CA Application Performance Management

IBM CICS Transaction Gateway ガイド

リリース 9.5

CA Technologies 製品リファレンス

このドキュメントは、以下の CA Technologies 製品および機能に関するものです。

- CA Application Performance Management （CA APM）
- CA Application Performance Management ChangeDetector （CA APM ChangeDetector）
- CA Application Performance Management ErrorDetector （CA APM ErrorDetector）
- CA Application Performance Management for CA Database Performance （CA APM for CA Database Performance）
- CA Application Performance Management for CA SiteMinder® （CA APM for CA SiteMinder®）
- CA Application Performance Management for CA SiteMinder® Application Server Agents （CA APM for CA SiteMinder® ASA）
- CA Application Performance Management for IBM CICS Transaction Gateway （CA APM for IBM CICS Transaction Gateway）
- CA Application Performance Management for IBM WebSphere Application Server （CA APM for IBM WebSphere Application Server）
- CA Application Performance Management for IBM WebSphere Distributed Environments （CA APM for IBM WebSphere Distributed Environments）
- CA Application Performance Management for IBM WebSphere MQ （CA APM for IBM WebSphere MQ）
- CA Application Performance Management for IBM WebSphere Portal （CA APM for IBM WebSphere Portal）
- CA Application Performance Management for IBM WebSphere Process Server （CA APM for IBM WebSphere Process Server）
- CA Application Performance Management for IBM z/OS® （CA APM for IBM z/OS®）
- CA Application Performance Management for Microsoft SharePoint （CA APM for Microsoft SharePoint）
- CA Application Performance Management for Oracle Databases （CA APM for Oracle Databases）

- CA Application Performance Management for Oracle Service Bus （CA APM for Oracle Service Bus）
- CA Application Performance Management for Oracle WebLogic Portal （CA APM for Oracle WebLogic Portal）
- CA Application Performance Management for Oracle WebLogic Server （CA APM for Oracle WebLogic Server）
- CA Application Performance Management for SOA （CA APM for SOA）
- CA Application Performance Management for TIBCO BusinessWorks （CA APM for TIBCO BusinessWorks）
- CA Application Performance Management for TIBCO Enterprise Message Service （CA APM for TIBCO Enterprise Message Service）
- CA Application Performance Management for Web Servers （CA APM for Web Servers）
- CA Application Performance Management for webMethods Broker （CA APM for webMethods Broker）
- CA Application Performance Management for webMethods Integration Server （CA APM for webMethods Integration Server）
- CA Application Performance Management Integration for CA CMDB （CA APM Integration for CA CMDB）
- CA Application Performance Management Integration for CA NSM（CA APM Integration for CA NSM）
- CA Application Performance Management LeakHunter （CA APM LeakHunter）
- CA Application Performance Management Transaction Generator （CA APM TG）
- CA Cross-Enterprise Application Performance Management
- CA Customer Experience Manager （CA CEM）
- CA Embedded Entitlements Manager （CA EEM）
- CA eHealth® Performance Manager （CA eHealth）
- CA Insight™ Database Performance Monitor for DB2 for z/OS®
- CA Introscope®
- CA SiteMinder®
- CA Spectrum® Infrastructure Manager （CA Spectrum）

- CA SYSVIEW® Performance Management （CA SYSVIEW）

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

目次

第 1 章: CTG の拡張機能の概要

IBM® CICS® Transaction Gateway の拡張機能により、CA Introscope® 管理者が CTG クライアント インターフェースを監視できるようになります。 これらのインターフェースは、Java と J2EE のアプリケーション、および CTG サーバによって使用されます。

このセクションには、以下のトピックが含まれています。

CTG の拡張機能について

CTG の拡張機能は、CICS Transaction Gateway Server （CTG）製品および CTG クライアント Java のリアルタイム監視を提供します。 また、CTG の拡張機能は、CTG を介して CICS （Customer Information Control System）を呼び出す WebSphere CTG クライアント アプリケーションを監視します。 拡張機能は、CTG サーバ自体と CTG サーバを呼び出すインスツルメントされたクライアントの両方を監視します。 CTG サーバは、Java と J2EE のフロントエンドと CICS Transaction Server バックエンドの中間にあります。 クライアントの監視には、JCA と Base API クライアント アプリケーションの両方の監視が含まれます。

注: CTG サーバは、CTG デーモンとも呼ばれます。 容易にサーバと理解できるようにするため、このガイドでは CTG サーバという名称を使用します。ただし、ほかの技術的なドキュメントでは CTG デーモンと呼ばれています。

CTG の拡張機能は、Java アプリケーション、WebSphere アプリケーション サーバ、および CTG 自体で発生する可能性があるボトルネックを、Introscope ユーザが関係付ける場合、および分離する場合に役立ちます。 CTG の拡張機能は、トランザクションを監視し、Java または J2EE フロントエンド、CTG サーバ ミドルウェア、または CICS バックエンドにボトルネックが存在しているかどうかを判断するために使用できる詳細なメトリックを提供します。

CTG の拡張機能をインストールすると、以下を表示できます。

- CTG を使用するアプリケーションのパフォーマンスおよびアクティビティのグラフ。 これらの項目は、アプリケーションが CTG クライアント ベース クラス、CCF インターフェース、または JCA インターフェースを使用して CTG を呼び出すかどうかにかかわらず利用可能です。
- すぐに使えるダッシュボードを介した CTG の稼働状況の高度な概要。
- パフォーマンスの階層化ビューおよび履歴ビュー。

CTG の拡張機能のシステム要件

CTG システム要件の全リストについては、CA APM マニュアル選択メニューの「Application Performance Management Compatibility Guide」を参照してください。 製品互換性マトリックスは、サポートされているすべてのオペレーティング環境のリストを提供します。

注: CTG の拡張機能の日本語版は、Introscope 9.0.5 でのみ動作します。

CTG の拡張機能のコンポーネント

以下の CTG の拡張機能のコンポーネントにより、Introscope とのやり取りが可能になります。

コンポーネント	説明
PPCTGServer_ ManagementModule.jar *PPCTGClient_ ManagementModule.jar*	CTG の拡張機能の管理モジュールは、メトリックに適用される監視およびレポートのロジックを定義します。これらのメトリックには、Workstation に示されるダッシュボード、およびメトリックしきい値を定義するアラートが含まれます。
PPCTGRequestExit_ ManagementModule.jar	CTG の拡張機能は、Request Exit インスツルメンテーションに対してこのモジュールを使用します。

PPCTGClient-full.pbd *PPCTGClient-typical.pbd*	Introscope 対応 CTG コンポーネントがエージェントにレポートするメトリックを制御する ProbeBuilder ディレクティブ ファイル。 PPCTGClient-full.pbd ファイルを使用すると、すべてのメトリックが Introscope に表示されます。 すべてのメトリックを表示するオーバーヘッドが発生しないようにする場合は、PPCTGClient-typical.pbd ファイルを使用します。
PPCTGServer-full.pbd *PPCTGServer-minimal.pbd* *PPCTGServer-typical.pbd*	Introscope 対応 CTG コンポーネントがエージェントにレポートするメトリックを制御する ProbeBuilder ディレクティブ ファイル。 PPCTGServer-full.pbd ファイルを使用すると、すべてのメトリックが Introscope に表示されます。 すべてのメトリックを表示するオーバーヘッドが発生しないようにする場合は、PPCTGServer-minimal.pbd ファイルまたは PPCTGServer-typical.pbd ファイルを使用します。
PPCTGTranTrace.pbd	PPCTGTranTrace.pbd ディレクティブ ファイルは、CTG トランザクションを追跡し、フロントエンドとバックエンドのトランザクションを関連付ける機能を提供します。
PPCTGAgent.jar	エージェントから Enterprise Manager に CTG サーバ メトリックをレポート可能にする拡張機能。CTG バージョン 6.x および 7.x の両方で有効。
PPCTGRequestExit.jar *PPCTGbmonitor.jar*	エージェントから Enterprise Manager に CTG 8.x サーバ メトリックをレポート可能にする拡張機能。起動するには、PPrunGlobalStats バッチ ジョブが必要です。CTG バージョン 7.x および 8.x で有効。
ctg.typeviewers.xml	ツリーで CTG コンポーネントが選択されている場合に、Investigator でタブ選択可能な CTG 固有のビューを定義する Enterprise Manager 拡張機能。
ctg.requestexit.typerviewers.xml	CTG-Request Extension 固有のビューを定義する Enterprise Manager 拡張機能。これらのビューは、ツリーで CTG Request Exist コンポーネントが選択されている場合、Investigator でタブ選択可能です。
ChangeDetector-config.xml	CTG に対するデフォルトの ChangeDetector 監視設定エントリが含まれている XML ファイル。

CTG_Tran_Trace_Template.profile	このファイルには、CTG の追跡の特定のメトリックを収集するために IntroscopeAgent.profile に含めることができる CTG トランザクション追跡プロパティが含まれます。 詳細については、「CTG Transaction Tracer (P. 66)」を参照してください。
PPrunGlobalStats.bat *PPrunGlobalStats.sh*	この要件は、Windows プラットフォームに対するものです。PPrunGlobalStats.bat ファイル内で CTG_CLASSES 変数と ISCOPE_AGENT 変数を指定し、PPCTGbmonitor.jar に組み込まれているグローバル統計情報の監視を実行できます。 CTG は Windows でサービスとして実行されているため、この要件は Windows に必須です。 Windows で実行するには、個別のプロセスを使用してグローバル統計情報を抽出する必要があります。 同等のファイルが Unix プラットフォームにも存在します。 PPrunGlobalStatus.sh は上記の .bat ファイルの Unix/Linux バージョンです。

CTG の拡張機能と CA Introscope® 環境

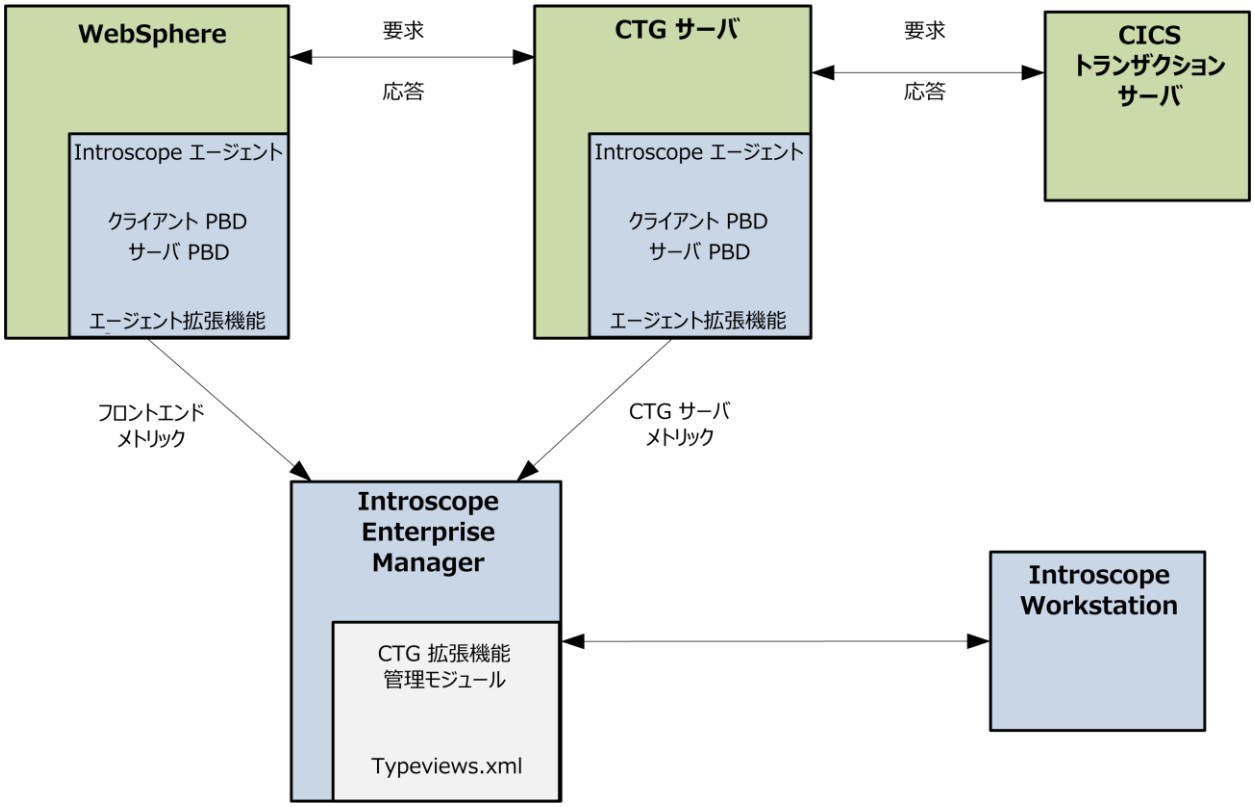
CTG の拡張機能の展開には、環境に応じて 1 台以上のコンピュータが必要な場合があります。 展開は PBD または Request Exit の使用により実行されます。 展開に含まれる CA Introscope® コンポーネントの ProbeBuilder ディレクティブには、以下のものがあります。

- CTG クライアント ライブラリを有効にするために使用される ProbeBuilder ディレクティブ。 これらは、アプリケーションをホストするコンピュータ、および CTG サーバをホストするコンピュータの両方に展開されます。
- CTG サーバ ライブラリを有効にするために使用される CTG エージェント拡張機能および ProbeBuilder ディレクティブの拡張。 これらは、CTG サーバをホストするコンピュータ、およびアプリケーションをホストするコンピュータの両方に展開されます。
- 管理モジュール。これらは、（通常、別のコンピュータ上の）Enterprise Manager に展開されます。

重要: ProbeBuilder と Request Exit の*両方*で CTG を有効にすることは推奨されていません。 これを行うと、メトリックが重複して CPU 使用率が高くなる可能性があります。

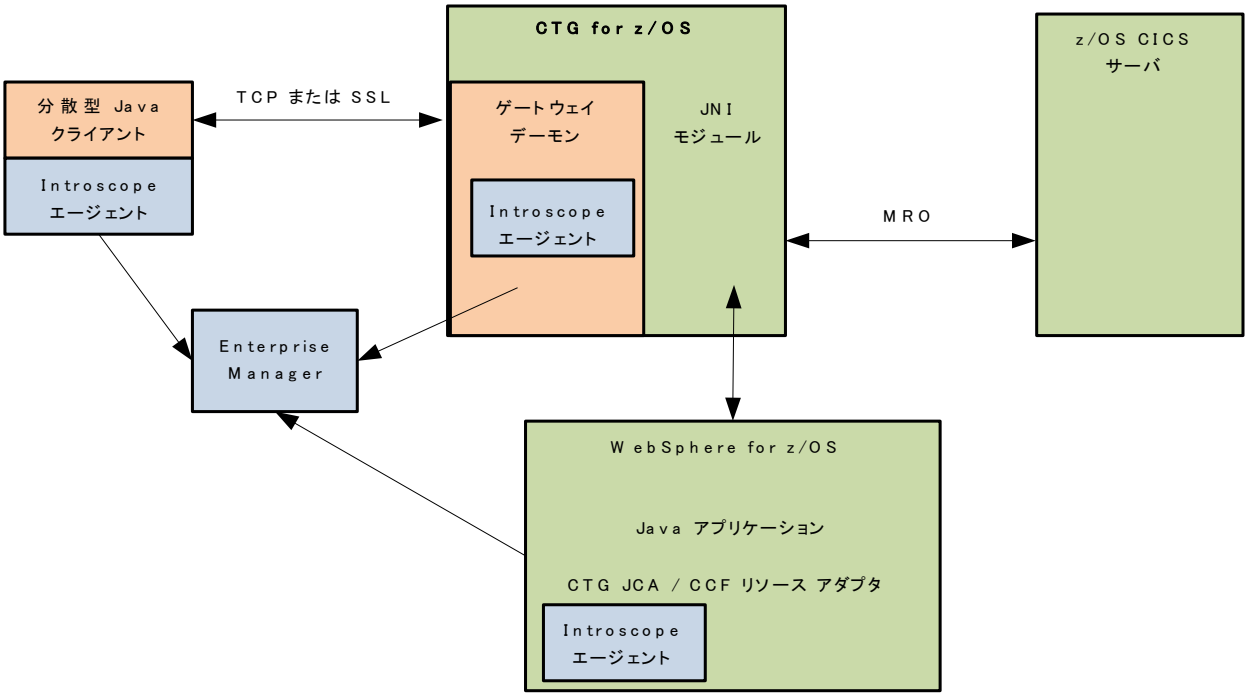
ProbeBuilder ディレクティブの展開

以下の図は、ProbeBuilder ディレクティブが展開に使用される場合の全体的なインタラクションを示しています。 CTG の拡張機能は、z/OS 環境および分散型（Windows/Linux/UNIX）環境の両方で動作します。



