CA Application Performance Management

.NET 에이전트 구현 안내서

릴리스 9.5

CA Technologies 제품 참조

이 문서에서는 다음과 같은 CA Technologies 제품과 기능을 참조합니다.

- 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 에 문의

기술 지원팀에 문의

온라인 기술 지원 및 지사 목록, 기본 서비스 시간, 전화 번호에 대해서는 http://www.ca.com/worldwide에서 기술 지원팀에 문의하십시오.

목차

제 1 장: .NET 에이전트 소개

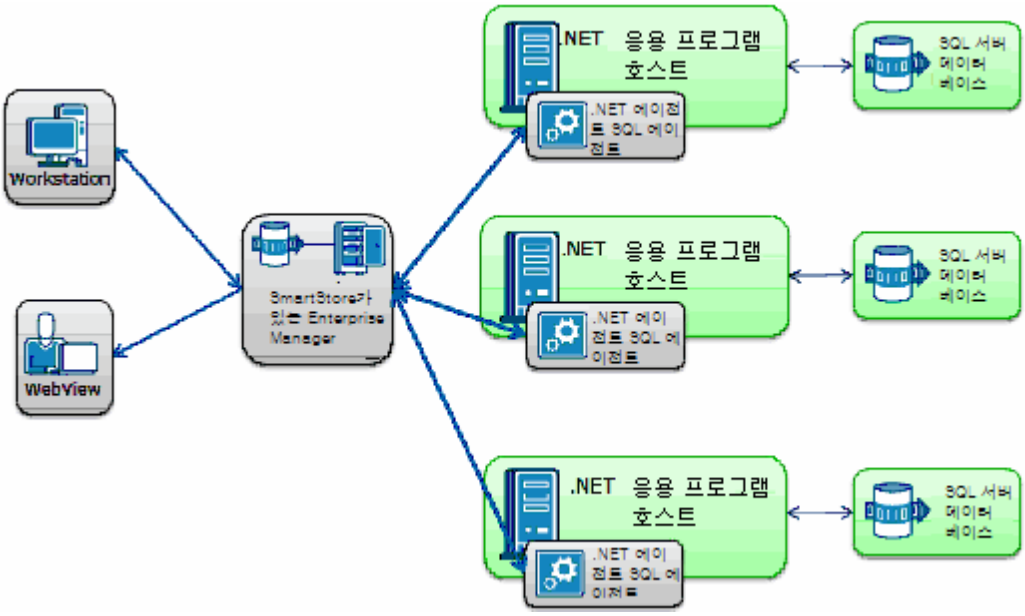
이 섹션은 다음 항목을 포함하고 있습니다.

Introscope 배포에서의 .NET 에이전트 정보

.NET 에이전트는 .NET 응용 프로그램을 위한 응용 프로그램 관리 솔루션입니다. .NET 에이전트는 Microsoft CLR(공용 언어 런타임) 환경에서 실행되는 중요 .NET 응용 프로그램을 모니터링하여 구성 요소 수준의 정보를 제공합니다.

Introscope 배포에서 에이전트는 응용 프로그램 메트릭과 환경 메트릭을 수집하여 Enterprise Manager 에 릴레이합니다. Introscope Agent 에 메트릭을 보고하는 응용 프로그램을 *계측된 응용 프로그램*이라고 합니다.시스템에 .NET 에이전트를 설치하고 구성하면 해당 시스템에서 실행되는 응용 프로그램이 시작 시 자동으로 계측됩니다.

다음 그림에서는 .NET 응용 프로그램이 SQL Server 데이터베이스에 연결되어 있는 간단한 Introscope 배포를 보여 줍니다. 각 응용 프로그램 서버는 .NET 에이전트와 SQL 에이전트를 호스트합니다. 에이전트는 응용 프로그램 동작을 모니터링하고 메트릭을 Enterprise Manager 에 보고합니다. 메트릭은 Workstation 과 WebView 를 통해 볼 수 있습니다. 규모가 더 크고 복잡한 배포 환경에는 에이전트 수가 많고 Enterprise Manager 도 여러 개 사용됩니다.



기본적으로 Microsoft IIS(인터넷 정보 서비스)를 사용하여 배포되고 시스템에서 활성 상태인 ASP.NET 응용 프로그램만 계측됩니다. 독립 실행형 .NET 실행 파일을 계측하거나, 응용 프로그램의 하위 집합만 계측할 수도 있습니다.

추적 프로그램이 수행하는 계측 프로세스는 ProbeBuilder 지시문 파일(PBD)에 정의됩니다. PBD 파일에 들어 있는 지시문은 모니터링할 대상 응용 프로그램 구성 요소를 식별합니다. 추적 프로그램은 이러한 구성 요소가 CLR 에서 실행되는 동안 에이전트가 해당 구성 요소에 대해 수집하는 메트릭을 식별합니다. 그러면 .NET 에이전트는 이 메트릭 정보를 Enterprise Manager 에 보냅니다.

Enterprise Manager 는 여러 에이전트에서 보고한 메트릭을 저장합니다. Introscope Workstation 이나 WebView 클라이언트 응용 프로그램을 사용하면 응용 프로그램 동작을 모니터링하고, 성능 문제의 원인을 검사하고, 문제를 진단할 수 있습니다.

