の指定が必要 • V13.0.3.0以降、非推奨 • コンテナ環境ではサポート不可
外部WXSグリッド (External WebSphere eXtreme Scale grid) ※非推奨		外部WXSの設定に依存	ACEの外部	• 外部WXSの管理 • server.conf.yaml	• Java 8の指定が必要 • V13.0.3.0以降、非推奨

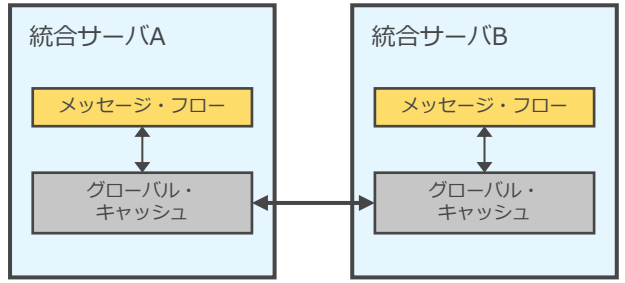
利用可能なキャッシュ機能についての詳細は以下を参照
https://www.ibm.com/docs/en/app-connect/13.0.x?topic=caching-data-overview

組み込みグローバル・キャッシュについて

- 組み込みグローバル・キャッシュとは、V13.0.3.0より追加された組み込みのキャッシュ機能
  - IIBまたはACEのV13.0.3.0より前のバージョンで使用されていたWebSphere eXtreme Scale(WXS) 組み込みキャッシュ（現在は、組み込みWXSグリッドと呼ばれる）の後継
    - 現在、組み込みWXSグリッドは非推奨であるが、ACEでJava 8がサポートされている限りは引き続き動作する
- 組み込みグローバル・キャッシュを使用して、一つのフロー内、異なるフロー間、または異なる統合サーバ間でデータを保存・共有することが可能
  - 単一統合サーバ内で共有する場合
    - 組み込みグローバル・キャッシュはデフォルトで有効のため、設定は不要
      - 統合サーバでWXSグリッドやローカル・キャッシュを使う構成をしていない、または外部のRedis接続用にRedisConnectionポリシーを使ってマップを作成していない限り、組み込みグローバル・キャッシュを使用していることになる
  - 統合サーバー間でレプリケーションする場合
    - データを共有する統合サーバごとにserver.conf.yamlの設定が必要
      - ResourceManagersセクションのGlobalCacheサブセクションにてプロパティ設定を行う
      - 参考：https://www.ibm.com/docs/en/app-connect/13.0.x?topic=caching-configuring-embedded-global-cache
- 組み込みグローバル・キャッシュの特徴
  - Java 8 および Java 17 に対応
  - コンテナ環境にも対応



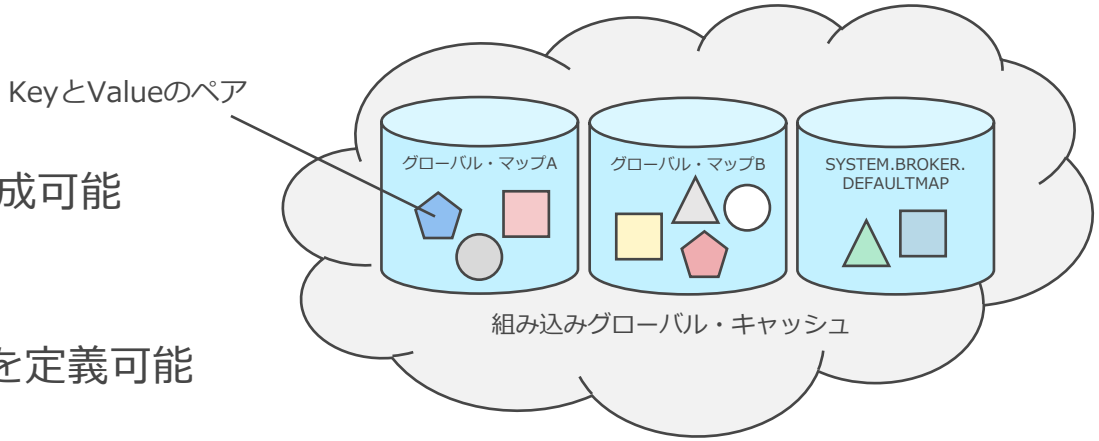
単一統合サーバ内の異なるフロー間でデータを共有



異なる統合サーバ間でデータを共有

組み込みグローバル・キャッシュの機能

- グローバル・キャッシュとグローバル・マップ
  - グローバル・キャッシュ
    - グローバル・キャッシュ機能の総称
    - ACE上(組み込み)または外部(Redisまたは外部WXS)に構成可能
  - グローバル・マップ
    - 一つひとつのKeyとValueのペアを格納する場所
    - 複数のグローバル・マップ(KeyとValueのペアの集合体)を定義可能
      - それぞれのマップはユニークな名前を持つ



- グローバル・マップの利用
  - KeyとValueのペアを格納・取得・更新・削除することが可能
  - マップは、マップ名を指定して新規作成することも、デフォルトのマップを使用することも可能
    - 存在しないマップにアクセスしようとしたときに、そのマップが自動的に新規作成される
      - "SYSTEM.BROKER" で始まる名前を除き、任意のマップ名を指定して使用可能
    - マップ名を指定せず、デフォルトのマップ(SYSTEM.BROKER.DEFAULTMAP)を使用することも可能
- Java ComputeノードまたはMappingノードを使用して、グローバル・マップにアクセス可能
  - マップに格納するデータは、KeyとValueのペアである必要がある
  - マップからデータが自動的に削除されるタイミング(保持期間)を秒単位で指定することも可能
    - デフォルトでは、マップ内のデータが自動で削除されないように保持期間は0に設定されている
      - ただし、統合ノードおよび統合サーバ再起動時にはクリアされる

Java Computeノードを使用したグローバル・キャッシュの利用方法

Java Computeノードを使用してグローバル・キャッシュ機能を利用

- Java Computeノードを使用してグローバル・マップとやり取りすることが可能
  - MbGlobalMapオブジェクトを用いてマップにアクセスしたり、そのマップ内にデータを格納・取得するといった操作を行う

MyMapという名前のマップにアクセスする場合（マップが存在しない場合は自動的に作成される）

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
```

デフォルトのマップにアクセスする場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap();
```

  - KeyとValueでサポートされるデータ型
    - グローバル・キャッシュで利用可能なKeyとValueのデータ型一覧

	サポートされるデータ型
Key	• プリミティブ型 • 文字列 • プリミティブ型や文字列の配列
Value	• プリミティブ型 • 文字列 • Javaオブジェクト（※独自のJavaクラスをグローバル・キャッシュで使用したい場合は、そのクラスを含むJARファイルをshared-classesディレクトリ下に配置する必要がある）

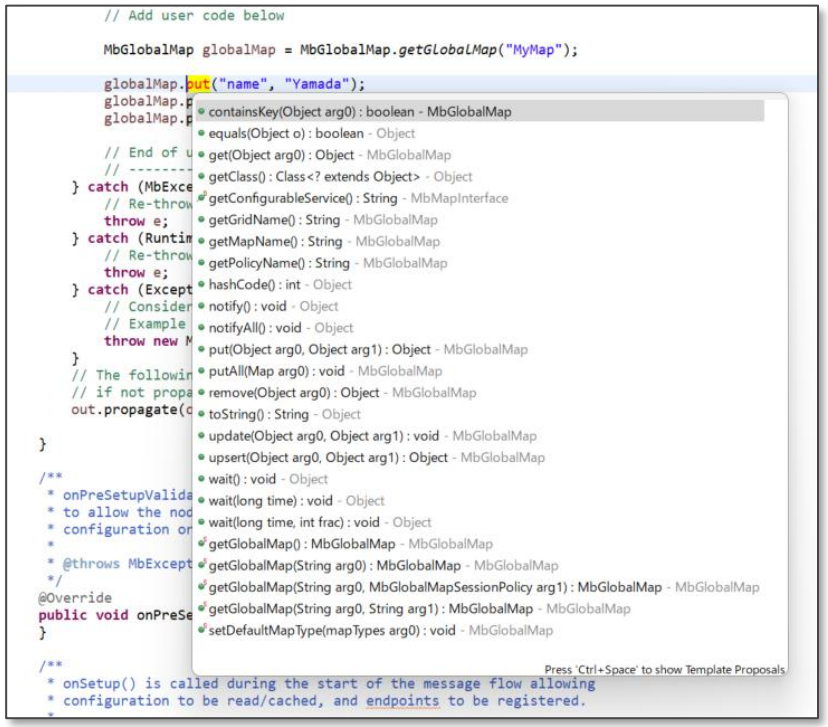
Java Computeノードを使用してグローバル・キャッシュ機能を利用

- Java Computeノードを使用してグローバル・マップとやり取りすることが可能 (続き)
  - グローバル・マップ内のデータに対する主な操作
    - 主なメソッド一覧