z/OS 上の CTG コンポーネント

以下の図は、z/OS 上の CTG コンポーネント間のインタラクションを示しています。



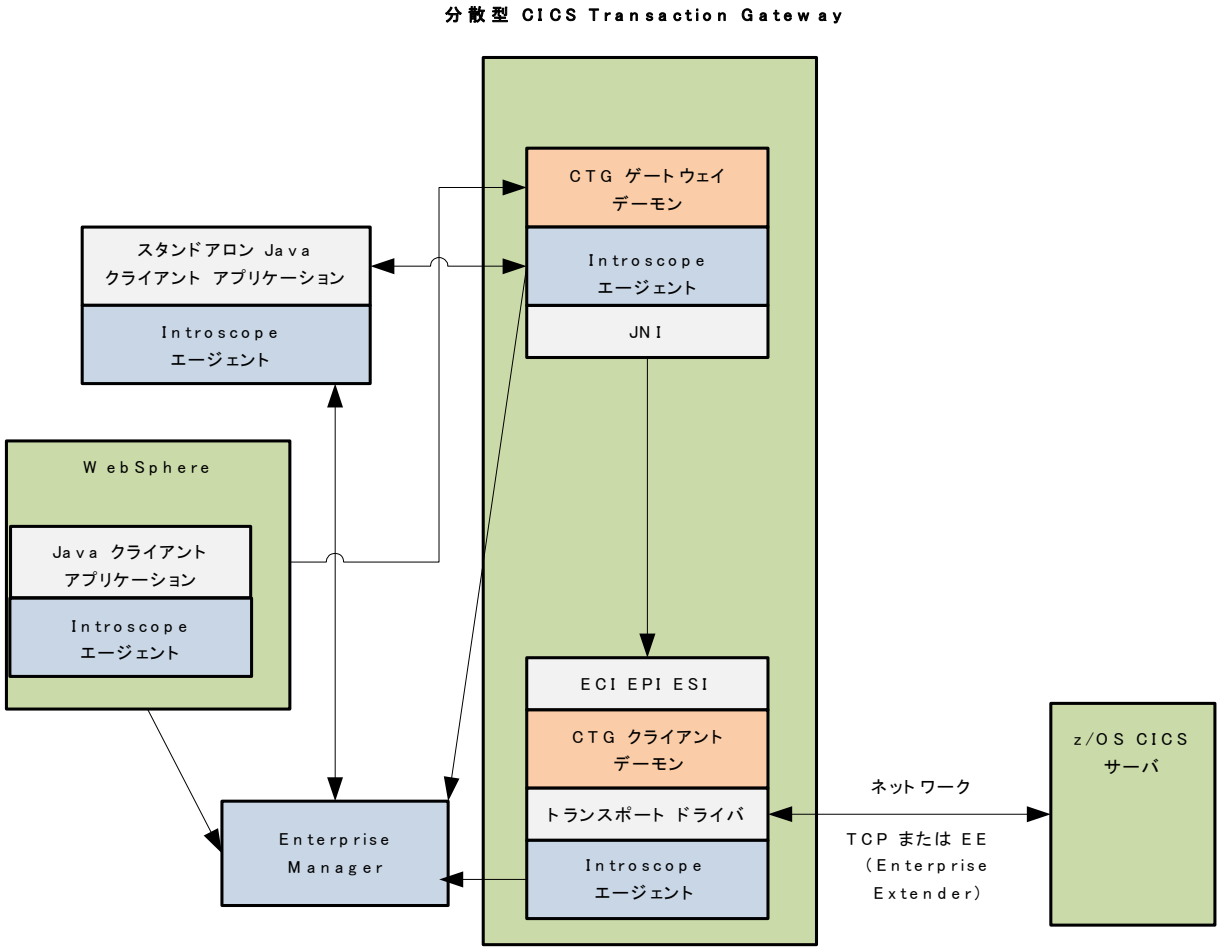
z/OS 用の CTG は、2 つの主要なランタイム コンポーネントによって構成されています。

- 受信作業をリスンし、IBM EXCI 機能を使用して、その作業をローカル CICS バックエンド サーバに転送する CTG Gateway デーモン。
- WebSphere ランタイム環境に展開された JCA（または CCF）リソース アダプタ。

CTG の拡張機能はこれらのコンポーネントを両方とも監視します。Introscope Investigator で、すべてのクライアント メトリックは［Frontends］-［Client_to_CTG_Aggregates］および［Frontends］-［Client_to_CTG_Details］ノード下に表示されます。 すべてのサーバ メトリックは、［Backends］-［CTG_to_CICS_xxx］および［Backends］-［JSSE_to_CTG］ノードに表示されます。

分散システム上の CTG コンポーネント

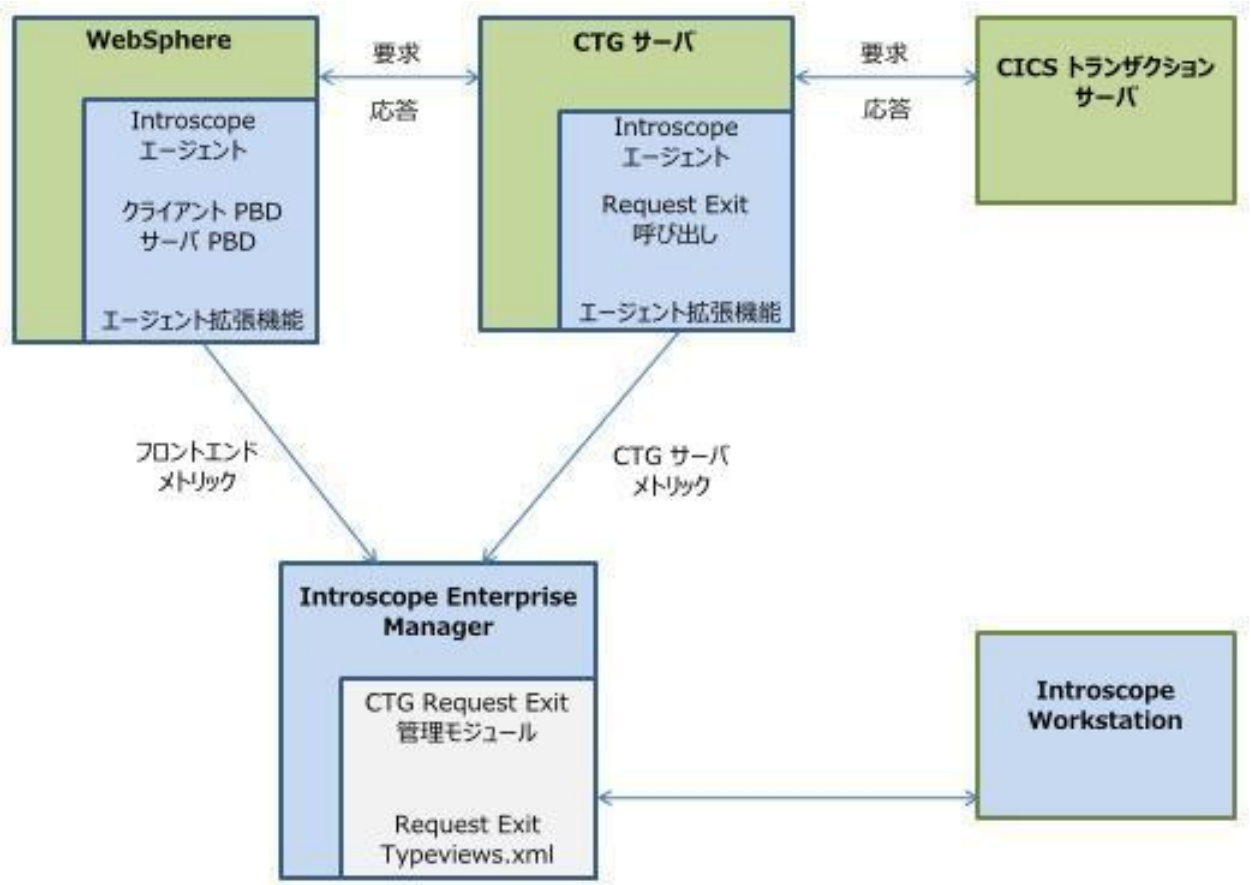
以下の図は、分散システム上の CTG コンポーネント間のインタラクションを示しています。



Request Exit の概要

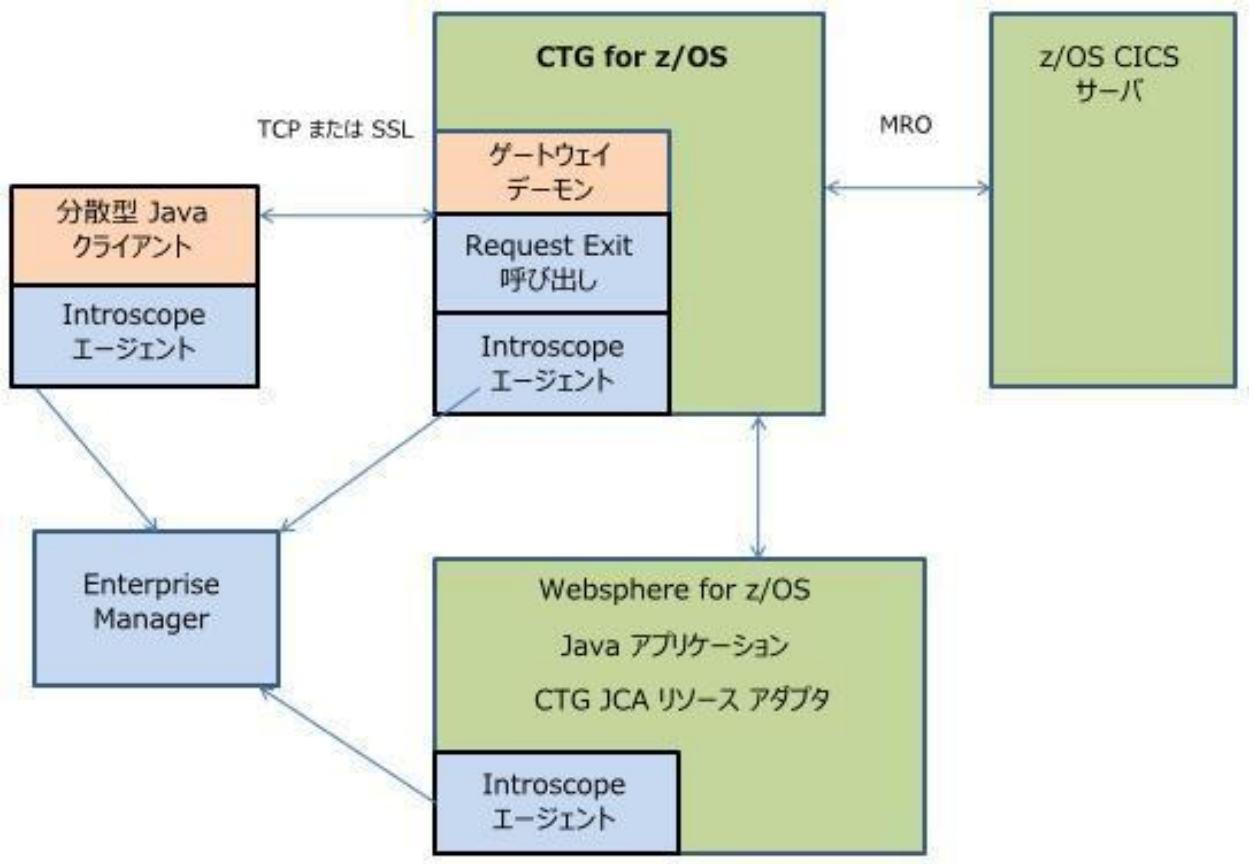
IBM は、サードパーティ プラグインが CTG によって処理された要求の統計情報を抽出することを可能にする Request Exit ポイントを提供しています。この Exit は、ECI トランザクションおよび IPIC トランザクションのみをサポートしています。 EPI はサポートされていません。

Request Exit 展開は、以下のコンポーネントおよびインタラクションを使用します。



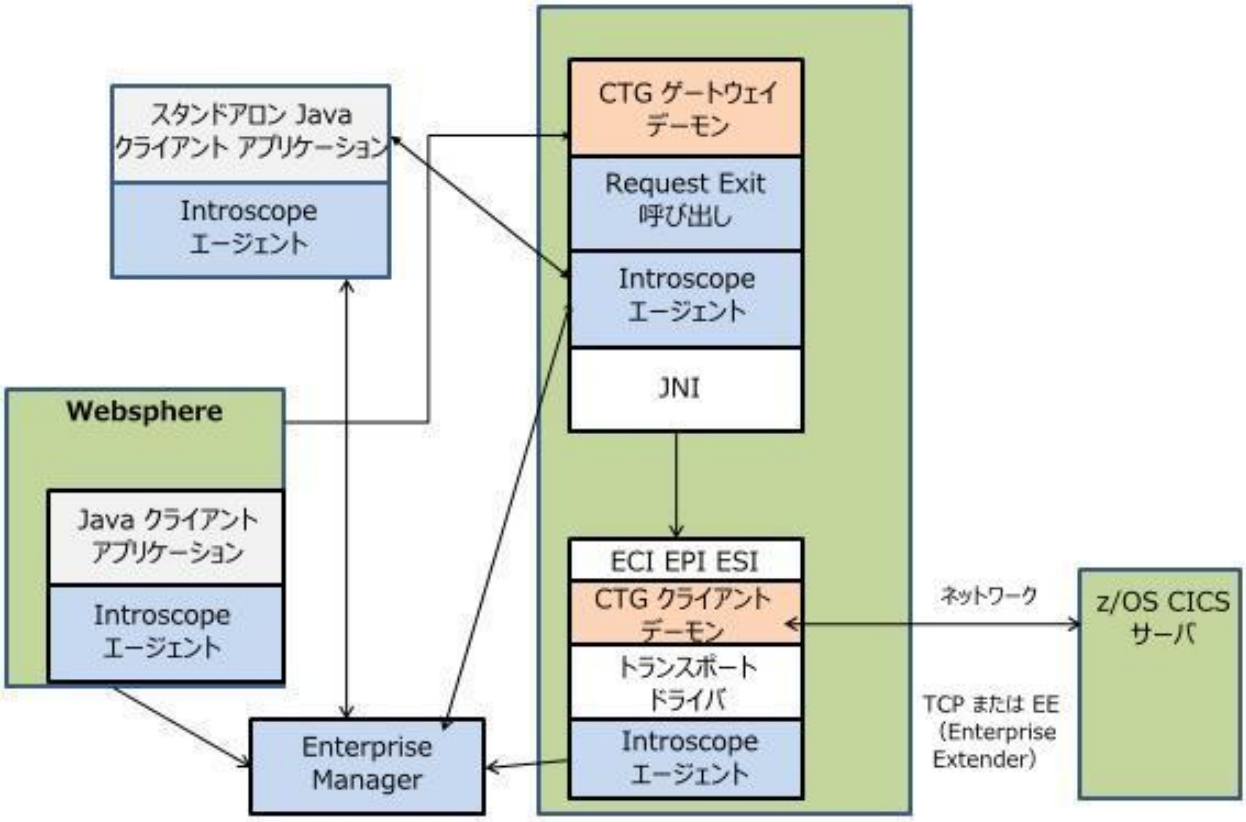
CTG コンポーネントでの Request Exit

以下の図は、z/OS 上の CTG Request Exit コンポーネント間のインタラクションを示しています。



分散システム上の Request Exit コンポーネント

以下の図は、分散システム上の CTG Request Exit コンポーネント間のインタラクションを示しています。



分散型 CTG と z/OS CTG の混在

CTG の拡張機能は、分散型環境と z/OS 環境の混在でも展開できます。 たとえば、分散型 WebSphere インスタンスは、z/OS で実行される CTG サーバと通信する CTG リソース アダプタで実行できます。

拡張機能によって両方の環境を監視できます。 したがって、単一のツール（Introscope）および単一の拡張機能スイート（CTG）で、必要に応じて複数の環境にわたって測定データを統合できます。

注: WebSphere は CICS と直接対話しません。 すべての通信は CTG を経由して行われます。

第 2 章: CTG の拡張機能のインストール

ここでは、CTG の拡張機能をインストールおよび設定する方法、および Introscope エージェント プロファイル ファイル内で CTG プロパティを定義する方法について説明します。

このセクションには、以下のトピックが含まれています。

インストールの準備

インストールを準備するには、必ず以下の手順を完了します。

次の手順に従ってください:

1. システムが要件 (P. 12)を満たしていることを確認します。

   注: Introscope をインストールしていない場合は、「*CA APM インストールおよびアップグレード ガイド*」の説明に従います。

2. Introscope 環境で以下のディレクトリの場所を確認します。

   - *Introscope Enterprise Manager ディレクトリ（<EM_Home>）* — Enterprise Manager のインストール ディレクトリ。
   - *Introscope エージェント ホーム ディレクトリ（<Agent_Home>）* - CTG サーバを監視する Introscope エージェントのインストール ディレクトリ。CTG サーバ データを収集するエージェントごとに、このディレクトリを識別します。
   - *Introscope エージェント プロファイル ディレクトリ* - CTG の拡張機能をインストールする各エージェント上で、*IntroscopeAgent.profile* ファイルがあるディレクトリ。

   注: *com.wily.introscope.agentProfile* のシステム プロパティ ディレクトリを識別してください。エージェント プロファイルは、通常、エージェント インストールの wily¥core¥config ディレクトリにあります。

3. CTG の拡張機能 をインストールする予定の Enterprise Manager を停止します。

   クラスタ化された環境の場合

   1. Manager of Managers として機能している Enterprise Manager を停止します。
   2. CTG 対応エージェントの拡張機能に接続される各コレクタ Enterprise Manager を停止します。

4. CTG の拡張機能をインストールする予定のすべてのコンポーネントを停止します。 これらには以下のものが含まれる可能性があります。

   - Java アプリケーション
   - WebSphere アプリケーション サーバ[WebSphere あぷりけーしょんさーば]
   - CTG サーバ ソフトウェア

注: CTG の拡張機能をインストールするために、一度に複数のコンポーネントを停止する必要はありません。 あるコンポーネントを停止し、インストールしてから別のコンポーネントに進みます。 たとえば、Enterprise Manager を停止して CTG の拡張機能をインストールし、次に各 Introscope エージェントを停止してインストールします。 クラスタ環境の場合は、Manager of Managers を停止してから CTG の拡張機能をインストールし、次に各コレクタ Enterprise Manager を停止してインストールし、その後に各エージェントを停止してインストールします。

インストール アーカイブの抽出

ご使用のシステムに適切な CTG アーカイブ用の拡張機能を抽出します。

CTG 管理モジュール用の拡張機能とサーバ拡張機能によって Enterprise Manager が拡張され、Introscope で CTG の使用状況とパフォーマンスが監視できるようになります。

重要: z/OS プラットフォームを使用している場合は、必ず FTP バイナリモードを使用して、アーカイブ全体を転送してください。ASCII に変換すると、ファイルが破損します。

PBD 用の管理モジュールおよびタイプ ビューのインストール

以下の手順は、PBD インスツルメンテーション用の管理モジュールおよびタイプ ビューアのインストールに適用されます。

次の手順に従ってください:

1. CTG の拡張機能のインストールを予定している Enterprise Manager を停止します。

   クラスタ化された環境の場合

   1. Manager of Managers として機能している Enterprise Manager を停止します。
   2. 接続されている各コレクタ Enterprise Manager を停止します。

2. *<EM_Home>*/config/modules に以下のファイルをコピーします。
   - PPCTGClient_ManagementModule.jar
   - PPCTGServer_ManagementModule.jar

   クラスタ化された Introscope 環境に CTG の拡張機能をインストールする場合は、Manager of Managers および接続されているすべての Enterprise Manager にインストールします。
3. ctg.typeviewers.xml を <EM_Home>/ext/xmltv にコピーします。

   クラスタ化された Introscope 環境に CTG の拡張機能をインストールする場合は、まず Manager of Managers として機能している Enterprise Manager に CTG サーバ拡張の拡張機能をインストールします。次に、Manager of Managers に接続されているすべてのコレクタ Enterprise Manager にインストールします。
4. 抽出されたインストール アーカイブ ファイル *ctg.typeviewers.xml* を *<Introscope_Home>*/ext/xmltv にコピーすることにより、拡張機能をインストールします。

Request Exit 用の管理モジュールおよびタイプ ビューのインストール

以下の手順は、Request Exit インスツルメンテーション用の管理モジュール タイプ ビューアのインストールに適用されます。

次の手順に従ってください:

1. CTG の拡張機能のインストールを予定している Enterprise Manager を停止します。

   クラスタ化された環境の場合

   1. Manager of Managers として機能する Enterprise Manager を停止します。
   2. 接続されている各コレクタ Enterprise Manager を停止します。

2. PPCTGRequestExit_ManagementModule.jar を <EM_Home>/config/modules にコピーします。

   クラスタ化された Introscope 環境に CTG の拡張機能をインストールする場合は、Manager of Managers およびそれに接続されているすべての Enterprise Manager にインストールします。

3. ctg.requestexit.typeviewers.xml を <EM_Home>/ext/xmltv にコピーします。

CTG サーバを監視する Introscope エージェント用のファイルのインストール

CTG の拡張機能により、CTG サーバの使用状況とパフォーマンスを監視できます。以下では、CTG サーバを監視するエージェントまたは Request Exit をインストールする方法について説明します。

CTG サーバを監視するエージェントをインストールするには、以下の手順に従います。

次の手順に従ってください:

1. 拡張機能をインストールする CTG サーバを停止します。
2. PPCTGAgent.jar を <Agent_Home>/examples/PowerPackForIBMCTG/ext から <Agent_Home>/wily/core/ext にコピーします。

CTG サーバを監視する *Request Exit* をインストールするには、以下の手順に従います。

次の手順に従ってください:

1. 拡張機能をインストールする CTG サーバを停止します。
2. PPCTGRequestExit.jar を <Agent_Home>/examples/PowerPackForIBMCTG/ext から CTG classes ディレクトリにコピーします。
3. PPCTGRequestExit.jar が CTG の CLASSPATH リストに存在することを確認します。
4. ［CTG Config］メニューで CTG Request Exit の名前を設定します。

AutoProbe を使用した CTG コードの有効化

PBD のデプロイでは、AutoProbe は、起動時に自動的にインスツルメンテーション プローブを作成します。

Request Exit を使用した CTG コードの有効化

Request Exit を使用して CTG サーバを動的にプローブできます。 CTG クライアント コードについては、JVM AutoProbe を代わりに使用する必要があります。

分散型 CTG プラットフォーム上で手動の ProbeBuilder を使用した CTG コードの有効化

手動の ProbeBuilder は、CTG Gateway Server クラスを Introscope 対応にする非動的な方法です。 ProbeBuilder を手動で実行すると、CTG サーバが実行される前に、ディスク上で CTG Java クラスが Introscope 対応になります。

推奨されている CTG インスツルメンテーションの方法は、JVM 自動プローブまたは Request Exit を使用する方法です。 ただし、CTG アプリケーションを手動でプローブする場合は、以下の手順に従います。

CTG サーバ コードを有効にするための準備

このセクションの説明は、以下のインストール タスクおよび設定タスクを実行していることを前提としています。

- Introscope エージェントをインストールしている。
- 以下の ProbeBuilder ディレクティブ インスツルメンテーションを完了している。
  - EM フォルダの custompbd に PPCTGClient-full.pbd および PPCTGServer-full.pbd をコピーしている。
  - examples から Agent extension フォルダに、PPCTGAgent.jar をコピーしている。

- PBD ファイルをインクルードする。詳細については、「WebSphere での CTG クライアント コードの有効化」を参照してください。
- エージェント名を設定している。
- すべての必要な ProbeBuilder オプションを設定している。
- ErrorDetector を使用している場合、errors.pbd を custompbd にコピーしている。

- Request Exit インスツルメンテーションを完了している。
  - Request Exit インスツルメンテーション用の PPCTGRequestExit.jar をインストールしている。
  - examples から Agent extension フォルダに、PPCTGAgent.jar をコピーしている。
  - PBD ファイルをインクルードしている。
  - エージェント名を設定している。
  - CTG classes フォルダに PPCTGRequestExit.jar をコピーしている。
  - PPCTGRequestExit.jar ファイルが CTG クラス パス リストにあることを確認した。
  - Windows 用に ctg.ini ファイルまたは CTG 設定メニューで CTG Request Exit の名前を設定した。

Windows 用の CTG サーバ Request Exit インスツルメンテーションの指定

CTG の Windows バージョンは、Windows サービスとしてのみ実行され、ctgservice コマンドの使用を必要とします。 このコマンドは、サービスの開始時に、CTG サーバに渡される Introscope プロパティを設定するために使用されます。 このコマンドは、一般に以下のように呼び出されます。

```
ctgservice -R
-A-j-Dcom.wily.introscope.agentProfile=C:¥<Agent_HOME>¥wily¥core¥config¥IntroscopeAgent.profile
-A-j-javaagent:C:¥<Agent_Home>¥wily¥Agent.jar
```

CTG サーバの手動プローブ作成オプションの入力

CTG サーバ ファイルのプローブ作成の前に、以下の手順を実行して、手動プローブ作成の準備を行います。

次の手順に従ってください:

1. 現在の CTG の classes/*.jar ファイルをバックアップ フォルダに保存します。
2. CTG の classes ディレクトリ内の以下の jar ファイルに対して ProbeBuilder を実行します。
   - *ctgclient.jar*
   - *ctgserver.jar*
   - *cicsj2ee.jar*

   このアクションでは、CTG の *¥classes* ディレクトリに **.isc.jar* ファイルのセット（たとえば *ctgclient.isc.jar*、*ctgserver.isc.jar*、*cicsj2ee.isc.jar*）が作成されます。
3. 以下の **.isc.jar* ファイルの名前を変更します。
   - Windows

     ```
     rename ctgserver.isc.jar ctgserver.jar
     rename ctgclient.isc.jar ctgclient.jar
     rename cicsj2ee.isc.jar cicsj2ee.jar
     ```
   - UNIX

     ```
     mv ctgserver.isc.jar ctgserver.jar
     mv ctgclient.isc.jar ctgclient.jar
     mv cicsj2ee.isc.jar cicsj2ee.jar
     ```
4. プローブされた jar ファイルの *.isc* バージョンが別のディレクトリに作成された場合は、CTG の *classes* ディレクトリにそれらを直接コピーします。
5. jar ファイルを手動でインスツルメントします。 これを実行するには、以下の 2 つの方法があります。
   - ProbeBuilder ウィザード — ProbeBuilder を実行する GUI ダイアログ ボックスを提供します。
   - コマンドライン ProbeBuilder — ウィンドウ システムのない環境向けの ProbeBuilder へのコマンドライン インターフェースです。

   注: バイトコードのインスツルメントの詳細については、「*CA APM Java Agent 実装ガイド*」を参照してください。

CTG サーバ用の ProbeBuilder ウィザードの使用

コンピュータに Windows 環境がある場合、GUI ベースの ProbeBuilder ウィザードを使用できます。

次の手順に従ってください:

1. ProbeBuilder ウィザードが、CTG 拡張 .pbd ファイルに対するアクセス権を持つことを確認します。それらは、*<Agent_Home>* ディレクトリにインストールする必要があります。
2. *ctgserver.jar*、*ctgclient.jar*、および *cicsj2ee.jar* に対して ProbeBuilder ウィザードを実行します。
3. 開始する前に、保管用のバックアップ領域に元の CTG jar ファイルをコピーします。

CTG サーバ用のコマンド ライン ProbeBuilder の使用

コンピュータにウィンドウ環境がない場合、コマンド ライン ProbeBuilder を使用して、CTG jar ファイルを手動でプローブできます。

次の手順に従ってください:

1. ProbeBuilder を実行する前に、保管用のバックアップ領域に元の CTG jar ファイルをコピーします。
2. ProbeBuilder のコマンドとディレクティブを使用します。
3. CTG の classes ディレクトリ内の分散型 CTG ファイルをプローブしている場合は、必ず手動で以下の jar ファイルをプローブします。
   - *ctgserver.jar*
   - *ctgclient.jar*
   - *cicsj2ee.jar*

注: ProbeBuilder のコマンドおよびディレクティブの詳細については、「*CA APM Java Agent 実装ガイド*」を参照してください。

WebSphere での CTG クライアント コードの有効化

z/OS 上の WebSphere で、JVM AutoProbe を使用して、CTG クライアントおよびサーバを自動的に監視できます。

次の手順に従ってください:

- *IntroscopeAgent.profile* ファイルを編集し、以下のファイルを定義に含めるように *introscope.autoprobe.directivesFile* プロパティを変更します。
  - *PPCTGClient-full.pbd*
  - *PPCTGServer-full.pbd*
  - *required.pbd*

  例：

  ```
  introscope.autoprobe.directivesFile=<list of existing PBDs>,
  PPCTGClient-full.pbd,PPCTGServer-full.pbd,required.pbd
  ```

  注: *introscope.autoprobe.directivesFile* の定義全体が 1 行に含まれている必要があります。

z/OS CTG 起動スクリプトの変更

z/OS 環境にインストールしている場合は、起動時に CTG JVM に Introscope 対応 JVM AutoProbe パラメータを渡すように CTG の *ctgstart* スクリプトを変更します。

次の手順に従ってください:

1. 元の *ctgstart* スクリプトのバックアップ コピーを作成します。
2. 編集のために CTG の *ctgstart* スクリプトを開きます。*ctgstart* スクリプトは、CTG の bin ディレクトリにあります。
3. *ctgstart* スクリプトの下部の近くにある、以下のステートメント ブロックを検索します。

```
 jvmoptions=$(echo $alloptions | $awk '{
    for (i = 1; i<=NF; i++) {
        if (substr($i,1,2) == "-j") jvmoptions = jvmoptions " "
substr($i,3)
    }
    print jvmoptions
  }')
```

4. AutoProbeConnector を使用して AutoProbe を有効にするには、手順 2 でステートメントの直後に以下の行を追加します。

```
wilyoptions="-Xbootclasspath/p:./wily/connectors/AutoProbeConnector.jar:./wily/Agent.jar -Dcom.wily.introscope.agentProfile=./wily/IntroscopeAgent.profile -Xverify:none"
jvmoptions="$jvmoptions $wilyoptions""
```

5. javaagent オプションを使用して AutoProbe を有効にするには、手順 2 でステートメントの直後に以下の行を追加します。

```
wilyoptions="-javaagent:./wily/Agent.jar
-Dcom.wily.introscope.agentProfile=./wily/IntroscopeAgent.profile"
jvmoptions="$jvmoptions $wilyoptions"
```

分散型システム用にインスツルメントされた CTG 起動スクリプトの作成

Windows と UNIX では、分散型 CTG Startup はバイナリ ファイルです。 このファイルを起動スクリプトで呼び出して、必要な Introscope パラメータを渡す必要があります。

次の手順に従ってください:

1. 手動の ProbeBuilder を使用して CTG クラスにプローブを追加した後、分散型 CTG サーバ用の起動スクリプトを作成します。 Windows では、スクリプトは *.bat* ファイルです。UNIX では、*.sh* シェル ファイルです。
2. 起動スクリプトで、Introscope クラスおよびエージェント プロファイルの場所を指定します。

Windows 用のサンプル起動スクリプト（.bat ファイル）を以下に示します （CTG と WebSphere がそれらの IBM 製品のインストールされたフォルダの場所であると仮定しています）。

```
set
CLASSPATH=c:¥CTG<ccc>¥wily¥Agent.jar;c:¥CTG<ccc>¥classes¥cicsj2ee.jar;c:¥CTG<
ccc>¥classes¥ctgserver.jar;c:¥CTG<ccc>¥classes¥ctgclient.jar;
c:¥CTG<ccc>¥classes¥ccf2.jar;c:¥CTG<ccc>¥classes¥ctgsamples.jar;
C:¥WebSphere<www>¥AppServer¥java¥jre¥bin;%CLASSPATH%
set JAVA_HOME=C:¥IBM_JVM<jjj>¥java¥jre
set PATH=C:¥WebSphere<www>¥AppServer¥java¥jre¥bin;.¥bin;¥%PATH%
ctgstart
-j-Dcom.wily.introscope.agentProfile=C:¥CTG<ccc>¥wily¥core¥config¥IntroscopeA
gent.profile
```

各項目の説明

```
<ccc> = CTG バージョン番号
<jjj> = Java バージョン番号
<www> = WebSphere バージョン番号
```

3. アプリケーション起動スクリプトのクラスパスを編集して、ProbeBuilder で作成され、インスツルメントされたコードを含むディレクトリの場所を含めます。