추가 정보:

기본 계측 수정 (페이지 52)

응용 프로그램 환경에서 .NET 에이전트가 작동하는 방법

Introscope 배포에서 모니터링할 대상 응용 프로그램을 실행하는 각 컴퓨터에 .NET 에이전트를 설치합니다. 에이전트를 설치한 후에는 Microsoft IIS(인터넷 정보 서비스)가 .NET 에이전트를 제어합니다. .NET 에이전트는 IIS 가 응용 프로그램에 대한 사용자 요청을 수신한 후에야 활성화됩니다. .aspx, .asmx 같은 응용 프로그램 코드가 실행되면 .NET 에이전트가 시작되고 NativeProfiler 가 해당 코드를 계측합니다.

IIS 가 .NET 에이전트를 제어하는 방법

기본적으로 .NET 에이전트는 IIS 가 관리하고 IIS 작업자 프로세스에서 실행되는 응용 프로그램만 모니터링합니다. 다음 단계는 .NET 응용 프로그램이 시작될 때 IIS 가 .NET 에이전트와 계측 프로세스를 제어하는 방법을 요약하여 보여 줍니다.

1. IIS 가 응용 프로그램에 대한 사용자 요청을 수신합니다.
2. IIS 가 .NET 작업자 프로세스를 시작합니다.
3. 요청된 .NET 응용 프로그램이 시작됩니다.
4. CLR(공용 언어 런타임)에서 NativeProfiler 를 시작합니다.
5. NativeProfiler 가 GAC(전역 어셈블리 캐시)로부터 .NET 에이전트를 로드합니다.
6. .NET 에이전트가 계측에 사용할 PBL 및 PBD 파일을 결정하기 위해 IntroscopeAgent.profile 을 읽습니다.
7. NativeProfiler 는 응용 프로그램 구성 요소에서 적절한 메트릭을 수집하기 위해 PBL 파일과 PBD 파일에 있는 정보를 사용하여 바이트 코드에 프로브를 삽입합니다. 응용 프로그램이 계측됩니다.

8. 계측된 응용 프로그램이 .NET 에이전트에 메트릭을 보고하기 시작합니다.

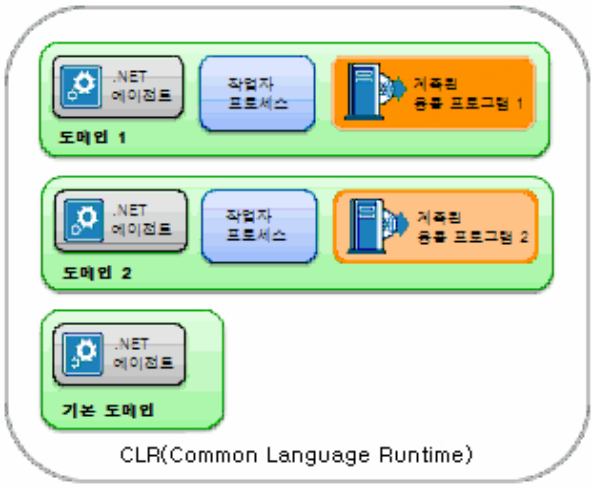
   IIS 작업자 프로세스가 실행되는 동안 에이전트는 계속해서 메트릭을 수집하여 Enterprise Manager 에 보고합니다. 계측된 응용 프로그램에서 사용자 동작이 일정 기간 동안 없으면 IIS 작업자 프로세스가 응용 프로그램 프로세스를 중지합니다. IIS 가 응용 프로그램 프로세스를 중지하면 사용자 동작이 계속될 때까지 .NET 에이전트가 중지됩니다.

참고: ASP.NET 을 사용하지 않는 독립 실행형 응용 프로그램도 .NET 에이전트가 해당 작업을 수행하도록 구성된 경우에 계측될 수 있습니다. 이 프로세스는 독립 실행형 응용 프로그램에 대해서도 유사하지만, Windows 운영 체제에서 독립 실행형 응용 프로그램을 부트스트랩할 수 있기 때문에 1 단계와 2 단계는 생략됩니다.

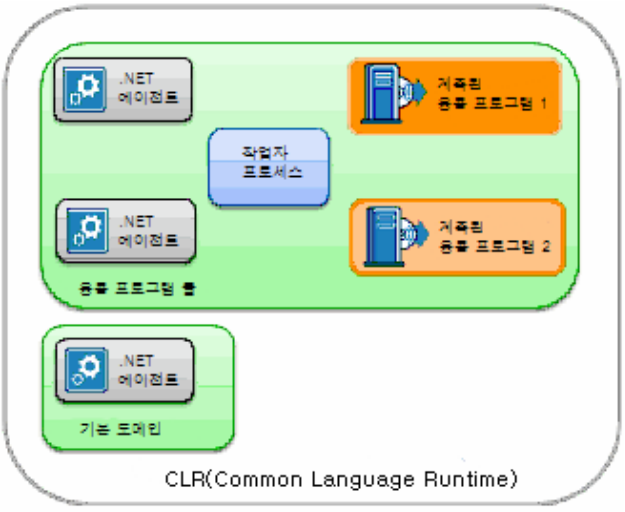
.NET 에이전트 인스턴스화 및 IIS 작업자 프로세스 정보

.NET 에이전트는 모니터링할 관리되는 .NET 응용 프로그램을 호스트하는 각 시스템에 설치합니다. .NET 에이전트를 시작하면 CLR 의 기본 도메인에 대해 에이전트 인스턴스 하나가 만들어지고, CLR 에서 실행되는 각 응용 프로그램에 대해서도 .NET 에이전트 인스턴스가 하나씩 만들어집니다.