メソッド	機能説明
put	マップにKeyとValueのペアを格納
get	マップに存在するKeyのValueを取得
remove	マップから単一のKeyとValueのペアを削除
containsKey	マップにKeyが存在するかどうかを判定
update	マップに存在するKeyのValueを更新
upsert	マップにKeyが存在すればValueを更新、 Keyが存在しなければKeyとValueのペアを格納

  - コンテンツアシストを使用して、利用可能な操作を一覧表示可能
    - カーソルを任意の位置に置き、Ctrl + Spaceを実行

コンテンツアシスト使用例



Java Computeノードを使用してグローバル・マップ内のデータを操作する詳細方法については以下を参照
https://www.ibm.com/docs/en/app-connect/13.0.x?topic=java-accessing-global-cache-by-using-javacompute-node

グローバル・マップに対する操作

- putメソッドを使用して、マップに対してKeyとValueを格納
  - データをマップに格納するために、 KeyとValueのペアを作成する

MyMapに対してKeyとValueを格納する場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
globalMap.put("name", "Yamada");
```

- getメソッドを使用して、マップに格納されたKeyに対するValueを取得

MyMapに格納されたKeyに対するValueを取得する例

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
String val = (String)globalMap.get("name");
```

- removeメソッドを使用して、マップに格納された特定のKeyとValueのペアを削除
  - キャッシュの肥大化を防ぐため、 使用済みのKeyとValueのペアをマップから削除する
  - ibmint clear cacheコマンドを使用して、マップ内のすべてのKeyとValueのペアを一括削除することも可能

MyMapに格納された特定のKeyとValueのペアを削除する場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
globalMap.remove("name");
```

グローバル・マップに対する操作

- containsKeyメソッドを使用して、指定したKeyがマップに存在するかどうかを確認
  - Keyが存在する場合はtrueを、存在しない場合はfalseを返す

指定したKeyがMyMapに存在するかどうか確認する場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
boolean exists = globalMap.containsKey("name");
```

- updateメソッドを使用して、マップに格納されているKeyに対するValueを更新
  - マップに存在しないKeyのValueを更新しようとするとエラーになる

MyMapに格納されているKeyに対するValueを更新する場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
globalMap.update("name", "Tanaka");
```

- upsertメソッドを使用して、マップにKeyが存在する場合はValueを更新、存在しない場合は新たにKeyとValueのペアを格納
  - upsertは、V13.0.4.0で新たに追加されたメソッド

MyMapにKeyが存在すればそのValueを更新、存在しなければKeyとValueのペアを格納する場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap");
globalMap.upsert("name", "Tanaka");
```

グローバル・マップ内のデータに保持期間を指定

- グローバル・マップからKeyとValueのペアを削除するタイミングを指定することが可能
  - MbGlobalMapオブジェクトを使用してグローバル・マップにアクセスし、データが自動的に削除されるまでの保持期間(Time to Live)を秒単位で指定
    - MbGlobalMapオブジェクトから参照可能なセッション・ポリシーを作成して保持期間を設定する
  - 保持期間は、セッション・ポリシーが最後に更新された時点からカウントされる
  - 指定した保持期間は、そのMbGlobalMapオブジェクトを使用して作成されるすべてのエントリに適用される
    - 既にマップに存在しているエントリや、他のMbGlobalMapオブジェクトによって作成されるエントリには影響しない
- 2通りの方法で指定可能
  - 以下の例では、いずれもMyMapに対してデータの保持期間を60秒に設定している

インラインでセッション・ポリシーを定義する場合

```
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap", new MbGlobalMapSessionPolicy(60));
```

事前にセッション・ポリシーを定義する場合

```
MbGlobalMapSessionPolicy sessionPol = new MbGlobalMapSessionPolicy(60);
MbGlobalMap globalMap = MbGlobalMap.getGlobalMap("MyMap", sessionPol);
```

グローバル・マップ内のデータに保持期間を指定

- 組み込みグローバル・キャッシュでは、MbGlobalMapオブジェクトごとに個別の保持期間を設定することが可能
  - 同じグローバル・マップを参照する場合でも、複数のMbGlobalMapオブジェクトに対して異なる保持期間を設定できる

MbGlobalMapオブジェクトごとにTimeToLiveを設定する場合

```
MbGlobalMap map1 = MbGlobalMap.getGlobalMap("MyMap", new MbGlobalMapSessionPolicy(10));
map1.put("key1", "value");  // エントリには10秒の保持期間が設定される

MbGlobalMap map2 = MbGlobalMap.getGlobalMap("MyMap", new MbGlobalMapSessionPolicy(20));
map2.put("key2", "value");  // エントリには20秒の保持期間が設定される

MbGlobalMap map3 = MbGlobalMap.getGlobalMap("MyMap");
map3.put("key3", "value");  // エントリには保持期間は設定されない
```

グローバル・マップ内のデータに保持期間を指定する詳細方法については以下の「Thread behaviour with MbGlobalMapSessionPolicy's」を参照
https://community.ibm.com/community/user/blogs/aaron-gashi/2025/03/27/the-new-embedded-global-cache-in-ace-13-0-3-0

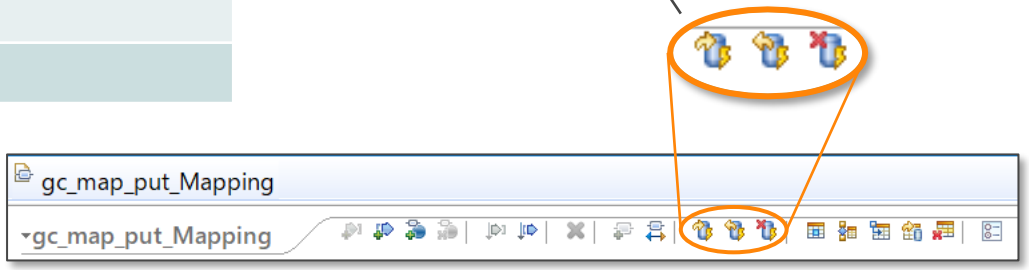
Mappingノードを使用したグローバル・キャッシュの利用方法

Mappingノードを使用してグローバル・キャッシュ機能を利用

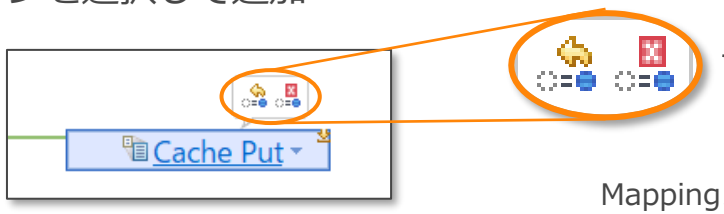
- Mappingノードを使用して、GUIでグローバル・マップとやり取りすることが可能
  - 前提として、メッセージ・マップ(.map)が作成されている必要がある
  - 以下のCache transformを使用して、マップ内のデータにアクセスすることが可能

Cache transform	機能説明
Cache Put	グローバル・キャッシュ内のマップにKeyとValueペアを追加
Cache Get	指定したKeyに対するValueをグローバル・キャッシュ内のマップから取得
Cache Remove	グローバル・キャッシュ内のマップからKeyとValueのペアを削除
Cache Return	Cache transformが成功した場合にValueを返す ※当資料では、Cache Get transformのスライド内で解説している
Cache Failure	Cache transformが失敗した場合に発生する例外を処理