   注: これらのエントリが、クラスパスの元のエントリよりも前にあることを確認します。 「インスツルメントされたコードの実行 (P. 33)」を参照してください。

4. アプリケーション起動スクリプトのクラスパスを編集して、以下のパスを含めます。

   ```
   <Agent_Home>/Agent.jar
   ```

   たとえば、以下のクラスパスを編集できます。

   ```
   java -classpath
   /<your-applicationpath>/classes:/<yourapplicationpath>/lib/app.jar MainClass
   ```

   以下のようになります。

   ```
   java -classpath
   /<your-applicationpath>.isc/classes:/<yourapplicationpath>.isc/lib/app.jar:<A
   gent_Home>/Agent.jar MainClass
   ```

5. 手動インスツルメンテーションの完了後、インスツルメントされたコードおよびエージェントの場所を反映するために、CTG サーバ起動スクリプトのクラスパスを更新します。

6. 新しい起動スクリプトでアプリケーションを開始します。

インスツルメントされたコードの実行

元のコードの代わりに Introscope 対応コードをポイントする方法は 3 つあります。

- クラスパスで元のクラスのパスを インスツルメントされたコードのパスと置き換えます。 この章の操作手順では、アプリケーションを初めてインスツルメントするときに、このプロセスを実行するように指示しています。

  注: Windows または UNIX 上のインストールで実行されていない場合、*Java¥bin* フォルダを使用するようにクラスパスを更新する必要がある場合があります。 このフォルダは、Java コミュニティから提供された Java ではなく、WebSphere アプリケーション サーバの *Java¥bin* ディレクトリから提供されます。 このクラスパスの更新は、JAVA-CTG の互換性のために必要です。

- パスをクラスパスの先頭に追加します。 クラスパスで、アプリケーション コードの一部のみがインスツルメントされている場合には、元のパスの前に、インスツルメントされたコードのパスを配置します。

  注: これを行うと、インスツルメントされたコードがロードされ、パフォーマンス データがレポートされます。インスツルメントされていないコードもロードされ、正常に動作しますが、パフォーマンス データはレポートされません。

- インスツルメントされたコードを元のクラスパスに配置します。

  クラスパスが多くの場所で設定されている場合、またはシステムを評価する場合は、この方法を使用します。 この方法を実運用環境で使用する場合は慎重に行ってください。 元のコードとインスツルメントされたコードのどちらを使用しているのかわからなくなることがあるためです。

  - 元のコードを新しい場所に移動します。 クラスパスを変更せず、次に元の場所にインスツルメントされたコードを移動します。
  - また、UNIX マシンでは、元のパスの場所からインスツルメントされたコードを指すシンボリック リンクを作成できます。

元のコードに戻す

元のインスツルメントされていないコードに戻すには、以下のようにインスツルメンテーションを元に戻します。

- インスツルメントされたコードへのパスを Java クラスパスに配置した場合は、インスツルメントされたコードへのパスを元の値と置き換えます。
- 元のコードへのパスの前にインスツルメントされたコードへのパスを追加した場合は、元のクラスパスのみが残るように、クラスパスの先頭に付けられた部分を削除します。
- 元のコードを削除した場合で、元のクラスパスにインスツルメントされたコードを配置した場合は、インスツルメントされたコードを元のパスから削除します。 元のクラスパスに元のコードを配置します。

注: UNIX システムでシンボリック リンクを使用した場合は、シンボリック リンクで元のディレクトリをポイントするか、リンクを削除して元のクラスパスにコードを移動します。

分散型 CTG プラットフォーム上での ChangeDetector の有効化

ChangeDetector は、Windows、Linux、および UNIX などの分散化（非 z/OS）システム上でサポートされています。 ChangeDetector は、CTG 設定ファイル（*ctg.ini* や *ctgenvvar* など）、および関連する JAR ファイルの両方を監視して、システムに対して何らかの変更が行なわれたかどうかを検出するように設計されています。 この設計により、CTG 環境で停止や問題を引き起こす可能性がある設定変更を追跡できます。

CTG の拡張機能には、ChangeDetector で CTG インストールを正しく監視できるようにするために、カスタマイズする必要があるデフォルトの ChangeDetector 設定ファイルが含まれています。 さらに、ChangeDetector によって監視するためにほかのファイルを追加できます。

注: 監視のためのファイルの追加方法の詳細については、「*CA APM ChangeDetector ユーザ ガイド*」を参照してください。

このセクションでは、CTG に関連する ChangeDetector の監視について明示的に焦点を当てます。 *ChangeDetector-config.xml* ファイルには、CTG および関連する Introscope エージェント ファイル エンティティを監視するための ChangeDetector 設定のデフォルト セットが含まれます。 この設定を正しく使用するには、*ChangeDetector-config.xml* ファイル内の 2 つのエントリを更新します。

注: 分散型 CTG システム上で ChangeDetector を使用している場合、ChangeDetector は、Java クラス監視を使用して Java クラスの変化を検出できないことに注意してください。 この制限は、CTG エージェントが手動でプローブされており、Java クラスの動的な変化を検出できないことによります。
（ChangeDetector は、ファイル システムおよび設定プロパティ ファイルの変化を想定通り監視および検出できます。）

次の手順に従ってください:

1. CTG ディレクトリ コマンド ブロックで ChangeDetector-config.xml ファイルを更新する方法

   ```
   <!-- ========================================== -->
   <!-- change the name= property below to point to your CTG directory -->
   <!-- ========================================== -->
   <scan-directory recursive="true" name="your CTG directory" fileset="default"
   enabled="true" />
   ```

   特定の CTG ディレクトリに指すように *name=* パラメータを変更します。

   たとえば、CTG インストールが ctg<ccc> ディレクトリ下にあった場合 /usr/lpp/ctg/ctg<ccc> となり、scan-directory エントリの name パラメータを以下のように設定します。

   ```
   <scan-directory recursive="true" name="/usr/lpp/ctg/ctg<ccc>"
   fileset="default" enabled="true" />
   ```

2. CA APM ディレクトリ コマンド ブロックで、以下のように変更します。

```
<!-- ========================================== -->
<!-- change the name= property below to point to your CA APM directory -->
<!-- ========================================== -->
<scan-directory recursive="true" name="your CA APM directory" fileset="default"
enabled="true" />
```

   特定の CA APM インストール ディレクトリを指すように、name= パラメータを変更します。

   たとえば、CA APM インストールが以下のディレクトリ下にあった場合 */usr/vendor/ca apm*

   scan-directory エントリの name パラメータを、以下のように変更します。

```
<scan-directory recursive="true" name="/usr/vendor/ca apm" fileset="default"
enabled="true" />
```

3. これらの変更が終わったら、Introscope エージェント ディレクトリに更新された *ChangeDetector-config.xml* ファイルを入れます。
4. エージェントの *.ext* ディレクトリに *ChangeDetectorAgent.jar* ファイルがインストールされることを確認します。
5. ChangeDetector 設定ファイルへのパスを指定するために、エージェント プロファイルの *introscope.changeDetector.profile=config* エントリを変更します。
6. エージェント プロファイルの *introscope.changeDetector.agentID= config* エントリには、ChangeDetector エージェントに対して使用する名前が含まれることを確認します。

   たとえば、CTG インストールが ctg<ccc> ディレクトリ下にあった場合 /usr/lpp/ctg/ctg<ccc> となり、scan-directory エントリの name パラメータを以下のように設定します。

```
<scan-directory recursive="true" name="/usr/lpp/ctg/ctg<ccc>"
fileset="default" enabled="true"/>
```

上記の手順を実行した後、ChangeDetector を使用して、CTG サーバを監視できます。

ChangeDetector データの表示と解釈の詳細については、「*CA APM ChangeDetector ユーザ ガイド*」を参照してください。

CTG サーバ RequestExit およびグローバル統計情報プローブの設定（オプション）

CTG サーバを設定して IBM RequestExit メトリックを取得するには、以下の手順に従います。

次の手順に従ってください:

1. CTG の classes ディレクトリに PPCTGRequestExit.jar を配置します。

   例：

   ```
   /u/usr/lpp/cicstg/ctg800/classes/PPCTGRequestExit.jar
   ```

2. CTG の CLASSPATH リストに PPCTGRequestExit.jar を追加します。

   例：

   ```
   export CLASSPATH=${CTG_CLASSES}/PPCTGRequestExit.jar:${CLASSPATH}
   ```

3. IntroscopeAgent.profile 内の以下の値をカスタマイズします

   ```
   ppctg.statistics.host=localhost
   ppctg.statistics.port=2980
   ppctg.statistics.sleep=30
   ppctg.statistics.enable=true
   ```

   注: CTG_Global_Statistics_StatsExit の概要タイプ ビューは意図的にブランクになっており、このタブにはメトリック データが含まれていません。CTG_Global_Statistics_StatsExit のタイプ ビューにデータが表示されない場合、それは ［CTG_Global_Statistics_StatsExit］ ノードにメトリック データが含まれていないことを示します。

設定ファイル ctg.ini 内での z/OS 上の CTG サーバ RequestExit およびグローバル統計情報プローブの設定（オプション）

設定ファイル ctg.ini 内で z/OS プラットフォームを設定して、z/OS 上での RequestExit の詳細およびグローバル ステータス プローブを取得するには、以下の手順に従います。

次の手順に従ってください:

1. ctg.ini に APM/Wily PPCTG Request Exit の名前を追加します。
2. CTG インストール環境に応じて、ほかのパラメータを追加します。

   例：

   ```
   requestexits = com.ibm.ctg.server.APM_RequestExit_Monitor
   protocol@statsapi.handler=com.ibm.ctg.server.RestrictedTCPHandler
   protocol@statsapi.parameters=port=2980;bind=;connecttimeout=2000;maxconn=5;
   ```

Windows プラットフォーム上で RequestExit 詳細とグローバル統計情報を別々に取得するための設定（オプション）

Windows プラットフォーム上で RequestExit 詳細とグローバル統計情報を別々に取得するには、以下の手順に従います。

次の手順に従ってください:

1. ctgservice コマンドを実行して、Request Exit の クラスパスを設定します。 PPCTGRequestExit.jar を Agent¥wily¥examples¥ext フォルダから IBM CTG¥Classes フォルダにコピーした後。

   例：

   ```
   ctgservice -R -A-classpath=C:¥<IBM CTG Home>¥classes¥PPCTGRequestExit.jar
   -A-j-Dcom.wily.introscope.agentProfile=C:¥<Agent-Home>¥wily¥core¥config¥Intro
   scopeAgent.profile
   -A-j-javaagent:C:¥<Agent-Home>¥wily¥Agent.jar
   ```

2. IBM CTG Configuration Tool を呼び出しゲートウェイ デーモン ノードを選択することにより、CTG プログラムへ RequestExit_Monitor 名を設定します。
3. ［Monitoring］タブを選択し、［Use These Request Monitors］フィールドに移動して、ボックスに以下の APM 終了値を入力します。

   ```
   com.ibm.ctg.server.APM_RequestExit_Monitor
   ```

4. ［Add］を選択して、上記のエントリを追加します。

RequestExit 詳細ステータスのパラメータを設定するには、以下の手順に従います。

次の手順に従ってください:

1. ［CICS Transaction Gateway］-［Gateway Daemon］-［Statistics API Options］ノードを展開し、選択します。
2. ［Enable Protocol Handler］チェック ボックスをオンにします。
3. （オプション）デフォルトをオーバーライドするには、TCP ポート番号、タイムアウト、およびその他のオプションを変更します。
4. フィールド［BinAddress］を空白のままにしている場合は、CTG は localhost を使用します。

   前の手順により、IBM Request Exit 詳細メトリックの収集が有効になります。

RequestExit グローバル統計情報のパラメータを設定するには、以下の手順に従います。

次の手順に従ってください:

注: IBM サポートでの制限により、Request Exit グローバル統計情報は、自動的には有効にできません。

1. PP によるグローバル統計情報の収集を呼び出すには、PPrunGlobalStats.bat ファイルを実行して、以下のエントリをカスタマイズします。
2. CTG クラスおよび Introscope エージェント ディレクトリを指すには、以下のように設定します。

```
set CTG_CLASSES= <point to the CTG class files>
set ISCOPE_AGENT= <point to Agent extension folder>
```

CTG クライアント サポートの設定

CTG の拡張機能は、CTG サーバおよび CTG クライアント アプリケーションの両方のメトリックを提供します。このセクションでは、CTG クライアント アプリケーションを、CTG サーバに対して要求を発行するアプリケーションとして定義する方法を説明します。さらに、CTG サーバは、特定の CICS ECI または EPI アプリケーションの実行を要求します。次に、その結果がクライアントに返されます。

CTG の拡張機能によって提供されるクライアント側メトリックにより、測定範囲が開始時のクライアントに拡張されます。 この範囲により、応答時間、切り分けなど、包括的な管理が有効になります。

CTG の拡張機能は、スタンドアロン CTG クライアント アプリケーション、または WebSphere 下で実行されるクライアント アプリケーションで動作するように設定できます。

スタンドアロン CTG クライアント アプリケーションの設定

スタンドアロン CTG クライアント アプリケーションでは、Introscope サポートは、Introscope を使用して測定されるその他の任意のスタンドアロン Java アプリケーションと同じ方法で設定されます。 さらに、*IntroscopeAgent.profile* の *introscope.autoprobe.directivesFile= <parameter>* に *PPCTGClient-full.pbd* ファイルと *required.pbd* ファイルを追加する必要があります。ローカル モード（EXCI）のインタラクションを監視する場合、リストに *PPCTGServer-full.pbd* も追加する必要があります。

注: CTG 拡張機能の *PPCTGAgent.jar* ファイルが、エージェントの *ext* ディレクトリに配置されていることを確認します。

チャネルおよびコンテナ メトリックの設定

チャネルおよびコンテナに関するメトリックの収集および CPU オーバーヘッドを最小限に抑えるために、チャネルおよびコンテナのメトリックを切り替えるオプションが PBD ファイルに用意されています。 切り替えるには、PPCTGServer-full.pbd および PPCTGServer-typical.pbd ファイルのディレクティブをコメント化またはコメント化解除します。

```
#  To disable IPIC Channel|Container metrics use this directive
#          Turnoff: IPICChannelContainerTracing
#  To enable use this directive
#          Turnon:  IPICChannelContainerTracing
```

変更を有効にするには、エージェントを再起動する必要があります。

WebSphere でのクライアント アプリケーションの設定

WebSphere で実行されるクライアント アプリケーションについては、CTG クライアント サポートを設定する 2 つの方法があります。

次の手順に従ってください:

- CTG の *ctgclient.jar*、*cicsj2ee.jar*、*ccf2.jar* などのファイルを、WebSphere JVM の［Classpath］設定メニューで直接追加する。

  注: CTG 8.0 以降、*cicsj2ee.jar* は *cicsjee.jar* に名前が変更されています。
- WebSphere で［Resource Adapter］画面を介して CTG jar ファイルをインストールし、関連する *cicseci.rar* ファイルと *cicsepi.rar* ファイル、またはそのいずれかをインストールする。

CTG クライアント サポートを WebSphere に追加した後、*PPCTGClient-full.pbd* ファイルと *required.pbd* ファイルを *IntroscopeAgent.profile* の *introscope.autoprobe.directivesFile=<parameter>* に追加する必要があります。 ローカル モード（EXCI）のインタラクションを監視する場合、リストに *PPCTGServer-full.pbd* を追加します。

注: CTG 拡張機能の *PPCTGAgent.jar* ファイルが、エージェントの *ext* ディレクトリに配置されていることを確認します。 含まれているファイルのリストについては、「CTG の拡張機能のコンポーネント (P. 12)」を参照してください。

CTG の拡張機能のアップグレード

以前のバージョンの CTG 拡張機能からは、アップグレードできません。代わりに、以前のバージョンをアンインストールし、次に現在のバージョンをインストールする必要があります。

インストールと設定の確認

CTG インストールの拡張機能が正しくインストールおよび設定されていることを確認することが重要です。

次の手順に従ってください:

1. 監視対象の WebSphere アプリケーション サーバ、Java アプリケーション、CTG サーバ ソフトウェア、および Introscope Enterprise Manager を再起動します。
2. これらのコンポーネントが正常に再起動した後、CTG サーバおよび WebSphere アプリケーション サーバを監視するエージェントが Enterprise Manager にデータをレポートし始めます。
3. Workstation を起動し、Enterprise Manager に接続します。
4. Investigator 内の CTG サーバおよびアプリケーション サーバからデータを表示 (P. 43) できることを確認します。

第 3 章: CTG の拡張機能の使用

このセクションでは、以下の内容について説明します。

- Introscope を使用して CTG 環境を監視する方法
- CTG の拡張機能で使用できる別の種類のデータの概要
- Workstation とコンソールにデータを表示する方法

このセクション内の手順では、インストール手順が完了しており、Introscope 対応 CTG アプリケーションが起動して稼働しており、Enterprise Manager にレポートしていると仮定しています。