다음 그림에서는 .NET 에이전트 하나가 시작된 각 도메인에 IIS 작업자 프로세스가 하나씩 있는 관리되는 ASP.NET 응용 프로그램을 보여 줍니다.

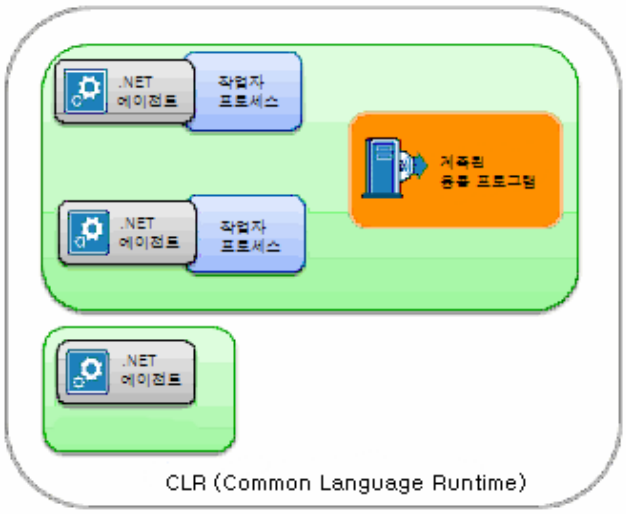


작업자 프로세스 하나를 공유하는 여러 .NET 응용 프로그램이 IIS 응용 프로그램 풀에 그룹화되어 있는 경우 다음 그림과 같이 기본 도메인에 대해 .NET 에이전트 하나가 있고 응용 프로그램 풀에 속해 있는 각 응용 프로그램에 대해서도 .NET 에이전트가 하나씩 있습니다.



일부 회사에서는 확장성 문제 때문에 응용 프로그램 하나에 여러 개의 작업자 프로세스를 할당할 수 있습니다. 따라서 기본 도메인에 대해 .NET 에이전트 인스턴스 하나가 만들어지고 응용 프로그램 관련 작업자 프로세스 각각에 대해서도 .NET 에이전트 인스턴스가 하나씩 만들어집니다.

참고: 다음 그림에 나와 있는 구성이 가장 일반적인 구성입니다.



작업자 프로세스가 여러 개 있고 그에 따라 관리되는 응용 프로그램 하나에 연결되어 있는 .NET 에이전트가 여러 개 있는 경우에는 해당 에이전트를 가상 에이전트로 구성합니다. 이와 같이 구성하면 물리적 .NET 에이전트 여러 개의 메트릭을 집계할 수 있습니다.

참고: 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

기본 도메인의 .NET 에이전트 인스턴스 정보

.NET 에이전트 인스턴스화 (페이지 20)에서 설명한 대로 .NET 에이전트는 CLR 기본 도메인에 대해 항상 만들어집니다. 기본적으로 기본 도메인의 .NET 에이전트는 Enterprise Manager 에 연결되거나 Workstation 또는 WebView 에서 Investigator 트리의 노드로 표시되지 않습니다. 그러나 필요한 경우에는 Enterprise Manager 에 연결하도록 기본 도메인의 .NET 에이전트를 구성할 수 있습니다. 자세한 내용은 기본 도메인 구성 (페이지 216)에서 에이전트 속성을 참조하십시오.

엔터프라이즈에 .NET 에이전트를 배포하는 방법 이해

응용 프로그램과 해당 응용 프로그램이 실행되는 환경에 적합한 .NET 에이전트 구성을 개발하는 것은 반복적인 프로세스입니다.

기본 기능 설치 및 평가

에이전트를 배포하는 첫 번째 단계에서는 기본 에이전트 구성을 설치하고 평가합니다. 기본 에이전트 구성에서는 컴퓨팅 환경과 응용 프로그램의 일반적인 여러 구성 요소에 대한 데이터 수집을 보여 줍니다. 또한 기본 에이전트 구성에서 일부 기능은 사용하도록 설정된 반면 사용 빈도가 적은 다른 기능은 사용하지 않도록 설정되어 있습니다.

이 단계의 목적은 기본적으로 제공되는 데이터 수집의 수준 및 범위를 평가하고 Introscope 와 응용 프로그램 모니터링 방법에 익숙해지는 것입니다. 에이전트에서 기본적으로 제공되는 성능 메트릭에 익숙해진 후에는 필요에 따라 데이터를 수집하도록 에이전트를 사용자 지정할 수 있습니다.