左からCache Put、Cache Get、Cache Remove



- Cache Put、Cache Get、Cache Remove
  - マッピング・エディタ上部のツールバーにあるアイコンを選択して追加
- Cache Return、Cache Failure
  - 単体では使用できず、Cache Put、Cache Get、Cache Removeと併用するため、Cache transformを選択すると表示されるアイコンを選択して追加

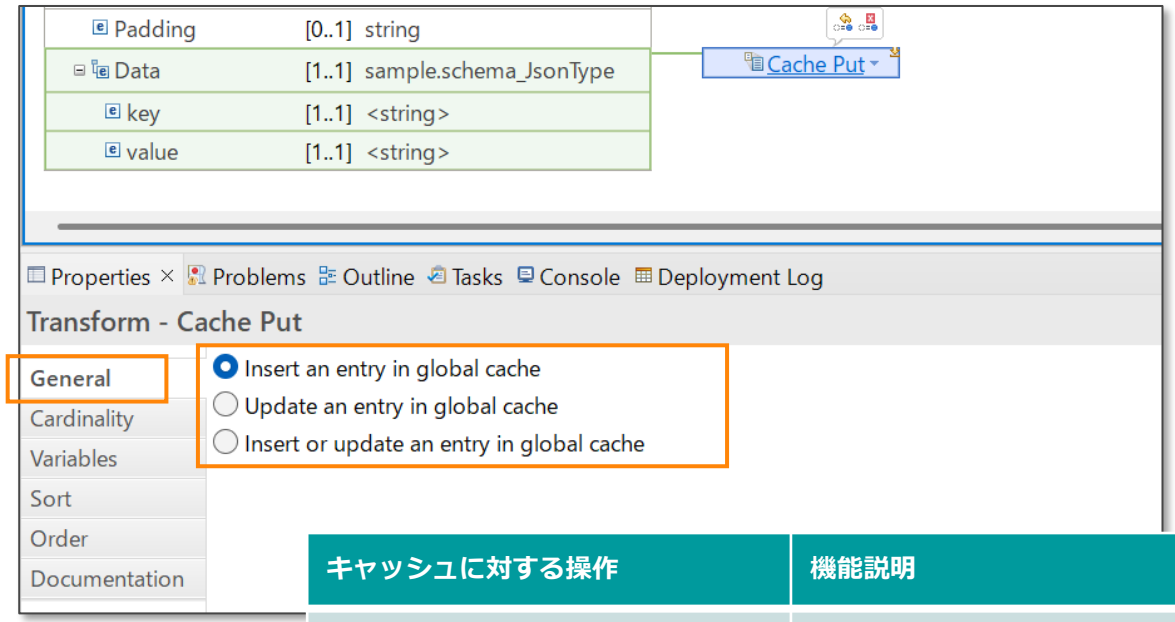


左からCache Return、Cache Failure

Mappingノードを使用してグローバル・マップ内のデータを操作する詳細方法については以下を参照
https://www.ibm.com/docs/en/app-connect/13.0.x?topic=maps-accessing-global-cache-by-using-mapping-node

グローバル・マップに対する操作

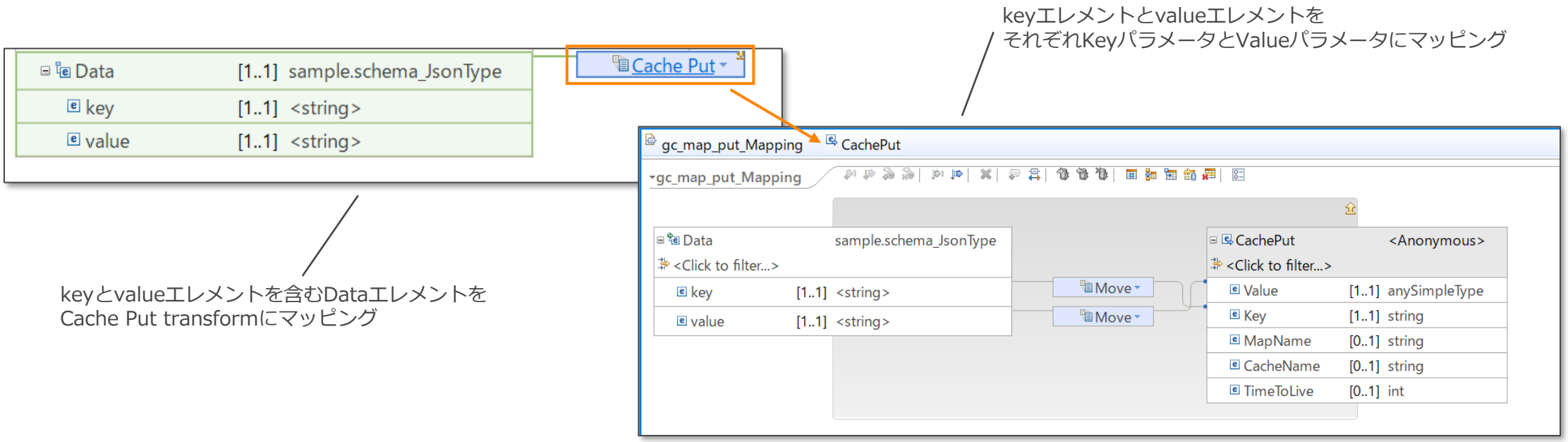
- Cache Put transformを使用して、マップに対してKeyとValueを格納・更新
  - Cache Putをクリックしてプロパティ設定のGeneralタブを選択し、実行する操作(Insert/Update/Upsert)を指定することが可能
    - デフォルトでは、Insert an entry in cacheが有効



キャッシュに対する操作	機能説明	エラー発生条件
Insert an entry in cache	新しいKeyとValueのペアをグローバル・マップに追加	同一Keyが既に存在する場合
Update an entry in cache	グローバル・マップ内の既存のKeyに対するValueを更新	Keyが存在しない場合
Insert or update an entry in cache	指定したKeyがグローバル・マップに存在するかどうかを確認し、 存在する場合はそのValueを更新し、存在しない場合は新しいKeyとValueのペアを追加	なし（Keyの存在有無に応じて処理）

グローバル・マップに対する操作（Cache Put）

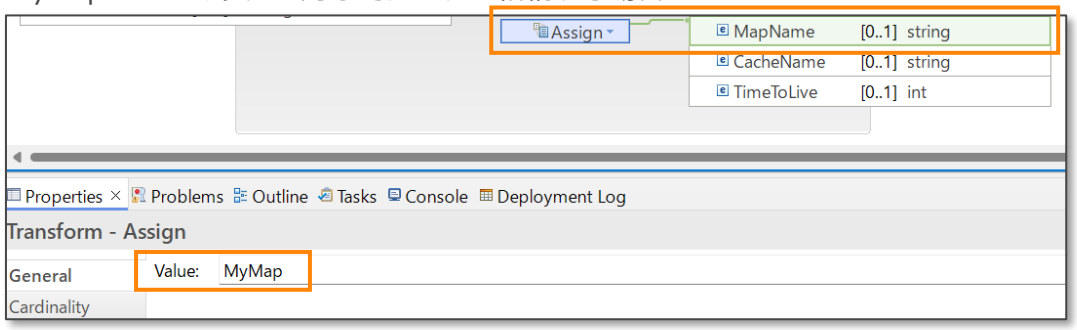
- Cache Put transformを使用して、マップに対してKeyとValueを格納・更新 (続き)
  - Cache Put transformをダブルクリックしてネストされたメッセージ・マップを編集
    - 出力エレメントは、Cache Putを実行するために必要な事前定義されたパラメータ
    - 入力エレメントをCache Put transformにマッピングする場合、それらのエレメントはネストされたマップの入力エレメントとしてマッピング可能
    - 以下のいずれかの方法で、Cache Put transformのパラメータに値を設定
      - 入力エレメントをパラメータにマッピング
      - Assign transformを使用してパラメータに固定値を割り当て
      - ユーザー定義プロパティを使用してパラメータに値を割り当て