このセクションには、以下のトピックが含まれています。

Introscope での CTG 拡張機能のデータの表示

Introscope で CTG 拡張機能のデータを表示するには、以下の方法があります。

- Investigator 内の未加工メトリック — 技術的なユーザに、CTG のすべてのリソースおよびコンポーネントについて、基礎となるパフォーマンスの詳細を表示します。
- Investigator 内のタブ ビュー — 技術的なユーザに、CTG システムのパフォーマンス、およびリソースとコンポーネントに関する集約を表示します。
- コンソール内のダッシュボード — CTG アーキテクチャの詳細を熟知していないユーザに、使いやすいインターフェースを提供します。
- Investigator 内のアラート — CTG ダッシュボードの拡張機能によって生成されたアラート、およびユーザが作成するアラートを表示します。

ツリー ビューでのメトリック データの表示

Investigator で CTG 拡張機能のメトリックを表示するには、以下の手順に従います。

次の手順に従ってください:

1. 管理対象のアプリケーションを起動します。
2. Enterprise Manager を起動します。
3. Workstation を起動してログインします。
4. Investigator ウィンドウを開きます。

CTG 拡張機能に固有のすべてのメトリックが、ツリー内のいくつかのノード下に表示されます。

注: 利用可能なメトリックは、ユーザ アプリケーションが使用する CTG と WebSphere のリソースによって異なります。 管理対象の Java アプリケーションによって使用されるメトリックのみ表示されます。

Frontends

以下のフロントエンド CTG メトリックが使用可能です。

- Apps — 個別のアプリケーション用のメトリックが、このノードの下にアプリケーション名ごとに表示されます
- Client_to_CTG_Aggregates — このノードには、［クライアント集約のグラフィカル サマリ］タブがあります。ツリーのノード下にいくつかの集約メトリック、および以下のサブノードがあります。
  - BASE_ECI_EPI — Request Exit インスツルメンテーションを持たず、また CTG 8.x ではない古い CTG バージョンを使用する場合、表示されることがあります。
  - JCA_ECI
  - JCA_EPI — Request Exit インスツルメンテーションを持たず、また CTG 8.x ではない古い CTG バージョンを使用する場合、表示されることがあります。
    - Screen
    - Terminal

- Client_to_CTG_Details — このノードには、［クライアント集約のグラフィカル サマリ］タブがあります。ツリーのこのノード下には、個別のクライアントを表すサブノードがあります。
- Client_to_CTG_JSSE_Session — このノードには［CTG ゲートウェイへの JSSE］タブがあります。 ツリーのこのノード下に、いくつかの集約メトリックがあります。

Backends

以下のバックエンド CTG メトリックが使用可能です。

- CTG_Global_Statistics — このノードは、以下の統計を表示します。
  - CICS Aggregates
  - Connection Manager
  - Connection Manager Threads
  - Gateway Daemon
  - HTTPRequest
  - Incoming Connection Requests From Clients
  - Individual Servers
  - Session
  - Worker Threads
- CTG_to_CICS_ECI_IPIC — このノードには、［ECI 全ゲートウェイのグラフィカル サマリ］タブがあります。ツリーのこのノード下には、各クライアントのメトリックを持つサブノードがあります。
- CTG_to_CICS_EPI — このノードには、［EPI すべてのゲートウェイ］タブがあります。 ツリーのこのノード下には、各クライアントのメトリックを持つサブノードがあります。

- JSSE_to_CTG — このノードには、［JSSE グラフィカル サマリ］タブがあります。 ツリーのこのノード下には、［受信 JSSE セッション サマリ］タブを表示するサブノードがあります。ここには、2 つのメトリックが表示されます。
  - 集約受信 SSL ハンドシェイク
  - 間隔ごとの受信 SSL ハンドシェイク数

注: IPIC トランザクションの実行時に、現在は［HTTPRequest］ノードが表示されます。

特定のメトリックに関する履歴データの分析

特定のメトリックに関する履歴パフォーマンス データを分析するには、*永続コレクション*をセットアップします。

注: 永続コレクションのセットアップの詳細については、「*CA APM インストールおよびアップグレード ガイド*」を参照してください。

サーバ メトリック

このセクションでは、CTG の拡張機能で使用できるバックエンドおよびフロントエンドのメトリックについて説明します。

Backends | CTG_Global_Statistics

［CTG_Global_Statistics］ノードを選択すると、Introscope Investigator は、以下のメトリックが表示される［グローバル統計情報グラフィカル サマリ］タブを表示します。

- 間隔ごとのクライアント要求呼び出しの集約
- クライアント要求応答時間（ミリ秒）を処理
- クライアント要求応答時間（ミリ秒）を送信
- 使用中接続マネージャ スレッド
- 使用中のワーカ スレッド

一般に、接続スレッド プールは、クライアント アプリケーションと CTG サーバ間の要求を処理するために使用されます。 また、ワーカ スレッド プールは、メインフレーム上の CTG サーバと CICS の間の要求を処理するために使用されます。

［CTG_Global_Statistics］ノードには、［グローバル統計情報の表サマリ］タブも含まれています。このタブには、CTG サーバが起動されているため、CTG サーバを介して送受信された作業量を示す実行集約統計情報のセットが表示されます。 このページには、以下の統計情報が示されます。

- 合計受信パケット数
- 合計送信パケット数
- 使用中の接続マネージャ スレッド
- 使用中のワーカ スレッド

Introscope Investigator ツリーの［CTG_Global_Statistics］ノードの下には、CTG サーバを介して送受信される作業に関するグローバル統計情報の全スイートが提供されます。 Investigator ツリーは階層的に構成されており、階層内の各サブノードに CTG サーバ内の特定の機能セットに関する主要な統計情報が表示されます。

［CICS Aggregates］サブノードは、CTG サーバとアップストリーム CICS システム間で送受信されるトラフィックに関する主要な統計情報を表示します。 このサブノードには、以下の統計情報が含まれます。

- 間隔および応答時間ごとの読み取りパケット数（受信した SNA または TCP パケット数）
- 間隔および応答時間ごとの書き込みパケット数（送信した SNA または TCP パケット数）

注: JCA では、［CommArea Aggregate Request Data］メトリックに変化するデータ値が表示されます。 Base では、メトリックが表示されません。

［CICS WLM］サブノードは、内部 WLM が処理対象の要求をどのようにスケジュールするか、またユーザが記述した終了処理がどのように行われるかを示します。

［ConnectionManager］サブノードは、CTG サーバと要求側の CTG クライアント間のトラフィックの処理に関する統計を示します。

［ConnectionManager Threads］サブノードは、クライアントへの ConnectionManager の要求/応答を処理するために使用されるスレッド プールの使用に関する統計を表示します。 これらの統計は、クライアントでスレッドのボトルネックが発生しているかどうかを示します。

［Gateway Daemon］サブノードは、送受信される要求全体の合計数、および CICS トランザクションのコミット、または実行中のロールバック（SYNCONRETURN など）に関連する合計数を示します。

［Incoming Connection Requests from Clients］サブノードは、クライアントから CTG サーバへの、間隔ごとの要求の到着レートを示します。

［Worker Threads］サブノードは、CICS に作業の要求をディスパッチするために使用されるスレッド プールの使用に関する統計を示します。 この統計は、上流の CICS に向かうスレッドのボトルネックが発生しているかどうかを示す場合があります。

Backends | CTG_to_CICS_ECI_IPIC ノード

［CTG_to_CICS_ECI_IPIC］ノードを選択すると、Introscope Investigator は、以下のメトリックが表示される［ECI 全ゲートウェイのグラフィカル サマリ］タブを表示します。

- 呼び出されたプログラム数の集約
- 間隔ごとのプログラム呼び出し数
- プログラム平均応答時間（ミリ秒）
- プログラム エラー数の集約

CTG to CICS ECI Host ID サブノード

各 CICS ECI ホストには、親の［CTG_to_CICS_ECI_IPIC］ノード下に独自のサブノードがあります。［CTG to CICS ECI Host ID］サブノードを選択すると、Introscope Investigator ビューア ペインの［ECI ゲートウェイ表サマリ］タブに以下のメトリックが表示されます。

- プログラム数の集約
- プログラムの平均応答時間（ミリ秒）
- 応答/間隔

- エラー/間隔
- ストール数

各ホストに関して、ツリーには、ホスト上で実行される各 CICS プログラムに対応するサブノードがさらに表示されます。

Backends | CTG_to_CICS_EPI ノード

［CTG_to_CICS_EPI］ノードを選択すると、Introscope Investigator は、以下のメトリックが表示される［EPI すべてのゲートウェイ］タブを表示します。

- 呼び出されたトランザクション数の集約
- トランザクション平均応答時間（ミリ秒）
- 間隔ごとのトランザクション呼び出し数
- トランザクション エラー数の集約
- 呼び出されたサービス要求数の集約
- サービス要求エラーの集約

CTG to CICS EPI Host ID サブノード

各 CICS EPI ホストには、親の［CTG_to_CICS_EPI］ノード下に独自のサブノードがあります。 ［CTG to CICS EPI Host ID］サブノードを選択すると、Introscope Investigator ビューア ペインの［EPI 集約サマリ］タブに以下のメトリックが表示されます。

- 呼び出されたトランザクション数の集約
- 秒あたりのトランザクション呼び出し数
- トランザクション平均応答時間（ミリ秒）
- トランザクション エラー数の集約

各ホストに関して、ツリーには、ホスト上で実行される各 CICS プログラムに対応するサブノードがさらに表示されます。

Backends | JSSE to CTG ノード

［JSSE_to_CTG］ノードを選択すると、Introscope Investigator は、以下の 2 つのメトリックが表示される［JSSE グラフィカル サマリ］タブを表示します。

- 集約受信 SSL ハンドシェイク
- 間隔ごとの受信 SSL ハンドシェイク呼び出し数

JSSE to CTG ノード下に、JSSE ServerSocket 用のサブノードがあります。

JSSE ServerSocket サブノード

［JSSE ServerSocket］サブノードは、以下の 2 つのメトリックが表示される［受信 JSSE セッション サマリ］タブを表示します。

- 受信 JSSE SSL ハンドシェイク数の集約
- 間隔ごとの受信 JSSE SSL ハンドシェイク呼び出し数

Frontends | Client_to_CTG_Aggregates ノード

［Client_to_CTG_Aggregates］ノードを選択すると、Introscope Investigator は、以下のメトリックが表示される［クライアント集約のグラフィカル サマリ］タブを表示します。

- TCP 集約オープン数
- TCP フロー集約数
- TCP フロー応答時間（ミリ秒）
- SSL 集約オープン数
- SSL フロー集約数
- SSL フロー応答時間（ミリ秒）

Introscope Investigator ツリーの［Client_to_CTG_Aggregates］ノード下には、以下の個別のメトリック（いくつかはタブ ビューに表示）と、いくつかのサブノードがあります。

- EPI エラーの集約
- トランザクション エラー数の集約
- 間隔ごとの CICS リソース例外
- 間隔ごとの CICS Txn 異常終了例外

- 間隔ごとの EPI ゲートウェイ例外
- 間隔ごとの EPI 要求例外
- 間隔ごとのリソース例外
- SSL 集約フロー
- SSL フロー集約数
- SSL フロー応答時間（ミリ秒）
- TCP 集約オープン数
- TCP フロー集約数
- TCP フロー応答時間（ミリ秒）

以下のセクションでは、Client_to_CTG_Aggregates ノード下のサブノードについて説明します。

Frontends | Client_to_CTG_Aggregates | BASE_ECI_EPI サブノード

このノードを選択すると、Introscope Investigator はクライアント集約サマリ タブを表示します。 このタブには、以下のメトリックのグラフが表示されます。

- 呼び出されたプログラム数の集約
- プログラム平均応答時間（ミリ秒）
- 間隔ごとのプログラム呼び出し数
- オープンな接続数の集約

Frontends | Client_to_CTG_Aggregates | CCF_ECI サブノード

このノードを選択すると、Introscope Investigator はクライアント集約サマリ タブを表示します。 このタブには、以下のメトリックのグラフが表示されます。

- 呼び出されたプログラム数の集約
- プログラム平均応答時間（ミリ秒）

- 間隔ごとのプログラム呼び出し数
- オープンな接続数の集約

このタブは、BASE_ECI_EPI ノードに対して表示されたものに類似しています。

Frontends | Client_to_CTG_Aggregates | CCF_EPI サブノード

このノードを選択すると、Introscope Investigator はクライアント集約サマリ タブを表示します。 このタブには、以下のメトリックのグラフが表示されます。

- 呼び出されたトランザクション数の集約
- トランザクション平均応答時間（ミリ秒）
- 間隔ごとのトランザクション呼び出し数
- オープンな接続数の集約

Frontends | Client_to_CTG_Aggregates | JCA_ECI サブノード

このノードを選択すると、Introscope Investigator はクライアント集約サマリ タブを表示します。 このタブには、以下のメトリックのグラフが表示されます。

- 呼び出された CICS プログラム数の集約
- 間隔ごとのプログラム呼び出し数
- プログラム平均応答時間（ミリ秒）
- オープンな接続数の集約

Frontends | Client_to_CTG_Aggregates | JCA_EPI サブノード

このノードを選択すると、Introscope Investigator はクライアント集約サマリ タブを表示します。 このタブには、以下のメトリックのグラフが表示されます。

- 呼び出されたトランザクション数の集約
- トランザクション平均応答時間（ミリ秒）
- 間隔ごとのトランザクション呼び出し数
- オープンな接続数の集約

Frontends | Client_to_CTG_Details ノード

［Client_to_CTG_Details］ノードは、すべてのゲートウェイにわたってエンド ツー エンドの集約応答時間を提供します。 このノードを選択すると、Introscope Investigator は［すべてのクライアント ゲートウェイの表サマリ］タブを表示します。 このタブには、以下の列が表示されます。

- サーバ
- プログラム名
- 集約数
- プログラムの平均応答時間（ミリ秒）
- 応答/間隔
- エラー/間隔
- ストール数

ECI ホスト詳細サブノード

［Client_to_CTG_Details］ノード下には、CICS プログラムをホストするサーバに対応するサブノードがあります。 これらのサーバ サブノードのいずれかを選択すると、Introscope Investigator ビューア ペインの［クライアント ゲートウェイ表サマリ］タブに、以下のメトリックが表示されます。

- 集約数
- プログラムの平均応答時間（ミリ秒）
- 応答/間隔
- エラー/間隔
- ストール数

ECI プログラム詳細サブノード

ECI ホスト サブノードのいずれかを展開すると、ツリーに、サーバ上の各プログラムのサブノードがさらに表示されます。 これらのプログラム ノードを選択すると、［プログラム詳細ビュー］タブが表示されます。 このタブには、以下のメトリックに対するグラフが表示されます。

- 要求数の集約
- 間隔ごとの呼び出し数

- プログラムの平均応答時間（ミリ秒）
- エラー数の集約

Frontends | Client_to_CTG_Details トランザクション ビュー ノード

Introscope Investigator ツリーの Frontends | Client_to_CTG_Details ノード下で、各サブノードはそれぞれ EPI ホストに対応します。

ホスト ID サブノードを選択すると、Introscope Investigator ビューア ペインに［クライアント ゲートウェイ表サマリ］タブが表示されます。 このタブには、ホスト上で実行される各プログラムに関して以下のメトリックが表示されます。

- 集約数
- プログラムの平均応答時間（ミリ秒）
- 応答/間隔
- エラー/間隔
- ストール数

各ホスト サブノードの下には、さらにそのホスト上で実行される各プログラムのサブノードがあります。 これらのプログラム サブノードのいずれかを選択すると、Introscope Investigator ビューア ペインの［プログラム詳細ビュー］タブに、これらのメトリックのグラフが表示されます。

- 要求数の集約
- 間隔ごとの呼び出し数
- プログラムの平均応答時間（ミリ秒）
- エラー数の集約

Frontends | Client_to_CTG_JSSE Session ノード

［Client_to_CTG_JSSE Session］ノードは、すべてのゲートウェイにわたってエンド ツー エンドの集約応答時間を提供します。 このノードを選択すると、Introscope Investigator は［CTG ゲートウェイへの JSSE］タブを表示します。 このタブには、以下のメトリックのグラフが表示されます。

- 作成された SSL セッション数の集約
- SSL ハンドシェイク平均応答時間（ミリ秒）
- 間隔ごとの SSL セッション呼び出し数
- 間隔ごとの SSL 暗号化データ要求数