기본 성능을 평가할 때는 에이전트에서 수집하는 메트릭이 많을수록 시스템 리소스가 많이 소비된다는 점에 유의하십시오. 반면 에이전트에서 수집하는 메트릭이 적을수록 잠재적 문제를 파악하기가 어려워집니다. 에이전트 구성을 미세 조정할 때는 데이터 수집 수준과 허용 가능한 성능 수준 간의 균형을 맞추도록 하십시오. 적절한 계측 수준은 일반적으로 에이전트가 배포되는 위치에 따라 달라집니다. 예를 들어 테스트 환경 내에서 모니터링하는 에이전트는 대개 많은 수의 메트릭을 수집하도록 구성됩니다. 그러나 프로덕션 서버의 에이전트는 대개 꼭 필요한 정보만 제공하도록 구성됩니다.

구성 요구 사항 확인

에이전트를 배포하기 전에 데이터 수집 요구 사항을 확인하십시오. 이 정보를 통해 에이전트의 데이터 수집 동작을 수정하고, 대체 에이전트 구성을 사용할 경우의 오버헤드 영향을 평가할 수 있습니다.

Introscope 는 응용 프로그램 수명 주기의 모든 단계에서 사용할 수 있습니다. 예를 들어 개발 단계부터 테스트, 부하 확인, 스테이징 및 프로덕션 단계에 이르는 전 과정에서 Introscope 를 사용할 수 있습니다. 수명 주기의 각 단계마다 모니터링 목적, 환경적 제약 및 서비스 수준 기대치가 서로 다를 수 있습니다. 이러한 차이를 해결하려면 에이전트가 모니터링 대상 환경의 유형에 따라 다르게 동작하도록 구성해야 합니다.

이 단계의 목적은 성능 세부 정보의 가시성과 리소스 오버헤드 사이에서 적절한 균형이 이루어지는 지점을 확인하는 것입니다. 또한 모니터링 대상 환경에 합리적인 비용으로 최적 수준의 가시성을 얻을 수 있어야 합니다.

개발과 같은 프로덕션 전 응용 프로그램 환경에서는 일반적으로 데이터 수집 수준을 높게 구성할수록 응용 프로그램 성능을 세부적으로 파악할 수 있습니다. 프로덕션 또는 고용량 트랜잭션 환경에서는 일반적으로 보고되는 메트릭 수를 줄여 에이전트 오버헤드를 제어합니다. 요구 사항에 따라 에이전트 속성을 구성하여 특정 에이전트 동작을 제어할 수도 있습니다. 예를 들어 수집되는 최대 개수의 메트릭을 추적하고 오래된 메트릭을 제거할 수 있습니다.

현재 환경을 고려하여 적절한 수준의 가시성과 허용 가능한 성능 오버헤드를 확인하고 그에 따라 요구 사항에 맞게 에이전트를 구성할 수 있습니다.

적절한 구성 속성을 사용하여 기준 에이전트 프로필 정의

모니터링할 응용 프로그램 환경의 유형을 확인한 후에는 "후보" 에이전트 구성을 생성할 수 있습니다. 대부분의 에이전트 작업은 에이전트 프로필(IntroscopeAgent.profile) 내의 속성을 사용하여 제어됩니다. 예를 들어 IntroscopeAgent.profile 파일에서는 에이전트가 사용하는 ProbeBuilder 지시문 파일과 ProbeBuilder 목록 파일을 정의합니다. 또한 에이전트 프로필에 나열된 파일은 에이전트가 수집하는 특정 메트릭을 제어합니다. IntroscopeAgent.profile 파일에서는 특정 기능을 사용 또는 사용하지 않도록 설정하는 속성이나 폴링 간격과 같이 작업을 조정하는 속성도 제공합니다.

사용 중인 구성 및 환경에 따라 에이전트 프로필의 속성을 조정하여 각 변경 사항의 영향을 평가할 수 있습니다. 예를 들어 처음에는 ChangeDetector 를 사용하도록 설정되지 않은 기본 에이전트 프로필을 사용하고, 나중에 프로필에서 ChangeDetector 속성을 사용하도록 설정하여 다른 기능을 추가하기 전에 변경 후의 에이전트 성능을 평가할 수 있습니다.

에이전트 성능 오버헤드 평가

에이전트 구성을 평가할 때는 수집된 메트릭을 통해 응용 프로그램의 성능과 가용성을 충분하게 파악할 수 있는지 확인하십시오. 또한 메트릭 수가 너무 많아 운영 환경에 지나친 부하가 발생하지는 않는지 확인하십시오. 에이전트에서는 성능 및 가용성 문제를 파악하고 지역화하는 데 필요한 것 이상의 메트릭을 보고할 수 없습니다.

응용 프로그램의 성능 특성을 이해하면 에이전트 성능을 효과적으로 평가할 수 있습니다. 예를 들어 영향을 확인하기 위해 기본 모니터링을 구현하기 전과 그 후에 응용 프로그램에 대한 부하 테스트를 실행할 수 있습니다.

데이터 수집을 제어된 방식으로 확장하려면 변경 사항을 구현하기 전후에 에이전트 작업 및 오버헤드를 확인하십시오. 예를 들어 각 추가 기능의 성능을 개별적으로 평가할 수 있도록 한 번에 한 응용 프로그램에 대한 모니터링 지원만 추가하십시오.