グローバル・マップに対する操作（Cache Put）

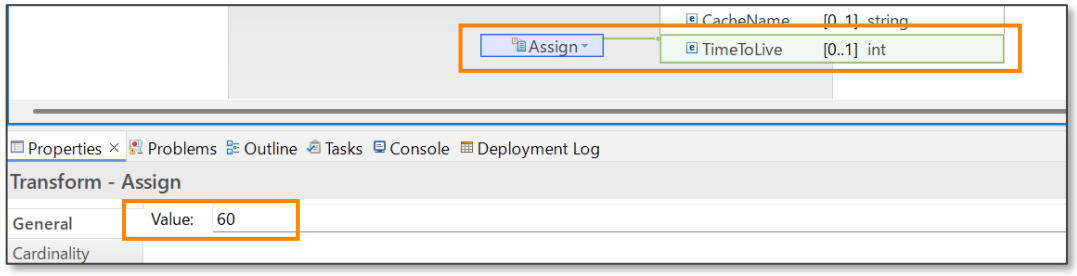
- Cache Put transformを使用して、マップに対してKeyとValueを格納・更新 (続き)
  - MapNameパラメータにグローバル・マップの名前を割り当て、指定したマップに対してKeyとValueのペアを格納することが可能
    - 指定しない場合は、デフォルトのマップ(SYSTEM.BROKER.DEFAULTMAP)が使用される
    - 存在しないマップを指定すると、KeyとValueのペアが追加される前にマップが新規に作成される

MyMapというマップに対してデータを格納する場合



  - TimeToLiveパラメータに、データが自動的に削除されるタイミング(保持期間)を割り当てることも可能
    - 秒単位で指定

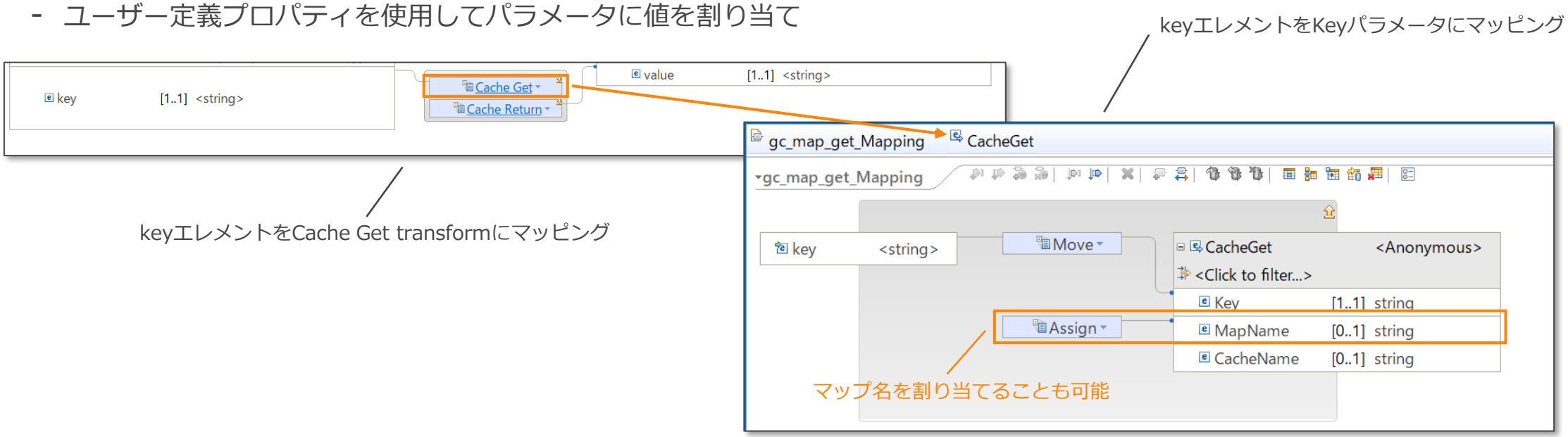
60秒後にマップからデータを削除する場合



Cache Put transformについての詳細は以下リンクを参照
https://www.ibm.com/docs/en/app-connect/13.0.x?topic=editor-cache-put

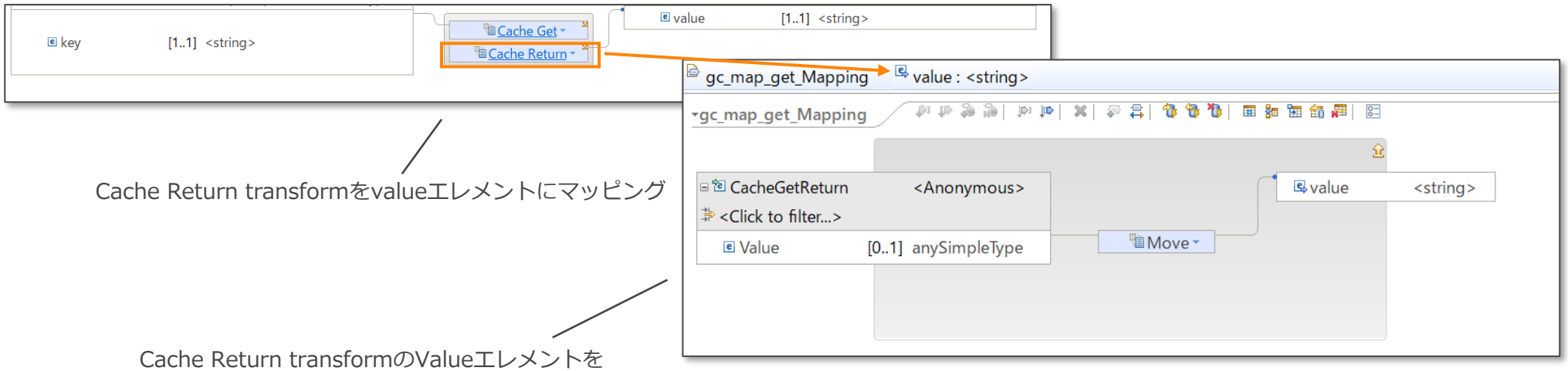
グローバル・マップに対する操作（Cache Get、Cache Return）

- Cache Get transformを使用して、マップ内のKeyに対するValueを取得
  - Cache Get transformを追加すると、自動的にCache Return transformが追加される
  - Cache Get transformをダブルクリックしてネストされたマップを編集
    - 出力エレメントは、Cache Getを実行するために必要な事前定義されたパラメータ
    - 入力エレメントをCache Get transformにマッピングする場合、それらのエレメントはネストされたマップの入力エレメントとしてマッピング可能
    - 以下のいずれかの方法で、Cache Get transformのパラメータに値を指定
      - 入力エレメントをパラメータにマッピング
      - Assign transformを使用してパラメータに固定値を割り当て
      - ユーザー定義プロパティを使用してパラメータに値を割り当て