Introscope Investigator ツリーで、［Client_to_CTG_JSSE Session］ノードの下に以下のメトリックが表示されます。

- Aggregate SSL Data Flows
- Aggregate SSL Handshakes
- Aggregate SSL Opens
- SSL Data Flow Response Time (ms)
- SSL Data Flows Per Interval
- SSL Handshake Allocate Response Time (ms)
- Stall Count

Introscope Investigator タブ ビューの使用

ペインの最上部のタブを使用して、Introscope Investigator ビューア ペインに表示されるビューを変更できます。 ツリー上の CTG に固有のノードの多くには、すぐに使用できる特別なサマリ ビューがあります。

以下のタブは、使用する拡張機能にかかわらず、Introscope Workstation で標準的に使用できます。

- 全般 - ［全般］タブは、Introscope Investigator ツリー内のエージェント下で、いずれかの項目が選択されている場合のデフォルト タブです。ライブ データまたは選択された履歴期間に対してメトリックが選択されている場合、［全般］タブによりメトリックが視覚化されます。ツリー内のノードについては、［全般］タブに、Introscope Investigator 階層内でのそのノード オブジェクトへのパスが表示されます。

- 概要 - ［概要］タブは、Introscope Investigator ツリー内でエージェントが選択されているときに使用可能であり、アプリケーションの監視を行うことができます。 この情報では、稼働状況の高レベル インジケータおよび関連するイベントのログ、およびメトリックの履歴情報が表示されます。
- 検索 - ［検索］タブは、メトリックを含む Introscope Investigator ツリーのノードが選択されているときに使用可能であり、メトリックを迅速に検索できます。
- 追跡 - ［追跡］タブは、Introscope Investigator ツリーでリソースまたはコンポーネントが選択されているときに使用可能であり、追跡ビューアに類似しています。 この情報には、現在選択されているリソースまたはコンポーネントが参加していたトランザクション追跡がリスト表示されます。
- エラー - ［エラー］タブは、Introscope Investigator ツリーでリソースまたはコンポーネントが選択されている場合に使用可能であり、選択した項目のエラーとその詳細をリスト表示します。

注: 標準の Introscope タブの詳細については、「*CA APM Workstation ユーザ ガイド*」を参照してください。

ダッシュボードでの CTG 拡張機能のデータの表示

CTG の拡張機能には、事前設定された複数の Introscope ダッシュボードおよびアラートが用意されています。

CTG クライアントまたはサーバのすべてのダッシュボードは、CTG クライアントまたは CTG サーバから始まり、Introscope にインストールされているその他の管理モジュールとは区別されます。

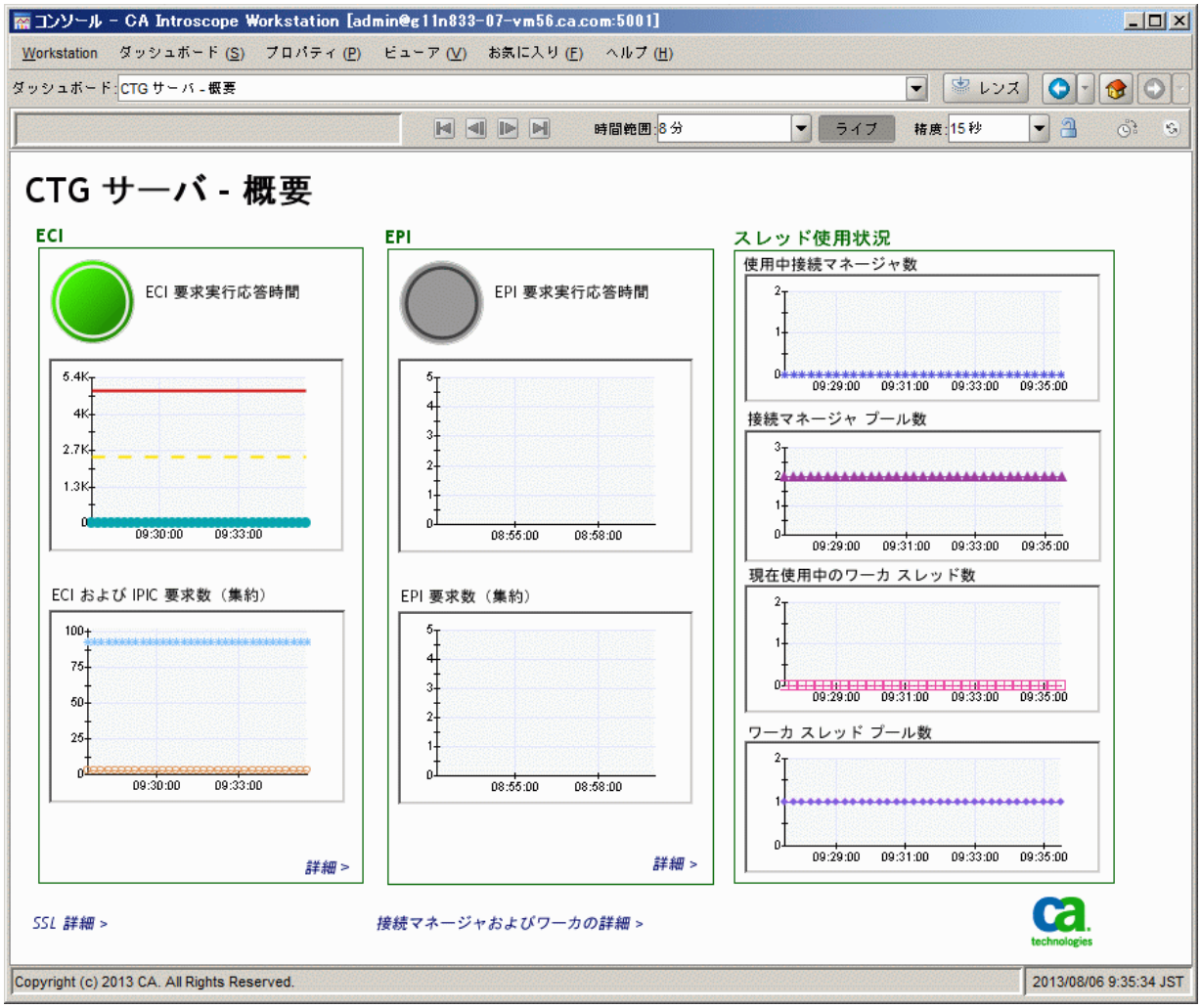
ダッシュボードを表示するには、Introscope Workstation を起動しコンソールを開きます。 CTG の拡張機能には、これらのダッシュボードに表示される多数のメトリックに関するデフォルトの警告/危険アラートしきい値が付属しています。

注: CTG の拡張機能に含まれるパネルとダッシュボードの一部は、CTG アプリケーションの独自のパターンにより空白のままの場合があります。

[CTG サーバ - 概要]ダッシュボード

[CTG サーバ - 概要] ダッシュボードには、Introscope によって監視されている 3 つの主要な CTG サーバ領域のサマリが表示されます。 また、これらのダッシュボードには、これらの領域の詳細を提供するダッシュボードおよび以下のダッシュボードへのリンクが含まれます。

- CTG クライアント - Java ゲートウェイおよび SSL
- CTG サーバ - 接続マネージャおよびワーカ



信号機は、基になるいずれかのメトリックが、その定義されたしきい値を超えたかどうかを視覚的に示します。

［スレッド使用状況］セクションでは、CTG による接続およびワーカ スレッドの現在の使用状況について説明します。一般に、接続スレッドは、クライアントが CTG に接続すると割り当てられ、クライアントが CTG から切断するとき解放されます。 CTG と CICS の間に流れる各アクティブな要求に対して、*ワーカ スレッド*が割り当てられます。

［CTG サーバ - 概要］ダッシュボードには、以下のメトリックが表示されます。

- ECI
  - ECI 要求実行反応時間
  - ECI 要求数
- EPI
  - EPI 要求実行反応時間
  - EPI 要求数

Currently in Use メトリックは、CTG サーバ コードによって、要求（または応答）が処理中であるスレッド数を反映しています。 Pool Count メトリックは、そのエンティティ（接続マネージャまたはワーカ）に割り当てられるスレッドの数を反映しています。プール数は、*ctg.ini* ファイル内の定義に従って、*initconnect* パラメータ（接続マネージャ スレッド用）と *initworker* パラメータ（ワーカ スレッド用）で初期的に設定されます。ワークロードの増大に伴って、対応するプール数が、*ctg.ini* の *maxconnect* パラメータおよび *maxworker* パラメータで指定された最大値まで増加します。Pool Count に記録されたピーク値を使用して、*ctg.ini* ファイル内の関連する *maxconnect* パラメータと *maxworker* パラメータを微調整できます。

［CTG サマリ - サーバ ECI アクティビティ］ダッシュボード

［CTG サマリ - サーバ ECI アクティビティ］ダッシュボードは、問題を起こしている可能性がある実行およびオブジェクトの簡単なスナップショットを提供します。

［CTG サマリ - サーバ ECI アクティビティ］ダッシュボードには、以下のメトリックが表示されます。

- ECI 全ゲートウェイ - 集約サマリ
  - 呼び出しプログラム数
  - 間隔ごとのプログラム呼び出し数
  - プログラムの平均応答時間
- ネットワークの集約
  - 間隔ごとの TPC ソケット新規受け入れ数
  - 間隔ごとの JSSE/SSL 呼び出し数
  - TCP ソケット受信帯域幅
  - TCP ソケット送信帯域幅
  - JSSE/SSL 新規集約数

注: ［TCP ソケット受信帯域幅］および［TCP ソケット送信帯域幅］の表示を有効にするには、IntroscopeAgent.profile でプロパティ *introscope.agent.sockets.reportRateMetrics=true* を設定します。

［CTG サーバ - グローバル統計情報］ダッシュボード

［CTG サーバ グローバル統計情報］ダッシュボードは、CTG サーバの観点から全体的な CTG 統計の簡単なスナップショットを提供します。 上方の隅のリンクから、［CTG サーバ - 概要］ダッシュボードに移動できます。

［CTG RequestExit - グローバル統計情報］ダッシュボードには、以下のメトリックが表示されます。

- 処理済み I/O 要求（集約）
- 間隔ごとの I/O 要求処理数
- クライアントからの受信接続要求（間隔ごと）
- CICS への ECI/IPIC/EPI 合計要求数（間隔ごと）
- 使用内の接続マネージャ スレッド
- 接続マネージャ スレッド プール数
- 現在使用中のワーカ スレッド
- ワーカ スレッド プール数

[CTG サーバ - ECI/IPIC 要求]ダッシュボード

［CTG サーバ - ECI 要求］ダッシュボードは、問題を引き起こしている可能性がある実行およびオブジェクトの簡単なスナップショットを提供します。 上方の隅のリンクから、［CTG サーバ - 概要］ダッシュボードに移動できます。

このダッシュボードには、以下のメトリックが表示されます。

- 実行パフォーマンス
  - 要求実行時間
  - リスト実行時間
  - 要求実行数
  - リスト実行数
- 要求の初期化/終了およびオブジェクトの書き込み/読み取り
  - オブジェクトの書き込み/読み取り時間
  - オブジェクトの書き込み/読み取り数

「*オブジェクトの書き込み/読み取り*」は IBM CTG 用語であり、CTG クライアントと CTG サーバ間で交換されるサービス メッセージ（パケット）を示すために使用されます。 この測定のグループは、送信されたそのようなメッセージの合計数、およびそれらの平均応答時間を示します。 CTG サーバが過負荷（ワーカ スレッドまたは接続スレッドの数が少なすぎる）になると、これらのメッセージはキューに入れられて利用可能なスレッドを待機し、応答時間が増大します。

[CTG サーバ - EPI 要求]ダッシュボード

［CTG サーバ - EPI 要求］ダッシュボードは、サーバ要求のパフォーマンス サマリを提供します。 上方の隅のリンクから、［CTG サーバ - 概要］ダッシュボードに移動できます。

［CTG サーバ - EPI 要求］ダッシュボードには、以下のメトリックが表示されます。

- 実行パフォーマンス
  - トランザクション実行時間
  - トランザクション開始数の集約

- オブジェクトの読み取り/書き込み
  - オブジェクトの書き込み/読み取りオーバーヘッド時間
  - オブジェクトの書き込み/読み取りオーバーヘッド数
- CICS CP 要求
  - CP 要求実行時間
  - CP 要求実行数

CTG が CICS への EPI セッションを確立できない場合、以下のメトリックは生成されません。

- トランザクション実行時間
- 書き込み/読み取りオーバーヘッド時間
- トランザクション開始数の集約
- オブジェクトの書き込み/読み取りオーバーヘッド数

代わりに、エラー メッセージ（［エラー］タブに表示）がログに記録され、接続の失敗を引き起こすエラーを示します。

［CTG サーバ - 接続マネージャおよびワーカ］ダッシュボード

［CTG サーバ - 接続マネージャおよびワーカ］ダッシュボードは、問題を引き起こしている可能性がある接続マネージャおよびワーカのパフォーマンスの簡単なスナップショットを提供します。 上方の隅のリンクから、［CTG サーバ - 概要］ダッシュボードに移動できます。

このダッシュボードには、以下のメトリックが表示されます。

- クライアント接続マネージャ パフォーマンス
  - ディスパッチ応答時間
  - 要求応答時間を送信
  - ディスパッチ数
  - 応答数を送信
- ワーカ スレッド パフォーマンス
  - ディスパッチ応答時間
  - 実行/クローズ応答時間

- ワーカ ディスパッチ数
- 実行/クローズ数

[CTG サーバ - SSL]ダッシュボード

［CTG サーバ - SSL］ダッシュボードは、問題を引き起こしている可能性がある SystemSSL および JSSE SSL のパフォーマンスの簡単なスナップショットを提供します。 上方の隅のリンクから、［CTG サーバ - 概要］ダッシュボードに移動できます。

［CTG サーバ - SSL］ダッシュボードには、以下のメトリックが表示されます。

- JSSE/SSL サーバ ソケット
  - ソケット受け入れ/クローズ応答時間
  - ソケット受け入れ/クローズ数
- SSL ソケット（レガシー）
  - ソケット クローズ応答時間
  - ソケット クローズ数
- JSSE ソケット
  - JSSE ソケット クローズ応答時間
  - JSSE のソケット クローズ数

[CTG クライアント - 概要]ダッシュボード

［CTG クライアント - 概要］ダッシュボードには、Introscope によって監視される 3 つの主要な CTG サーバ領域のサマリが表示されます。 また、このダッシュボードには、これらの 3 つの領域の詳細を提供するダッシュボードへのリンクが含まれます。信号機は、基になるいずれかのメトリックが、その定義されたしきい値を超えたかどうかを視覚的に示します。

［CTG クライアント - 概要］ダッシュボードには、以下のメトリックが表示されます。

- Java ゲートウェイおよび SSL セッション
  - ゲートウェイ フロー応答時間
  - JSSE SSL セッション割り当て応答時間
- EPI
  - EPI 接続オペレーション応答時間
  - EPI ターミナル トランザクション オペレーション応答時間
- ターミナルおよびターミナル要求
  - ターミナルおよびターミナル要求送信応答時間
  - ターミナル接続および切断応答時間

［CTG クライアント - Java ゲートウェイおよび SSL セッション］ダッシュボード

［CTG クライアント - Java ゲートウェイおよび SSL セッション］ダッシュボードは、問題を引き起こしている可能性があるゲートウェイおよびセッションの簡単なスナップショットを提供します。 上方の隅のリンクから、［CTG クライアント - 概要］ダッシュボードに移動できます。

［CTG クライアント - Java ゲートウェイおよび SSL セッション］ダッシュボードには、以下のメトリックが表示されます。

- Java ゲートウェイ パフォーマンス
  - フロー応答時間
  - 現在のアクティブ セッション数
  - TCP 集約セッション呼び出し数
  - オープン応答時間
  - クローズ応答時間
- JSSE SSL セッション パフォーマンス
  - JSSE 集約 SSL セッション数
  - JSSE SSL セッション割り当て時間
  - 間隔ごとの JSSE SSL セッション呼び出し数

[CTG クライアント - EPI 要求]ダッシュボード

［CTG クライアント - EPI 要求］ダッシュボードは、ゲートウェイ接続のスナップショットを表示します。

ダッシュボードには、以下のメトリックが表示されます。

- ゲートウェイ応答時間メトリック
  - EPI ゲートウェイ接続オペレーション応答時間
  - EPI 集約ゲートウェイ接続数
- ターミナルおよび基本画面ハンドラ メトリック
  - EPI ターミナル接続オペレーション応答時間
  - EPI ターミナル トランザクション オペレーション応答時間
  - EPI 基本画面ハンドラ接続数