에이전트 구성 유효성 검사 및 배포

후보 에이전트 구성이 필요한 가시성을 제공하는 것으로 확인된 후에는 해당 환경에 구성을 배포하십시오.

대상 환경에 다음 구성 항목을 설치하면 유효성이 검사된 구성을 배포할 수 있습니다.

- IntroscopeAgent.profile 파일
- 수정되거나 사용자 지정된 PBD 파일

.NET 에이전트 배포

.NET 에이전트를 배포하려면 다음 단계를 수행하십시오.

1. .NET 에이전트를 설치합니다 (페이지 27).
2. .NET 에이전트를 구성합니다 (페이지 41).
3. .NET 에이전트 속성을 구성합니다 (페이지 61).

추가 정보:

.NET 에이전트의 디렉터리 구조 이해 (페이지 38)

제 2 장: .NET 에이전트 설치

이 단원에 나와 있는 지침은 .NET 에이전트를 Microsoft IIS(인터넷 정보 서비스) 서버에 설치하고 Enterprise Manager 와 통신하도록 구성하는 방법을 설명합니다.

참고: Enterprise Manager, Workstation 및 WebView 구성 요소를 설치하는 방법에 대한 자세한 내용은 *CA APM 설치 및 업그레이드 안내서*를 참조하십시오.

이 섹션은 다음 항목을 포함하고 있습니다.

.NET 에이전트를 설치하기 전에

Introscope 나 .NET 에이전트를 처음 사용하는 경우에는 배포 프로세스를 검토해 보십시오.

다음 단계를 따르십시오.

1. 다음과 같은 구성 요소의 지원되는 버전을 설치했는지 확인합니다.
   - .NET framework

     참고: 기본적으로 에이전트는 Framework 하나만 모니터링합니다. 에이전트 속성을 구성하면 .NET Framework 4 및 In-Process Side-by-Side 실행을 사용하여 모니터링할 대상을 제어할 수 있습니다.
   - Windows 운영 체제 및 IIS
   - Microsoft SQL Server 또는 Oracle 데이터베이스
   - ODP.NET 드라이버

2. 모니터링할 대상 응용 프로그램이 IIS 작업자 프로세스를 사용하여 실행되는지 확인합니다.

   웹 응용 프로그램이 작업자 프로세스를 사용하는지 확인하려면 응용 프로그램에서 페이지를 로드한 후 작업 관리자를 열어 다음 파일을 찾습니다.

   - IIS 의 aspnet_wp.exe
   - IIS 의 w3wp.exe

   작업자 프로세스가 표시되지 않으면 관리되는 구성 요소 처리를 위해 IIS 를 사용하도록 설정하십시오. 사용 중인 .NET 버전의 .NET Framework 디렉터리로 이동하여 다음 명령을 실행하십시오.

   ```
   aspnet_regiis.exe –i
   ```

   예:

   ```
   C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_regiis.exe -i
   ```

3. 에이전트 파일을 설치하기 위한 디스크 공간이 충분한지 확인합니다. CA Technologies 에서는 설치 관리자 실행 파일 크기의 3 배에 해당하는 디스크 공간을 확보할 것을 권장합니다.

4. 새 .NET 에이전트를 설치할 컴퓨터에서 이전 버전의 .NET 에이전트를 모두 제거했는지 확인합니다.

   명령 창을 열고 "*set*"을 입력하여 현재 정의되어 있는 환경 변수를 나열하는 방법으로 .NET 에이전트가 있는지 여부를 확인할 수 있습니다. com.wily.introscope.agentProfile=*<path>*가 목록에 있으면 계속하기 전에 에이전트를 제거해야 합니다.

5. .NET 에이전트를 설치할 컴퓨터에 다른 모니터링 에이전트나 CLR 프로파일러가 설치되어 있지 않은지 확인합니다.

6. Introscope Enterprise Manager 와 Workstation 이 설치되었는지 확인합니다. Enterprise Manager 연결을 구성하는 데 필요한 호스트 이름과 포트 번호를 기록해 두어야 합니다.

   ping 또는 *텔넷*을 사용하여 에이전트와 Enterprise Manager 간의 연결을 확인할 수 있습니다.

참고: 지원되는 .NET 에이전트 버전과 설치 요구 사항은 *Compatibility Guide*(호환성 안내서)를 참조하십시오.

추가 정보:

엔터프라이즈에 .NET 에이전트를 배포하는 방법 이해 (페이지 22)
에이전트 속성 구성 (페이지 61)
.NET 에이전트 제거 (페이지 57)

.NET 에이전트 설치 방법 선택

.NET 에이전트는 다음 방법 중 하나로 설치할 수 있습니다.

- GUI(그래픽 사용자 인터페이스) 설치 관리자를 사용하여 대화식으로 설치 (페이지 30)
- 대화식 확인 메시지 없이 자동으로 설치 (페이지 32)
- 설치 아카이브를 사용하여 수동으로 설치 (페이지 36)

컴퓨터에 파일을 로컬로 설치하려는 경우 대개 대화식 설치 관리자를 사용합니다. 컴퓨터에 원격으로 파일을 설치하거나 사전 구성된 설치 지침 집합을 배포하려는 경우에는 명령줄 매개 변수를 사용하여 스크립트에서 설치 관리자를 실행할 수 있습니다. 설치 관리자를 대화식이나 자동 방식으로 실행하지 않으려는 경우에는 설치 아카이브를 사용하여 에이전트를 수동으로 설치하고 구성할 수 있습니다.