グローバル・マップに対する操作（Cache Get、Cache Return）

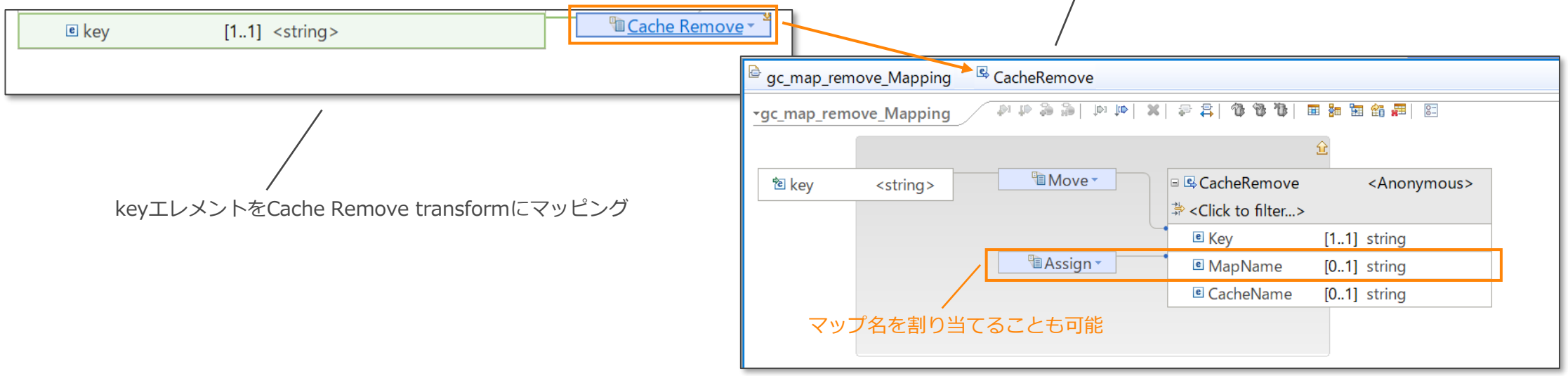
- Cache Get transformを使用して、マップ内のKeyに対するValueを取得 (続き)
  - Cache Return transformは、Cache Put、Cache Get、またはCache Remove transformと併用することで、それらのCache transformが正常完了した場合に、Keyに対するValueを返すことが可能
    - 当資料では、Cache Get transformと併用する場合について解説する
  - Cache Return transformをダブルクリックしてネストされたマップを編集
    - 入力エレメントをCache Return transformにマッピングして、Cache Return transformを複数の出力エレメントにマッピングすることで、ネストされたマップ内でValueの生成や変換を行い、それを接続先の出力エレメントにマッピングすることが可能
    - Cache Get transformの正常実行時に取得されるデータを受け取るための事前定義された入力エレメントが含まれる



Cache Get transform、Cache Return transformについての詳細は以下リンクを参照
・Cache Get: https://www.ibm.com/docs/en/app-connect/13.0.x?topic=editor-cache-get
・Cache Return: https://www.ibm.com/docs/en/app-connect/13.0.x?topic=editor-cache-return

グローバル・マップに対する操作（Cache Remove）

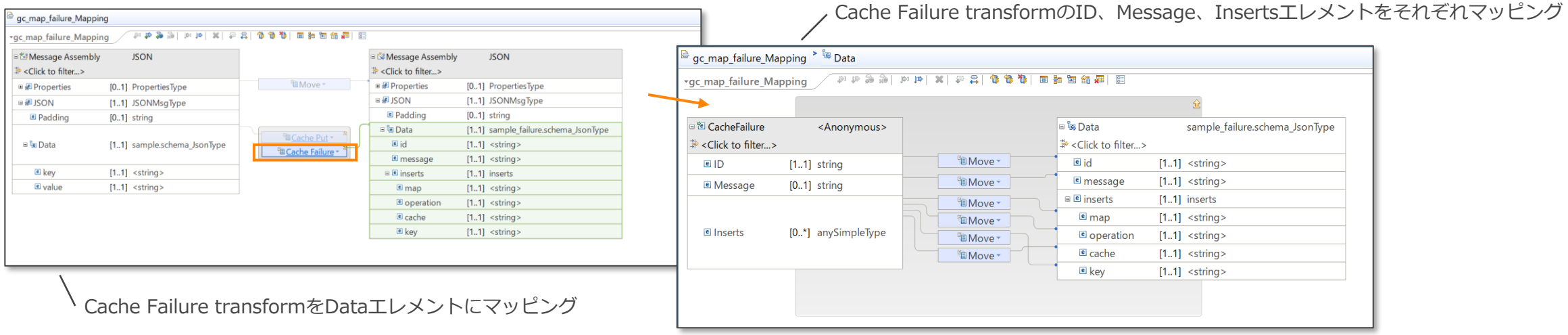
- Cache Remove transformを使用して、マップ内のKeyとValueのペアを削除
  - Cache Remove transformを追加後、ダブルクリックしてネストされたマップを編集
    - 出力エレメントは、Cache Removeを実行するために必要な事前定義されたパラメータ
    - Cache Remove transformに入力エレメントをマッピングしている場合、それらのエレメントはネストされたマップの入力エレメントとしてマッピング可能
    - 以下のいずれかの方法で、Cache Remove transformのパラメータに値を指定
      - 入力エレメントをパラメータにマッピング
      - Assign transformを使用してパラメータに固定値を割り当て
      - ユーザー定義プロパティを使用してパラメータに値を割り当て



Cache Remove transformについての詳細は以下リンクを参照
https://www.ibm.com/docs/en/app-connect/13.0.x?topic=editor-cache-remove

グローバル・マップに対する操作（Cache Failure）

- Cache Failure transformを使用して、Cache transformで発生する例外を処理
  - Cache Put、Cache Get、またはCache Removeと併用することで、それらのCache transformで発生した例外をCache Failure transformがキャッチして処理することが可能
  - Cache Failure transformを使用しない場合にCache transformで例外が発生した場合、メッセージ・マップの処理は停止し、例外はMappingノードのFailureターミナルに伝搬される
  - Cache Failure transformを追加後、ダブルクリックしてネストされたマップを編集
    - Cache Failure transformを出力エレメントにマッピングすると、Cache transformで発生した例外をどのように処理するかを定義できる
      - 入力エレメントをCache Failure transformにマッピングして、Cache Failure transformを複数の出力エレメントにマッピングすることで、ネストされたマップ内でValueの生成や変換を行い、それを接続先の出力エレメントにマッピングすることも可能
    - Cache transformで発生した例外を受け取るための事前定義された入力エレメントが含まれる



グローバル・マップに対する操作（Cache Failure）

- Cache Failure transformを使用して、Cache transformで発生する例外を処理 (続き)
  - Cache Failure transformを利用する際の注意点
    - Cache Failure transformがCache transformと併用されていて、かつ1つ以上の出力先にマッピングされている場合に例外がキャッチされ、Cache Failure transformで処理される
    - Cache Failure transformを追加していても、出力先にマッピングされていない場合やネストされたマップ内でマッピングが定義されていない場合は、以下の動作となる
      - 発生した例外はCache Failure transformによってキャッチされるが、無視される
      - メッセージ・マップに警告が表示される

Cache Failure transformの構成と実行時の挙動

Cache Failure transformの構成	結果
Cache Failure transformが1つ以上の出力オブジェクトに接続されている場合	例外はCache Failure transformによってキャッチされ、処理される
Cache Failure transformが出力オブジェクトに接続されていない場合	• 例外はCache Failure transformによってキャッチされるが、無視される • メッセージ・マップに警告が表示される
Cache Failure transformを使用せず、Cache transformで例外が発生した場合	例外はMappingノードから後続のメッセージ・フローに渡される

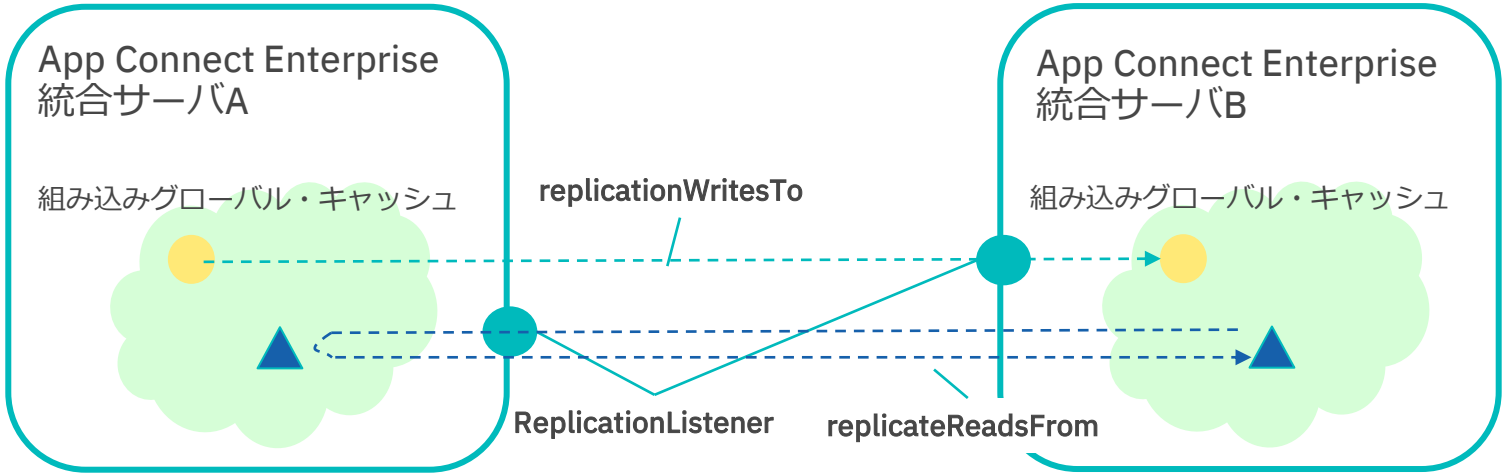
Cache Failure transformについての詳細は以下リンクを参照
https://www.ibm.com/docs/en/app-connect/13.0.x?topic=editor-cache-failure