[CTG クライアント - EPI ターミナル要求]ダッシュボード

［CTG クライアント - EPI ターミナル要求］ダッシュボードは、問題を引き起こしている可能性があるターミナルおよびターミナル要求アクティビティの簡単なスナップショットを提供します。 上方の隅のリンクから、［CTG クライアント - EPI ターミナル要求］ダッシュボードに移動できます。

このダッシュボードには、以下のメトリックが表示されます。

- ターミナル アクティビティおよびパフォーマンス
- 送信応答時間
- 接続および切断時間
- 送信数
- 接続数

[Introscope ChangeDetector]ダッシュボード

［Introscope ChangeDetector］ダッシュボードは、問題を引き起こしている可能性がある CTG 設定ファイルおよび .JAR ファイルに対する変更のサマリを提供します。

［Introscope ChangeDetector］ダッシュボードには、以下のメトリックが表示されます。

- 間隔ごとの変更検出数
- 間隔ごとの最多変更検出数
- 前回のエージェント再起動以降の総変更数
- 前回のエージェント再起動以降の完了スキャン数

Introscope の警告/危険アラートしきい値の変更

多くの CTG 拡張機能ダッシュボードにはアラートが含まれています。

Introscope の警告/危険アラートしきい値を変更する方法

1. Introscope Investigator で、［管理モジュール］-［CTGCLIENT］-［アラート］に移動し、アラートを選択します。

   下部ペインにアラート定義が表示されます。
2. アラート定義ペインで［期間］フィールドを変更して、どれくらいの頻度でパフォーマンス メトリックを［警告しきい値］および［危険しきい値］と比較するか指定します。
3. 組織のサービス レベルに対して適切な［警告しきい値］と［危険しきい値］を指定します。
4. ［適用］をクリックして変更を適用します。

CTG Transaction Tracer

CTG Transaction Tracer は、サーバ名とプログラム名によって追跡する機能を提供します。 この機能は、主に CTG クライアント（Websphere など）と CTG サーバの間のトランザクション追跡を実行するためのものです。CICS メインフレーム製品間 Transaction Tracer SYSVIEW 機能に結び付ける必要はありません。 また、トレーサは、カスタマが指定した共通領域内のユーザ相関フィールドを抽出し包含する機能、応答時間しきい値を指定する機能、および特定のプログラム名を追跡する機能をサポートします。この機能を使用すると、エージェント プロパティ エントリを使用してカスタマイズし、プログラム名ごと、またはカスタム拡張機能を最上部に置くことにより、応答時間しきい値に基づく付加価値付きの CTG 追跡を作成できます。

PP CTG Transaction Tracer による特殊文字の処理

Introscope Workstation で実行される Introscope トランザクション追跡の相関ロジックでは、ユーザが生成する相関 ID での特殊文字の使用が厳しく制限されます。Introscope Workstation の相関サポートによって許可されない特殊文字は、自動的にアンダースコアに変換されます。

以下の特殊文字は、常にアンダースコアに変換されます。

```
:   -   +   (   )   [   ]   *   "   ~   ^   ?
```

以下の追加の特殊文字のセットも、デフォルトでアンダースコアに変換されます。

```
¥   /   {   }   |   ,   .   ;   =   '   <   >
```

ただし、これらの特殊文字を許可させる（そしてすべての特殊文字と追跡を使用できるようにする）には、以下のプロパティを IntroscopeAgent.profile に追加します。

```
introscope.ctg.trantracer.corrid.specialcharstrip=false
```

この値は、デフォルトでは「true」です。これは、すべての特殊文字をアンダースコアに変換し、フロントエンドとバックエンドの追跡を相関させます。

相関 ID で重複したアンダースコアを削除する（複数のアンダースコアを単一のアンダースコアにする）場合は、IntroscopeAgent.profile で以下のプロパティを設定する必要があります。

```
introscope.ctg.trantracer.corrid.underscorestrip=true
```

CTG Transaction Tracer プロパティの詳細なリスト

以下の表に、IntroscopeAgent.profile ファイルで設定できる CTG Transaction Tracer プロパティの詳細なリストを示します。

プロパティ	使用方法
introscope.ctg.trantracer.publish.metrics	単純なメトリック（応答時間と呼び出し数）を有効にする必要があるかどうかを示すブーリン。 これは、「軽量」トレーサを意味することに注意してください。 デフォルト = true
introscope.ctg.trantracer.publish.trantrace	トランザクション追跡を有効にする必要があるかどうかを示すブーリン。 デフォルト = true
introscope.ctg.trantracer.corridscan.offset	ユーザ相関 ID が、共通領域の先頭から、どのオフセットで開始するかを示す整数。 デフォルト = 147
introscope.ctg.trantracer.corridscan.length	抽出されるユーザ相関 ID の長さを示す整数。 デフォルト = 100
introscope.ctg.trantracer.corrid.format.id	EM Transaction Tracer 文字列のエントリで許可されていない不正な文字（たとえば、「:」（コロン）および「-」（ダッシュ））をユーザ相関 ID でスキャンする必要があるかどうかを示すブーリン。
introscope.ctg.trantracer.corridscan.starttrace	追跡の終了時ではなく、追跡の開始時にユーザ相関 ID を抽出する必要があるかどうかを示すブーリン。 デフォルト = false
introscope.ctg.trantracer.rspthreshhold.value	このトレーサで許可された最大応答時間しきい値を示す整数。 この値を超える応答時間は、EM に対するトランザクションが追跡されます。 その値を下回る値は追跡されません。 デフォルト = 0 = RSP しきい値確認を実行しません。

プロパティ	使用方法
introscope.ctg.trantracer.program.name	CICS プログラム名が完全または部分的に一致していることを示す文字列。一致する値は、EM に対するトランザクションが追跡されます。 一致しない値は追跡されません。 デフォルト = NULL = 一致するプログラムはありません
introscope.ctg.trantracer.program.match.criteria	実行する必要があるプログラム名の一致の種類を定義する整数。 1 = フル ネームの一致 2 = 上記の名前文字列で始まるプログラム名 3 = 上記の名前文字列で終わるプログラム名 0 = 一致なし。 デフォルト = 0 （一致なし）。
introscope.ctg.trantracer.metric.HLQname	必要な Investigator ツリーの高レベル名を示す文字列。 デフォルト： Backends|CTG_to_CICS_ECI_IPIC_Trace および Frontends|Client_to_CTG_Trace
introscope.ctg.trantracer.debug	エージェント ログへのデバッグ メッセージを追跡する必要があるかどうかを示すブーリン。 デフォルト = false

付録 A: CTG パフォーマンス メトリック

この付録では、CTG の拡張機能でパフォーマンス メトリックが収集され、Introscope Investigator に表示される、CTG クラスおよびメソッドについて説明します。

このセクションには、以下のトピックが含まれています。

Frontend メトリック

フロントエンド メトリックは CTG クライアント インスツルメンテーションによってキャプチャされます。 データは、アプリケーション サーバ（WebSphere など）内、またはスタンドアロン（WebSphere の外部）を実行しており、インスツルメントされている、ユーザが作成した CTG クライアント アプリケーション内でキャプチャされます。 フロントエンド メトリックは、実行されている CTG アプリケーションのタイプ（JCA、CCF、または ECI か EPI のいずれかのベース クラス）をキャプチャおよび測定するために使用されます。

メトリックは、集約レベル（たとえば、WebSphere アプリケーション サーバによって呼び出されているすべての ECI プログラム）でキャプチャされます。 詳細なメトリックは、個別のプログラム レベルまたはトランザクション レベル（たとえば呼び出された個別の ECI プログラムの統計）でキャプチャされます。 多くのメトリック（特に詳細なメトリック）は、BlamePoint トレーサを使用するため、一般的な形式が提供されます。

メトリック	定義
Aggregate Request Count	プログラムまたはトランザクションに対する要求が開始された合計回数。オープン接続に失敗する場合（以下を参照）は、このメトリックが表示されない場合があります。 **注**: Aggregate Service Count の注を参照してください。
Aggregate Service Count	EPI は、複数のセットアップおよび実際の CICS トランザクションを実行しない分解フロー（AddTerminal、PurgeTerminal、DeleteTerminal）を使用します。それらは、CICS へのアクティブな 3270 接続をセットアップする、唯一のサービス要求です。このメトリックは、これらすべてのサービス タイプ要求数の集約を保持します。 **注**: Aggregate Request Count は、実際の 3270 トランザクションの呼び出しおよび応答（つまり、EPI 要求による 3270 バッファ、および CICS トランザクションからの関連する 3270 応答バッファの送信）を記録します。
Aggregate Errors Count	オープン接続、またはプログラムやトランザクションの開始の要求がエラーによって失敗した回数の合計。 このメトリックは、実際にエラーが発生した場合のみ表示されます。
Average Response Time	クライアント エンドから処理され測定される要求の平均応答時間（つまり、往復時間の合計）。
Concurrent Invocations	進行中だったプログラムまたはトランザクション要求の同時進行中の呼び出し数。
Errors per Interval	最新の測定間隔中（通常は 7 ～ 15 秒）にレポートされたエラーの数の合計。

メトリック	定義
Responses per Interval	最新の測定間隔中にレポートされた、CTG からの応答数の合計。
Stall Count	プログラムの要求が開始されたが、ストールの検出時間制限内に CICS による応答がない場合に、ストールが検出されたかどうかを示します。 ストールの検出時間制限は、*IntroscopeAgent.profile* の *introscope.agent.stalls.thresholdseconds=nn* プロパティによって設定されます。 *nn* 値は、ストールの検出時間を秒単位で指定します。

Client_to_CTG_Aggregates の場合は、以下の追加メトリックが追跡されます。

メトリック	定義
Aggregate Opens	CTG クライアントから CTG サーバに発行された、オープンな TCP または SSL 接続の合計。CTG オープン接続が実行されるまで、CTG 要求を開始できません。
Open Response Time	クライアント エンドから処理および測定される接続オープン要求の平均応答時間。 つまり、CTG クライアントが CTG サーバ接続をオープンするまでに費やした往復時間の合計。

Frontend Client to CTG aggregates

これらのメトリックは、CTG クライアントから CTG サーバに発行されたすべての ECI 呼び出しおよび EPI 呼び出しの集約数を提供します。 便宜上、呼び出しは CTG API タイプによって分類されます。以下の表にリスト表示されたすべてのメトリックは、BlamePoint メトリックを使用します。

オープン接続要求が失敗した場合は、［Client_to_CTG_Aggregates］ツリーの下に表示される［Open Errors］タブに詳細が表示されます。

メトリック	定義
BASE_ECI_EPI	ベース ECI クラス API を使用するすべての ECI 呼び出し、およびベース EPI クラス API を使用するすべての EPI 呼び出し。
JCA_ECI	より新しい JCA （Java Connector Architecture） ECI API を使用するすべての ECI 呼び出し。
JCA_EPI	より新しい JCA （Java Connector Architecture） EPI API を使用するすべての EPI 呼び出し。
Stall Count	プログラムの要求が開始されたが、ストールの検出時間制限内に CICS による応答がない場合に、ストールが検出されたかどうかを示します。 ストールの検出時間制限は、IntroscopeAgent.profile の *introscope.agent.stalls.thresholdseconds=nn* プロパティによって設定されます。 nn 値は、ストールの検出時間を秒単位で指定します。

Frontend Client to CTG details

これらのメトリックは、CTG クライアントによって呼び出された各プログラムまたはトランザクションの詳細な数を提供します。 便宜上、呼び出しは CLI （ゲートウェイ）サーバによって分類されます。 各ゲートウェイサーバには、呼び出された個別の ECI または EPI プログラムのリストが、プログラムの詳細なメトリックと共に存在します。

バックエンド メトリック

バックエンド メトリックは、Introscope によってインスツルメントされた CTG サーバ内でキャプチャされます。 バックエンド メトリックは、クライアントと CICS システム間の ECI 要求と EPI 要求のフローをキャプチャおよび測定するために使用されます。

メトリックは、集約レベル（たとえば、クライアントが呼び出すすべての ECI プログラム）でキャプチャされます。 詳細なメトリックは、個別のプログラム レベル（たとえば呼び出された個別の ECI プログラムの統計）でキャプチャされます。

各ゲートウェイ サーバには、呼び出された個別の ECI または EPI プログラムのリストが、そのプログラムの詳細なメトリックと共に存在します。詳細なメトリックは、一般的な形式の BlamePoint トレーサを使用します。

メトリック	定義
Aggregate Count Aggregate Request Count	プログラムまたはトランザクションに対する要求が開始された合計回数。 オープン接続に失敗する場合（以下を参照）は、このメトリックが表示されない場合があります。
Aggregate Errors Count	オープン接続、またはプログラムやトランザクションの開始の要求がエラーによって失敗した回数の合計。このメトリックは、実際にエラーが発生した場合のみ表示されます。
Average Response Time	処理される要求の往復時間の合計の平均
Errors per Interval	最新の測定間隔中にレポートされたエラーの総数。
Responses per Interval	最新の測定間隔中にレポートされた、CTG からの応答の総数。
Stall Count	検出されたストールの総数。 プログラムの要求が開始されたが、ストールの検出時間制限内に CICS による応答がない場合に、ストールは発生します。ストールの検出時間制限は、IntroscopeAgent.profile の introscope.agent.stalls.thresholdseconds=nn プロパティによって設定されます。nn は、ストールの検出時間（秒単位）です。

Backend CTG_to_CICS_ECI_IPIC aggregates

これらのメトリックは、CTG クライアントおよび対応する CICS サーバ間で発行されたすべての ECI/IPIC 呼び出しの集約数を提供します。 各ゲートウェイ サーバには、表に示すように、BlamePoint トレーサ メトリックを使用して呼び出された個別の ECI/IPIC プログラムのリストがあります。ゲートウェイ サーバ上では、以下の集約メトリックがゲートウェイ サーバ単位で提供されます。

メトリック	定義
Program Aggregate Count	ECI/IPIC プログラムに対する要求が開始された回数の合計集約数。
Program Average Response Time	CTG サーバから CICS に処理され測定される、ECI/IPIC 要求の平均合計応答時間。つまり、CTG サーバと CICS の間のラウンド トリップ時間の合計です。
Program Invocations per Interval	最新の測定間隔中にレポートされた、ECI/IPIC プログラムの呼び出し数の合計。

Backend CTG Global Statistics

メトリック	定義
Process Client Request Response Time	クライアント要求を読み取り、CICS に転送するまでに必要とされる全体時間。
Send Client Reply Response Time	CICS からの応答を読み取り、要求を行ったクライアントに転送するまでに必要とされる全体時間。
Connection Manager Threads In Use	クライアント接続を処理するために現在使用されているスレッドの数。
Worker Threads In Use	CICS 要求を処理するために現在使用されているスレッドの数。

Backend CTG_to_CICS_EPI aggregates metrics

CTG_to_CICS_EPI メトリックは、分散型（非 z/OS）バージョンの CTG でのみ利用可能です。これらのメトリックは、CTG クライアントおよび対応する CICS サーバ間で発行された EPI コールの集約数を提供します。

メトリック	定義
Transaction Aggregate Count	EPI トランザクションに対する要求が開始された回数の総集約数
Transaction Average Response Time	CTG サーバから CICS に処理され測定される、EPI 要求の平均合計応答時間。 つまり、CTG サーバと CICS の間のラウンド トリップ時間の合計です。
Transaction Invocations per Interval	最新の測定間隔中にレポートされた、EPI トランザクションの呼び出し数の合計。

注: 各ゲートウェイ サーバの下には、表に示すように、呼び出された個別の EPI トランザクションに関する、BlamePoint トレーサ メトリックを使用したリストがあります。

Backend CTG_to_CICS threads

これらのメトリックは、CTG サーバによる、接続およびワーカ スレッドの現在の使用状況を表示します。

CTG ダッシュボード メトリック

以下の表では、拡張ダッシュボードに表示されるラベルと特定の CTG パフォーマンス メトリックの対応について説明します。

右の列では、Introscope Investigator 内の位置の観点からメトリックを説明します。

[CTG クライアント - 概要]ダッシュボード

ラベル	CTG パフォーマンス メトリック
ゲートウェイ フロー応答時間	CTGCLIENT|{.+}JavaGateway:open Response Time (ms)
JSSE SSL セッション割り当て応答時間（ミリ秒）	Client_to_CTG|JSSE Session:SSL Handshake Allocate Response Time

ラベル	CTG パフォーマンス メトリック
EPI 接続オペレーション応答時間	CTGCLIENT|EPI|Gateway: open Response Time (ms) CTGCLIENT|EPI|Terminal:{connect|disconnect|process Connect|processDisconnect|terminate} Response Time (ms)
EPI ターミナル トランザクション オペレーション応答時間	CTGCLIENT|EPI|Terminal:{send|setTransactionData|startTran|terminate} Response Time (ms)
ターミナルおよびターミナル要求送信応答時間	CTGCLIENT|{TerminalTerminalRequest}:send Response Time (ms)
ターミナル接続および切断応答時間	CTGCLIENT|Terminal{connect|disconnect} Response Time (ms)