중요! 선택한 방법에 관계없이, "서비스로서 로그온" 권한을 가지려면 SharePoint 도메인 사용자 계정이 필요합니다.

추가 정보:

.NET 에이전트가 제대로 설치되었는지 여부 확인 (페이지 157)

대화식으로 .NET 에이전트 설치

.NET 에이전트 설치 관리자 프로그램을 시작한 후 표시되는 확인 메시지에 응답하여 .NET 에이전트를 대화식으로 설치할 수 있습니다.

다음 단계를 따르십시오.

1. 32 비트 또는 64 비트 운영 체제용 에이전트 설치 관리자 실행 파일을 두 번 클릭합니다. 예를 들어 다음을 두 번 클릭하십시오.
   `introscope<version>windowsAgent_dotNET_32.exe` 파일

   "시작" 페이지가 나타납니다. "Next"(다음)를 클릭하여 설치를 시작합니다.

   참고: .exe 또는 .msi 실행 파일 중 하나를 사용하여 설치할 수 있습니다. .msi 실행 파일은 그룹 정책 또는 예약된 작업 같이 다른 자동화된 방법으로 소프트웨어를 제공하는 데 사용됩니다. .exe 실행 파일은 그 외에 추가적인 기능을 제공합니다. 예를 들어 .exe 실행 파일을 사용하면 관리자로 로그온하지 않은 상태에서 마우스 오른쪽 단추를 클릭하여 설치 관리자를 관리자로 실행할 수 있습니다.

2. "다음"을 클릭하여 기본 설치 디렉터리를 사용하거나, "변경"을 클릭하여 다른 디렉터리를 선택한 후 "다음"을 클릭합니다.

   .NET 에이전트의 기본 설치 디렉터리는 `C:\Program Files\CA APM\Introscope<version>`입니다.

   설치 관리자는 루트 설치 디렉터리 내에 *<Agent_Home>* 디렉터리인 *wily* 디렉터리를 만듭니다.

3. 에이전트가 메트릭을 보고할 Enterprise Manager 의 호스트 이름과 포트 번호를 입력하고 "다음"을 클릭합니다.

   참고: 에이전트의 기본 Enterprise Manager 는 *localhost* 로, Enterprise Manager 와 에이전트를 컴퓨터 하나에 함께 설치하는 랩 환경에 적합합니다. 일반적인 프로덕션 환경에서는 원격 Enterprise Manager 와의 연결을 구성하십시오. 기본 Enterprise Manager 포트는 *5001* 입니다.

4. 사용할 모니터링 옵션을 선택하고 "다음"을 클릭합니다.

   - ChangeDetector 는 응용 프로그램 환경의 변경 내용을 추적하여 Workstation 에 이를 보고합니다. ChangeDetector 를 사용하도록 설정할 경우에는 ChangeDetector 에이전트의 ID 를 입력해야 합니다. ChangeDetector 를 설정하고 사용하는 데 대한 자세한 내용은 *CA APM ChangeDetector 사용자 안내서*를 참조하십시오.

- CA APM for SOA 는 클라이언트 측 및 서버 측 웹 서비스에 대한 확장된 모니터링 기능을 제공합니다. 이 옵션은 기본적으로 선택되어 있습니다.

  참고: .NET 에이전트는 기본적으로 웹 서비스 모니터링 기능을 제공합니다. CA APM for SOA 를 선택하면 향상된 트랜잭션 추적, WCF 서비스 모니터링 및 추가적인 종속성 메트릭 같은 기능이 추가로 제공됩니다. CA APM for SOA 를 사용하도록 설정했는지 여부에 따라 설치 후 .pbd 파일의 내용이 달라집니다. 자세한 내용은 *CA APM for SOA 구현 안내서*를 참조하십시오.

- CA APM for Microsoft SharePoint 와 CA APM Standalone Agent for Microsoft SharePoint(SPMonitor)를 사용하면 Microsoft SharePoint 웹 응용 프로그램과 서비스를 모니터링할 수 있습니다. 이 옵션은 Windows Server 2003 또는 2008 에서만 사용할 수 있습니다. 자세한 내용은 *CA APM for Microsoft SharePoint 안내서*를 참조하십시오.

모니터링 옵션을 설정하지 않은 경우 나중에 모니터링 옵션을 수동으로 설정할 수 있습니다.

5. 추가적인 에이전트 서비스를 자동으로 시작할지 아니면 수동으로 시작할지 선택합니다.

   CA APM Standalone Agent for Microsoft SharePoint(SPMonitor)를 선택한 경우, 서비스를 실행하려면 SharePoint 도메인 사용자 계정과 암호를 지정해야 합니다.

   예: *<domain>\<username>*

6. "Install"(설치)을 클릭합니다.

   설치 관리자가 .NET 에이전트 루트 설치 디렉터리를 만들고 에이전트가 사용하는 파일을 설치합니다.

7. 설치가 완료되면 "마침"을 클릭합니다.