複数統合サーバ間でのグローバル・キャッシュ共有

複数統合サーバ間でのキャッシュ情報の共有

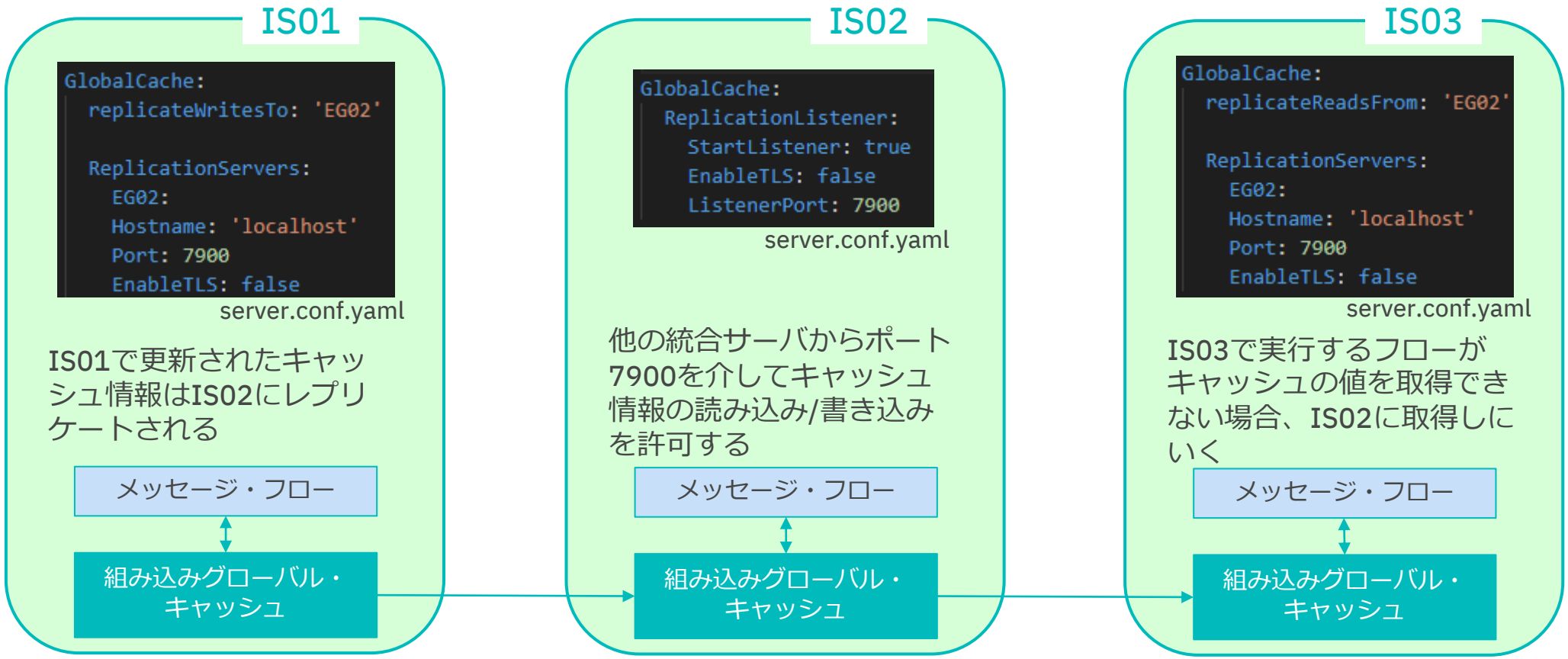
- 複数統合サーバ間で組み込みグローバル・キャッシュ情報の共有が可能
  - 主な設定項目 (server.conf.yaml)
    - ResourceManagers>GlobalCache セクションで設定

項目	用途
replicationWritesTo	自身の統合サーバ上のフローから更新されたキャッシュ情報を、登録した統合サーバに非同期でレプリケーションする （複数統合サーバの指定が可能）
ReplicationListener	指定したポートを介して、他の統合サーバからのREAD/WRITEを許可する
replicateReadsFrom	自身のキャッシュ内に必要な情報がない時に、登録した統合サーバにキャッシュ情報を同期的に問い合わせる （複数の統合サーバを指定した場合は、順番に問い合わせる）



複数統合サーバ間でのキャッシュ情報の共有

■ 3台構成の例



◆ 上記構成での基本的な動作

- IS01で更新したキャッシュ情報はIS02に連動される
- IS03でキャッシュ情報を利用するフローの稼働時に、ローカルにキャッシュ情報がない場合は、IS02に問い合わせにいく

複数統合サーバ間でのキャッシュ情報の共有

- キャッシュ情報のValueのコンシステンシーを担保するには注意が必要
  - replicationWritesToを一方向で設定していると、レプリケーション先でKeyのValueが更新されると、レプリケーション元とレプリケーション先でKeyに対して異なるValueを持つ
    - 相互にreplicationWritesToを設定するか、外部にキャッシュサーバーを立てる必要がある
  - replicationWritesToで連動されるキャッシュ・データは自身の統合サーバ上で稼働するメッセージ・フローが更新した情報のみ
    - 他統合サーバからreplicationWritesToで連動されたキャッシュ・データは、自身のreplicationWritesToに設定された統合サーバには連動されない
  - replicateReadsFromでキャッシュ・データを取得後、取得元でキャッシュ・データが更新されても気づかない
    - 取得元での値更新に連動して、古いKeyを削除して再取得するような仕組みが必要
  - replicateReadsFromを設定していれば、統合サーバダウンでキャッシュ情報がクリアされても、キャッシュ情報利用時に他の統合サーバから必要なキャッシュ情報を取得することができる

複数統合サーバ間で、同一のキャッシュ情報を保持する要件がある場合には、
replicateReadsFromとreplicationWritesToにキャッシュを共有する全統合サーバ名（自分以外）を登録する

Tips

- 組み込みグローバル・キャッシュ機能を使用する場合は、V13.0.3.1以上を使用する
  - V13.0.3.0ではreplicateReadsFromの設定がうまく動作しない（ APAR IT47994 ）
- キャッシュ情報を共有する統合サーバのいずれかが管理コマンドでキャッシュをクリアしても、他の統合サーバのキャッシュ情報には影響しない
  - キャッシュ情報をremove()した場合は、「replicationWritesTo」でキャッシュ情報を共有するすべての統合サーバ上のキャッシュから該当のキー情報は削除される
- リスナーに届いたリクエストはプロパゲートされない
  - replicationWritesToで連動されるキャッシュ・データは自身の統合サーバ上で稼働するメッセージ・フローが更新した情報のみ
    - 他統合サーバからreplicationWritesToで連動されたキャッシュ・データは、連動されない
  - 自身の統合サーバ上で稼働するメッセージ・フローが取得しようとしたキャッシュ情報がローカルになければreplicateReadsFromに設定された統合サーバに聞きにいく
    - replicateReadsFromに設定された統合サーバ（例EG01）上のキャッシュにデータがない場合、その先の統合サーバ（EG01のreplicateReadsFromに設定された統合サーバ）には問い合わせない
- キャッシュ情報のレプリケーションの書き込みや読み込みが失敗しても、フローの実行には影響しない
  - Readに失敗してもデータなしとして扱われる