[CTGClient - EPI]ダッシュボード

ラベル	CTG パフォーマンス メトリック
EPI ゲートウェイ接続オペレーション応答時間	CTGCLIENT|EPI|Gateway:{.*} Response Time (ms)
EPI 集約ゲートウェイ接続数	CTGCLIENT|EPI|Gateway:{.*} Count
EPI 接続オペレーション応答時間	CTGCLIENT|EPITerminal:{connect|diswconnect|processConnect|processDisconnect|terminate} Response Time (ms)
EPI ターミナル トランザクション オペレーション応答時間	CTGCLIENT|EPITerminal{send|setTransactionData|startTran|terminate} Response Time (ms)
EPI モニタおよびターミナル接続数	CTGCLIENT|EPITerminal:Count CTGCLIENT|EPIMonitor:terminalConnectedCount
EPI 基本画面ハンドラ接続数	CTGCLIENT|EPIBasicScreenHandler:terminalConnectedCount

[CTG クライアント - Java ゲートウェイおよび SSL]ダッシュボード

ラベル	CTG パフォーマンス メトリック
JavaGateways: フロー応答時間	CTGCLIENT|{.+}JavaGateway:flow Response Time (ms)
JavaGateways: 合計アクティブ セッション数	CTGCLIENT|{.+}JavaGateway:Count
JavaGateways: オープン応答時間	CTGCLIENT|{.+}JavaGateway:open Response Time (ms)
JavaGateways:クローズ応答時間	CTGCLIENT|{.+}JavaGateway:close Response Time (ms)
JSSE 集約 SSL セッション数	Client_to_CTG|JSSE Session:Aggregate SSL Opens
JSSE SSL セッション割り当て時間（ミリ秒）	Client_to_CTG|JSSE Session:SSL Handshake Allocate Response Time
間隔ごとの JSSE SSL セッション呼び出し数	Client_to_CTG|JSSE Session:SSL Opens Per Interval

[CTG クライアント - ターミナルおよびターミナル要求]ダッシュボード

ラベル	CTG パフォーマンス メトリック
送信数	CTGCLIENT|EPIGateway:{.*}Response Time (ms)
接続および切断時間	CTGCLIENT|EPIGateway:{.*} Count
送信応答時間	CTGCLIENT|TerminalRequest:send Response Time (ms)
ターミナル接続および切断応答時間	CTGCLIENT|TerminalRequest:{release|allocate} Response time (ms)

[CTG サーバ - 概要]ダッシュボード

ラベル	CTGServer パフォーマンス メトリック
ECI 要求実行反応時間	CTGSERVER|ServerECIRequest|execute Response Time (ms)

ラベル	CTGServer パフォーマンス メトリック
ECI 要求数	CTGSERVER|ServerECIRequest|execute Count
EPI 要求実行反応時間	CTGSERVER|ServerEPIRequest|execute Response Time (ms)
EPI 要求数	CTGSERVER|ServerEPIRequest|execute Count

[CTG サーバ - 接続マネージャおよびワーカ]ダッシュボード

ラベル	WebSphere パフォーマンス メトリック
接続マネージャ: ディスパッチ応答時間	CTGSERVER|ConnectionManager:kick Response Time (ms)
接続マネージャ: ディスパッチ数	CTGSERVER|ConnectionManager:kick Count
接続マネージャ： 要求応答時間を送信	CTGSERVER|ConnectionManager:sendReply Response Time (ms)
接続マネージャ: 返信数	CTGSERVER|ConnectionManager|sendReply Count
ワーカ: ディスパッチ応答時間	CTGSERVER|Worker:kick Response Time (ms)
ワーカ: ディスパッチ数	CTGSERVER|Worker:kick Count
ワーカ: 実行/クローズ応答時間	CTGSERVER|Worker:{close|run} Response Time (ms)
ワーカ: 実行/クローズ数	CTGSERVER|Worker:{close|run} Count

[CTG サーバ - ECI 要求]ダッシュボード

ラベル	WebSphere パフォーマンス メトリック
要求実行時間	CTGSERVER|ServerECIRequest:execute Response Time (ms)
リスト実行時間	CTGSERVER|ServerECIRequest:executeList Response Time (ms)
要求実行数	CTGSERVER|ServerECIRequest:execute Count

ラベル	WebSphere パフォーマンス メトリック
リスト実行数	CTGSERVER|ServerECIRequest:executeList Count
要求初期化/終了時間	CTGSERVER|ServerECIRequest:{initialize|terminate} Response Time (ms)
オブジェクトの書き込み/読み取り時間	CTGSERVER|ServerECIRequest:{read|write}Object Response Time (ms)
要求初期化/終了数	CTGSERVER|ServerECIRequest:{initialize|terminate} Count
オブジェクトの書き込み/読み取り数	CTGSERVER|ServerECIRequest:{read|write}Object Count

[CTG サーバ - EPI 要求]ダッシュボード

ラベル	WebSphere パフォーマンス メトリック
要求実行時間	CTGSERVER|ServerEPIRequest:execute Response Time (ms)
リスト実行時間	CTGSERVER|ServerEPIRequest:executeList Response Time (ms)
要求実行数	CTGSERVER|ServerEPIRequest:execute Count
リスト実行数	CTGSERVER|ServerEPIRequest:executeList Count
要求初期化/終了時間	CTGSERVER|ServerEPIRequest:{initialize|terminate} Response Time (ms)
オブジェクトの書き込み/読み取り時間	CTGSERVER|ServerEPIRequest:{read|write}Object Response Time (ms)
CicsCp 要求実行反応時間	CTGSERVER|ServerCicsCpRequest|execute Response Time (ms)
CicsCp 要求数	CTGSERVER|ServerCicsCpRequest|execute Count
要求初期化/終了数	CTGSERVER|ServerEPIRequest:{initialize|terminate} Count
オブジェクトの書き込み/読み取り数	CTGSERVER|ServerEPIRequest:{read|write}Object Count

[CTG サーバ グローバル統計情報]ダッシュボード

ラベル	CTGServer パフォーマンス メトリック
処理済み I/O 要求（集約）	処理された TCP I/O 要求の合計数。
間隔ごとの I/O 要求処理数	最後の 15 秒間隔で処理された TCP I/O 要求の数。
クライアントからの受信接続要求（間隔ごと）	最後の 15 秒間隔で発生した、クライアントから受信した新規要求の数。
CICS への ECI/IPIC/EPI 合計要求数（間隔ごと）	最後の 15 秒間隔で処理された要求（ECI、IPIC、EPI）の合計数。
使用内の接続マネージャ スレッド	クライアント接続を処理するために現在使用されているスレッドの数。
接続マネージャ スレッド プール数	クライアント接続を処理するために現在割り当てられているスレッドの数。
現在使用中のワーカ スレッド	CICS 要求を処理するために現在使用されているスレッドの数。
ワーカ スレッド プール数	CICS 要求を処理するために現在割り当てられているスレッドの数。

[CTG サーバ - SSL]ダッシュボード

ラベル	WebSphere パフォーマンス メトリック
JSSE SSL 集約受信 SSL ハンドシェイク	受信 SSL ハンドシェイクの総数
間隔ごとの JSSE SSL 受信 SSL ハンドシェイク数	間隔ごとの SSL ハンドシェイクの数

Request Exit メトリック

以下の表では、Request Exit 機能のメトリックについて説明します。

Backends | CTG_to_CTG_ECI_IPIC_RequestExit メトリック

この表では、Investigator ノードの下に表示される Backends | CTG_to_CICS_ECI_IPIC_RequestExit のメトリックの詳細について説明します。

詳細なメトリック名	説明
Program Average Response Time (usec)	すべてのプログラムの平均応答時間
Program Invocations Per Interval	すべてのプログラムに関する間隔ごとの呼び出し合計数
Program Aggregate Count ECI_IPIC	プログラムが呼び出された回数の集約
Program Aggregate Errors	すべてのプログラムのエラー数の集約
{server_name|program_name}:Average Response Time (usec)	プログラムの平均応答時間
{server_name|program_name}:Responses Per Interval	プログラムに対する間隔ごとの応答数（呼び出し数）
{server_name|program_name}:Concurrent Invocations	指定された（15 秒）間隔内のプログラムの同時呼び出しの数
{server_name|program_name}:Stall Count	指定された（15 秒）間隔内に、プログラムに対して記録されたストール/ハング アップの数
{server_name|program_name}:Errors Per Interval	指定された（15 秒）間隔内に、プログラムに対して記録されたエラーの数
{server_name|program_name}:Aggregate Program Count ECI_IPIC	プログラムが呼び出された回数の合計集約数
{server_name|program_name}:Aggregate Errors	プログラムが呼び出されたときに、エラーがレポートされた時間の合計集約数
{server_name|program_name}:CommArea Aggregate Request Data	このプログラムに関して CICS まで送信された共通領域データの合計バイト数
{server_name|program_name}:CommArea	このプログラムに関して CICS から受信された共通領域データの合計バイト数

CTG_Global_Statistics_RequestExit メトリック

この表では、CTG_Global_Statistics_RequestExit のメトリックの詳細について説明します。メトリックの詳細な名前は、IBM がメトリックに割り当てる名前に対応します。 この表により、Introscope がメトリックと呼ぶものと、IBM のメトリック定義との間の混乱を最小限にとどめることができます。

詳細なメトリック名	説明
CICS Server:Amount of CICS request data	接続されている CICS サーバに送信された要求データ量（バイト単位）。 この量には、アプリケーションおよび CICS プロトコル データの両方が含まれます。
CICS Server:Amount of CICS response data	接続されている CICS サーバから受信した応答データ量（バイト単位）。 この量には、アプリケーションおよび CICS プロトコル データの両方が含まれます。
CICS Server:Current number of installed terminals	現在、確立およびインストールされている（EPI）ターミナル セッション数。
CICS Server:Current number of Orphaned CICS requests	所有するアプリケーションがタイムアウトになったか終了した CICS からの応答を待機している要求数。
CICS Server:Maximum number of active requests	設定ファイルに定義されるアクティブな要求に関して定義された最大数。
CICS Server:Number of CICS servers	ゲートウェイ デーモンが要求の送信を試行した送信先の CICS サーバの数。
CICS Server:Number of IPIC session failures	CICS サーバへの IPIC セッションでの失敗の数。
CICS Server:Number of IPIC session in use	CICS サーバで使用中の IPIC セッションの数。
CICS Server:Number of active requests	現在、クライアント デーモンで実行されているアクティブな要求の数
CICS Server:Number of connect failures	CICS サーバへの接続の試行が失敗した数。
CICS Server:Number of defined CICS servers	設定ファイルで定義された CICS サーバの数。
CICS Server:Number of lost connections	CICS サーバとの確立された接続が失われた回数。

詳細なメトリック名	説明
CICS Server:Amount of CICS request data	接続されている CICS サーバに送信された要求データ量（バイト単位）。 この量には、アプリケーションおよび CICS プロトコル データの両方が含まれます。
CICS Server:Number of negotiated IPIC sessions	CICS サーバとネゴシエートされた IPIC セッションの数。
CICS Server:Number of orphaned CICS requests	応答を待っている、タイムアウトした、または終了した CICS サーバ要求の数。
CICS Server:Number of requests processed	処理された CICS サーバ要求の数。
CICS Server:Number of requests waiting on a response	現在、CICS サーバからの応答を待っている要求の数。
CICS Server:Number of terminal install requests	CICS サーバに送信されたターミナル インストール要求の数。
CICS Server:Number of terminal uninstall events	ゲートウェイによって処理されたターミナル アンインストール イベントの数。
CICS Server:Number of timed out connections	CICS サーバへの接続がタイムアウトになった回数。
CICS Daemon:Client daemon running Time	クライアント デーモンが正常に初期化されてからの時間（秒）。
CICS Daemon:Number of connected Client applications	現在、クライアント デーモンに接続されているクライアント アプリケーション プロセスの数。
CICS Daemon:Number of requests Processed	処理された API 呼び出し要求の数。
Connection Manager:Pool Count	ピーク時に、クライアントに同時に割り当てられている接続マネージャ スレッドの数。
Connection Manager:Thread dispatch count	クライアントから確立された接続の数を表す、接続マネージャ スレッドの割り当て数。
Connection Manager:Threads allocated to clients	現在、クライアントに割り当てられている接続マネージャ スレッドの数。
Gateway Daemon:Amount of client request data	クライアント アプリケーションから受信した要求データの量（バイト単位）。
Gateway Daemon:Amount of client response data	クライアント アプリケーションに送信された応答データの量（バイト単位）。

詳細なメトリック名	説明
CICS Server:Amount of CICS request data	接続されている CICS サーバに送信された要求データ量（バイト単位）。 この量には、アプリケーションおよび CICS プロトコル データの両方が含まれます。
Gateway Daemon:Average Gateway daemon response time	ゲートウェイ デーモンがクライアントからの API 要求に応答するために要した平均時間（ミリ秒）。
Gateway Daemon:End of interval time HHMMSS	次にスケジュールされた間隔統計リセット イベントのローカル時刻。 この時点では、間隔統計はすべてゼロにリセットされます。
Gateway Daemon:Extended LUW transactions committed	コミットされた拡張 LUW ベースの 1 フェーズ トランザクション数。
Gateway Daemon:Extended LUW transactions rolled back	ロールバックされた拡張 LUW ベースの 1 フェーズ トランザクション数。
Gateway Daemon:Failed SYNCONRETURN Transactions	現在の間隔で失敗した SYNCONRETURN トランザクションの数。
Gateway Daemon:Gateway daemon running time	ゲートウェイ デーモンが正常に初期化されてからの時間（秒）。
Gateway Daemon:Interval running Time	最後の間隔リセット イベントからの時間（秒）（現在の間隔の経過期間）。
Gateway Daemon:Length of the statistics interval HHMMSS	ゲートウェイ デーモンによって使用されている統計間隔の持続期間。
Gateway Daemon:Logical End of Day time HHMMSS	ゲートウェイ デーモンによって 1 日の論理的な終了として指定されたローカル時刻。 その時点では、間隔統計はすべてゼロにリセットされます。
Gateway Daemon:Number of CICS request exit calls	ゲートウェイ デーモンによって呼び出された Request Exit 呼び出しの数。
Gateway Daemon:Number of Extended LUW transactions	最後の間隔で処理された拡張 LUW トランザクションの数。
Gateway Daemon:Number of SYNCONRETURN transactions	最後の間隔で処理された SYNCONRETURN トランザクションの数。
Gateway Daemon:Number of requests processed	最後の間隔で処理された要求の数。
Gateway Daemon:Successful SYNCONRETURN transactions	ゲートウェイ デーモン プロセスで成功した SYNCONRETURN トランザクションの数。

詳細なメトリック名	説明
CICS Server:Amount of CICS request data	接続されている CICS サーバに送信された要求データ量（バイト単位）。 この量には、アプリケーションおよび CICS プロトコル データの両方が含まれます。
Gateway Daemon:Total Requests Processed/Second	処理された API 呼び出しの集約数。
Protocol Handler:SSL protocol handler port number	SSL トラフィックに使用されている TCP ポート。
Protocol Handler:TCP protocol handler port number	非 SSL トラフィックに使用されている TCP ポート。
System Environment:JVM GC count	ガベージ コレクション イベントの数。
System Environment:JVM GC time	JVM がガベージ コレクションに要したミリ秒。
System Environment:JVM heap size after GC	最後のガベージ コレクションの後の JVM ヒープ サイズ。
System Environment:JVM initial heap size	JVM ヒープの初期サイズ。
System Environment:JVM maximum heap size	JVM ヒープの最大サイズ。
Worker Threads:Current number of worker threads	現在作成されているワーカ スレッド数。
Worker Threads:Currently allocated worker threads	現在、接続マネージャによって使用されているワーカ スレッド数。
Worker Threads:Initial number of worker threads	ゲートウェイ デーモンによって作成された設定済みワーカ スレッドの初期数。
Worker Threads:Maximum number of worker threads	ゲートウェイが処理可能な設定済み並列ワーカ スレッドの最大数。
Worker Threads:Number of times worker timeout reached	ゲートウェイ デーモンが、設定されたタイムアウト時間内に、接続マネージャへのワーカ スレッドの割り当てに失敗した回数。
Worker Threads:Peak number of allocated worker threads	接続マネージャ スレッドに同時に割り当てられたワーカ スレッドのピーク数。