8. 다음 중 *하나*를 수행하여 에이전트가 성공적으로 설치되었는지 확인합니다.

   - .exe 설치 관리자:

     IntroscopeDotNetAgentInstall*.exe.log 파일이 설치 관리자 실행 파일과 같은 디렉터리에 있는지 확인합니다.

   - .msi 설치 관리자:

     MSI*.LOG 파일이 %temp% 폴더 안에 생성되었는지 확인합니다. 이 로그 파일은 Windows Installer 4.0 이상에서만 사용할 수 있습니다.

9. IIS Admin Service 를 재시작하여 설치를 완료합니다.

참고: .NET 에이전트를 설치하면 *com.wily.introscope.agentProfile, Cor_Enable_Profiling*, *COR_PROFILER* 같은 시스템 환경 변수가 운영 체제에 추가됩니다. 시스템 환경 변수가 변경되었음을 IIS 에 알리려면 IIS 를 재시작하거나 로컬 컴퓨터를 재부팅해야 합니다.

추가 정보:

.NET 에이전트의 디렉터리 구조 이해 (페이지 38)
Enterprise Manager 및 에이전트 연결을 구성하는 방법 (페이지 41)

자동으로 .NET 에이전트 설치

설치 관리자를 자동 모드로 실행하려면 명령줄에서 .NET 에이전트 설치 관리자를 호출하고 설치 지침을 명령줄 매개 변수로 지정합니다. 그러면 진행 상태에 대한 정보가 표시되지 않고 백그라운드에서 자동으로 설치가 실행됩니다. 이 설치 방법은 사용자 상호 작용 없이 에이전트를 설치할 수 있기 때문에 .NET 에이전트를 원격 컴퓨터에 설치하는 데 가장 일반적으로 사용됩니다. 동일한 구성으로 에이전트를 여러 개 설치할 수도 있습니다. 설치 시 명령에 *.exe* 또는 *.msi* 실행 파일을 지정할 수 있습니다.

다음 단계를 따르십시오.

1. 명령 프롬프트 창을 엽니다.
2. 적절한 명령줄 매개 변수와 함께 .exe 설치 관리자 실행 파일이나 .msi 설치 관리자 실행 파일을 호출합니다. 예를 들어 64 비트 시스템에 설치하려면 다음과 같은 명령줄 매개 변수를 사용할 수 있습니다.

   .msi 설치 관리자:

   ```
   IntroscopeDotNetAgentInstall64_9.1.0.0.msi /qn
   [INSTALLDIR="<install_dir>"] [EMHOST=myhost] [EMPORT=5001] [ENABLECD=1
   CDAGENTID=<change_dectector_agent_id>] [ENABLESOA=1] [ENABLESPP=1]
   [INSTALLSPMONITOR=1 IS_NET_API_LOGON_USERNAME=<SP Monitor Domain username>
   IS_NET_API_LOGON_PASSWORD=<SP Monitor Domain password>]
   ```

.exe 설치 관리자:

```
IntroscopeDotNetAgentInstall64.exe /s /v"/qn [INSTALLDIR="<install_dir>"]
[EMHOST=myhost] [EMPORT=5001] [ENABLECD=1
CDAGENTID=<change_dectector_agent_id>] [ENABLESOA=1] [ENABLESPP=1]
[INSTALLSPMONITOR=1] [INSTALLSPMONITOR=1 IS_NET_API_LOGON_USERNAME=<SP
Monitor Domain username> IS_NET_API_LOGON_PASSWORD=<SP Monitor Domain
password>]"
```

참고: */s /v /qn* 옵션을 사용하여 설치 관리자를 호출하면 자동 설치가 실행됩니다. 그 외에 다른 매개 변수도 선택적으로 지정할 수 있습니다. 명령줄에 매개 변수를 지정하지 않으면 해당 매개 변수의 기본 설정이 사용됩니다.

다음과 같은 선택적 매개 변수는 설치 관리자를 대화식으로 실행할 때 선택하는 옵션과 동일합니다.

INSTALLDIR="*<root_installation_directory>*"

.NET 에이전트를 설치할 디렉터리를 지정합니다. 경로에 백슬래시가 포함되어 있으면 경로 앞에 백슬래시를 추가합니다. 예: `\"C:\IntroscopeDir"`

기본 루트 설치 디렉터리는 `C:\Program Files\CA APM\Introscope<version>`입니다.

기본 디렉터리를 변경하려면 이 속성을 수정하십시오.

EMHOST=*<host_name>*

에이전트가 메트릭을 보고하는 Enterprise Manager 의 호스트 이름을 지정합니다. 에이전트의 기본 호스트 이름은 localhost 입니다.

EMPORT=*<port_number>*

에이전트가 메트릭을 보고하는 Enterprise Manager 의 수신 포트 번호를 지정합니다. 기본 포트는 5001 입니다.

ENABLECD=0

ChangeDetector 를 사용할지 여부를 지정합니다. 이 매개 변수의 기본값은 0 으로, ChangeDetector 를 사용하지 않음을 지정합니다.

ChangeDetector 를 사용하려면 *ENABLECD* 매개 변수를 1 로 설정합니다.