グローバル・キャッシュの運用管理

管理コマンド　キャッシュ情報の表示

- ibmint display cacheコマンド
  - ◆ 組み込みグローバル・キャッシュの情報を表示するコマンド
    - 利用可能なマップのリストを表示する

```
ibmint display cache --integration-node <統合ノード名> --integration-server <統合サーバ名> --all-maps
```

- 実行例

```
$ ibmint display cache --integration-node BK1303 --integration-server EG01 --all-maps
BIP15355I: The server contains '1' maps:
BIP15349I: The map 'GctestMap' contains 2 keys, and is using 33 Bytes of memory.
```

- 特定のマップの情報を確認する

```
ibmint display cache --integration-node <統合ノード名> --integration-server <統合サーバ名>
  --map-name <マップ名>
```

- 実行例

```
$ ibmint display cache --integration-node BK1303 --integration-server EG01 --map-name GctestMap
BIP15349I: The map 'GctestMap' contains 2 keys, and is using 33 Bytes of memory.
```

コマンド詳細（https://www.ibm.com/docs/en/app-connect/13.0.x?topic=commands-ibmint-display-cache-command）

管理コマンド　キャッシュ情報のクリア

- ibmint clear cacheコマンド
  - 組み込みグローバル・キャッシュの情報を削除するコマンド
    - 指定したマップのKey情報を削除する

```
ibmint clear cache --integration-node <統合ノード名> --integration-server <統合サーバ名>
  --map-name <マップ名>
```

- 実行例

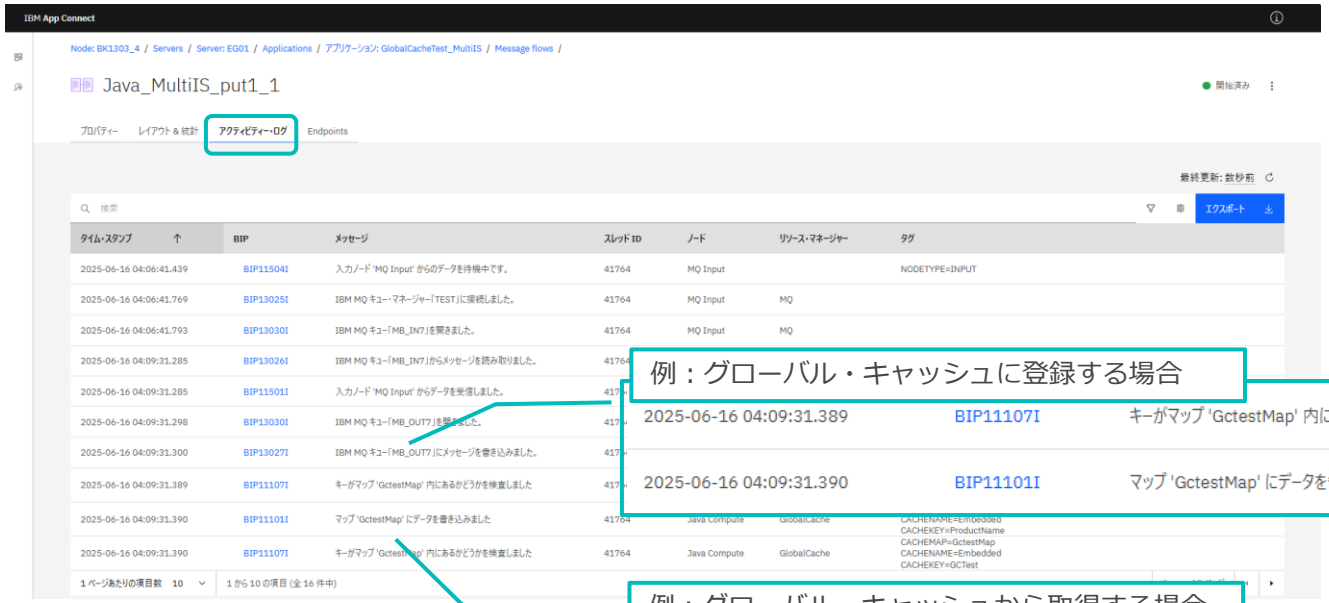
```
$ ibmint clear cache --integration-node BK1303_4 --integration-server EG03 --map-name GctestMap
BIP15358I: The map 'GctestMap' has had all data entries cleared from this server.

$ ibmint display cache --integration-node BK1303_4 --integration-server EG03 --all-maps
BIP15355I: The server contains '1' maps:
BIP15349I: The map 'GctestMap' contains 0 keys, and is using 0 Bytes of memory.
```

コマンド詳細（https://www.ibm.com/docs/en/app-connect/13.0.x?topic=commands-ibmint-clear-cache-command）

アクティビティ・ログ

- アクティビティ・ログはフローの処理状況や外部リソースへのアクセス状況を簡易的に示すログ
  - 稼働フローからグローバル・キャッシュへの書き込みや読み込みが発生するとログを出力
  - 内部的にreplicationWritesToなどでキャッシュがレプリケートされた情報などは出力されない
    - あくまでもフローからグローバル・キャッシュにアクセスした情報のみ



例：グローバル・キャッシュに登録する場合

2025-06-16 04:09:31.389	BIP11107I	キーがマップ 'GctestMap' 内にあるかどうかを検査しました	41764	Java Compute	GlobalCache	CACHEMAP=GctestMap CACHENAME=Embedded CACHEKEY=ProductName
2025-06-16 04:09:31.390	BIP11101I	マップ 'GctestMap' にデータを書き込みました	41764	Java Compute	GlobalCache	CACHEMAP=GctestMap CACHENAME=Embedded CACHEKEY=ProductName

例：グローバル・キャッシュから取得する場合

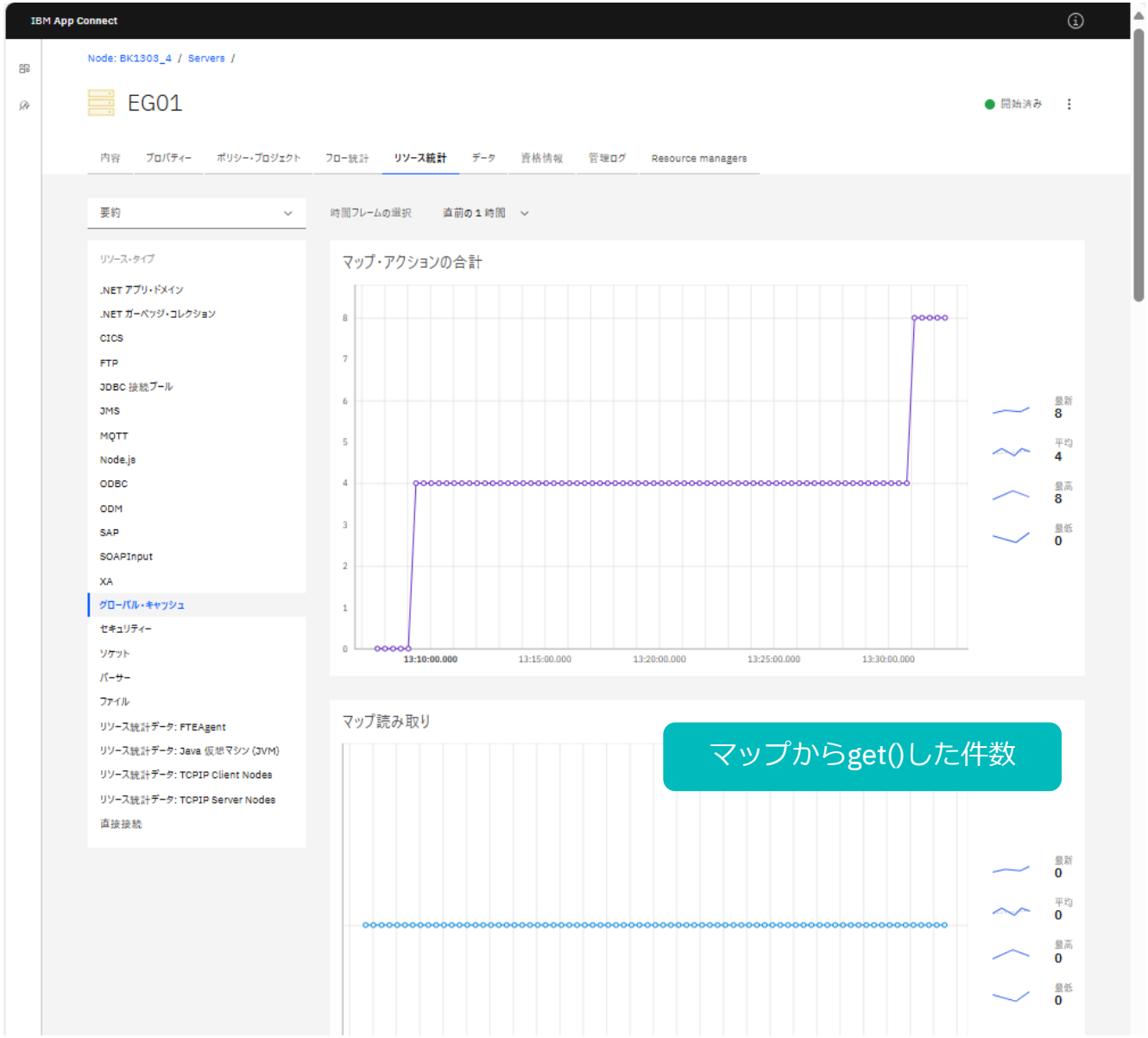
2025-06-16 04:19:37.596	BIP11103I	マップ 'GctestMap' からデータを取得しました	9896	Java Compute	GlobalCache	CACHEMAP=GctestMap CACHENAME=Embedded CACHEKEY=ProductName
2025-06-16 04:19:37.597	BIP11103I	マップ 'GctestMap' からデータを取得しました	9896	Java Compute	GlobalCache	CACHEMAP=GctestMap CACHENAME=Embedded CACHEKEY=Version

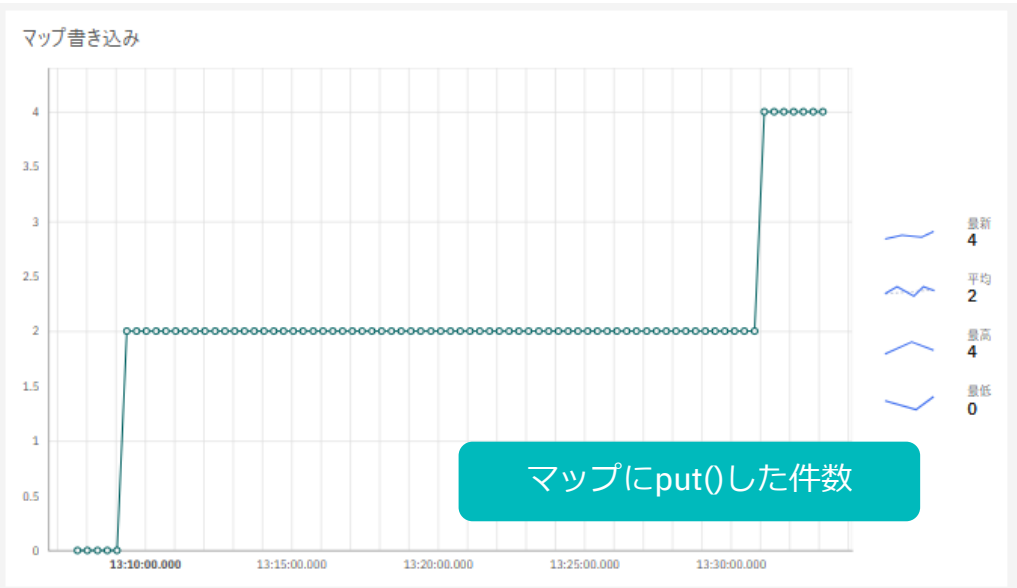
リソース統計

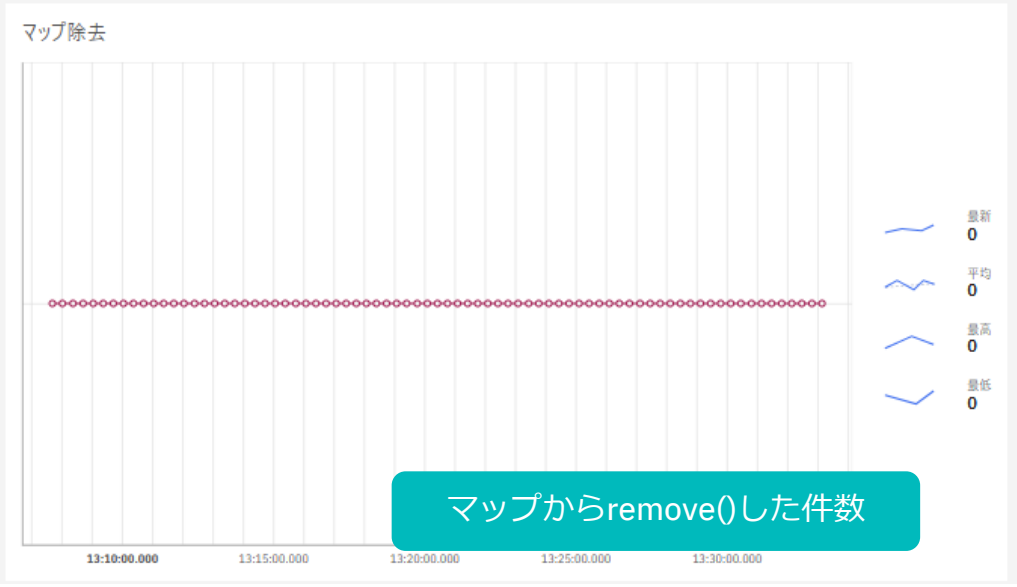
- リソース統計でグローバル・キャッシュの利用状況を確認可能
  - グローバル・キャッシュ・リソース・マネージャーは組み込みグローバル・キャッシュおよび外部WebSphere eXtreme Scale Grid内の統計情報が確認可能
  - 以下の情報を確認可能

取得可能な情報	説明
マップ・アクションの合計	フローから実行したマップ操作の総数（読み取り、書き込み、削除、キー検査を含む）
マップ読み取り	フローからマップへの読み取り操作をした回数
マップ書き込み	フローからマップへの書き込み操作をした回数
マップ除去	フローからマップへの削除操作をした回数
失敗したアクション	フローからのマップ操作のうち、失敗した回数
使用されているマップ	フローによって使用されたマップの数
接続失敗	この統合サーバーから指定したキャッシュへの接続が失敗した回数
接続	この統合サーバーから指定したキャッシュへの接続が正常に確立された回数

（参考）リソース統計画面


IBM App Connect
Node: BK1303_4 / Servers /
EG01
開始済み
内容
プロパティー
ポリシー・プロジェクト
フロー統計
リソース統計
データ
資格情報
管理ログ
Resource managers
要約
時間フレームの選択
直前の 1 時間
リソース・タイプ
.NET アプリ・ドメイン
.NET ガーベッジ・コレクション
CICS
FTP
JDBC 接続プール
JMS
MQTT
Node.js
ODBC
ODM
SAP
SOAPInput
XA
グローバル・キャッシュ
セキュリティー
ソケット
パーサー
ファイル
リソース統計データ: FTEAgent
リソース統計データ: Java 仮想マシン (JVM)
リソース統計データ: TCPIP Client Nodes
リソース統計データ: TCPIP Server Nodes
直接接続
マップ・アクションの合計
最新 8
平均 4
最高 8
最低 0
13:10:00.000
13:15:00.000
13:20:00.000
13:25:00.000
13:30:00.000
マップ読み取り
マップからget()した件数
最新 0
平均 0
最高 0
最低 0


マップ書き込み
最新 4
平均 2
最高 4
最低 0
13:10:00.000
13:15:00.000
13:20:00.000
13:25:00.000
13:30:00.000
マップにput()した件数


マップ除去
最新 0
平均 0
最高 0
最低 0
13:10:00.000
13:15:00.000
13:20:00.000
13:25:00.000
13:30:00.000
マップからremove()した件数

（参考）リソース統計画面　つづき