ChangeDetector 를 사용하지 않으면 나중에 사용할 수 있도록 관련 파일이 *<Agent_Home>*\examples 디렉터리에 저장됩니다.

CDAGENTID=<*agent_name*>

ChangeDetector 에이전트의 이름을 지정합니다. 이 매개 변수는 *ENABLECD* 를 1 로 설정한 경우에만 명령줄에 포함해야 합니다. ChangeDetector 에이전트의 기본 에이전트 이름은 *SampleApplicationName* 입니다.

ENABLESOA=0

CA APM for SOA 를 사용할지 여부를 지정합니다. 이 매개 변수의 기본값은 0 으로, CA APM for SOA 를 사용하지 않음을 지정합니다.

CA APM for SOA 를 사용하려면 *ENABLESOA* 매개 변수를 1 로 설정합니다.

CA APM for SOA 를 사용하지 않으면 나중에 사용할 수 있도록 관련 파일이 <*Agent_Home*>\examples 디렉터리에 저장됩니다.

ENABLESPP=0

CA APM for Microsoft SharePoint Portal 을 사용할지 여부를 지정합니다. 이 매개 변수의 기본값은 0 으로, CA APM for Microsoft SharePoint 를 사용하지 않음을 지정합니다.

CA APM for Microsoft SharePoint Portal 을 사용하려면 *ENABLESPP* 매개 변수를 1 로 설정합니다. 운영 체제가 Windows Server 2003 또는 Windows Server 2008 인 경우에만 Microsoft SharePoint 에 대한 모니터링을 사용하십시오.

CA APM for Microsoft SharePoint Portal 을 사용하지 않으면 나중에 사용할 수 있도록 관련 파일이 <*Agent_Home*>\examples 디렉터리에 저장됩니다.

INSTALLSPMONITOR=0

CA APM Standalone Agent for Microsoft SharePoint Portal 을 설치할지 여부를 지정합니다. 이 매개 변수의 기본값은 0 으로, CA APM Standalone Agent for Microsoft SharePoint 를 설치하지 않음을 지정합니다.

CA APM Standalone Agent for Microsoft SharePoint 를 설치하려면 *INSTALLSPMONITOR* 매개 변수를 1 로 설정합니다.

CA APM Standalone Agent for Microsoft SharePoint 를 설치하지 않을 경우 나중에 CA APM Standalone Agent 설치 관리자를 실행하여 설치할 수 있습니다.

예를 들어 기본 설정을 변경하려면 64 비트 시스템에서 명령줄에 다음과 유사하게 지정하십시오.

```
IntroscopeDotNetAgentInstall64.exe /s /v"/qn INSTALLDIR=\"C:\IntroscopeAgent\"
EMHOST=dell-M65.org EMPORT=5001 ENABLECD=1 CDAGENTID=myCDAgent ENABLESOA=1
ENABLESPP=1 INSTALLSPMONITOR=1 IS_NET_API_LOGON_USERNAME=apm-domain\apmuser
IS_NET_API_LOGON_PASSWORD=apmPassword"
```

IS_NET_API_LOGON_USERNAME=*<domain><username>***

Microsoft SharePoint 모니터 도메인 사용자 이름을 지정합니다. 이 매개 변수는 *INSTALLMONITOR* 매개 변수를 0 으로 설정한 경우에 지정합니다. 이 매개 변수의 기본값은 null 입니다.

IS_NET_API_LOGON_PASSWORD=*<password>*

Microsoft SharePoint 모니터의 암호를 지정합니다. 이 매개 변수의 기본값은 null 입니다.

3. 에이전트가 제대로 설치되었는지 확인하려면 다음과 같이 로그 파일을 확인합니다.
   - *.exe* 설치 관리자:

     설치 관리자 실행 파일과 같은 디렉터리에 있는 IntroscopeDotNetAgentInstall*.exe.log 파일을 봅니다.
   - *.msi* 설치 관리자:

     %temp% 폴더의 MSI*.LOG 파일을 봅니다. 이 로그 파일은 Windows Installer 4.0 이상에서만 사용할 수 있습니다.
4. (선택 사항) 예약된 작업으로 적절한 설정을 사용하여 설치 관리자를 호출하거나, 추가 컴퓨터에서 원격으로 실행되는 스크립트를 만듭니다.
5. IIS Admin Service 를 재시작하여 설치를 완료합니다.

참고: .NET 에이전트를 설치하면 *com.wily.introscope.agentProfile, Cor_Enable_Profiling* 및 *COR_PROFILER* 같은 시스템 환경 변수가 운영 체제에 추가됩니다. 시스템 환경 변수가 변경되었음을 IIS 에 알리려면 IIS 를 재시작하거나 로컬 컴퓨터를 재부팅해야 합니다.

설치 아카이브를 사용하여 수동으로 .NET 에이전트 설치

설치 관리자를 대화식으로 실행하거나 응답 파일을 구성하지 않고 시스템에 에이전트 파일을 포함할 수 있습니다. 응용 프로그램 서버별 아카이브를 사용하여 에이전트를 설치할 수 있습니다.

설치 아카이브에는 에이전트 설치 관리자에 사용되는 것과 동일한 파일이 들어 있습니다. 아카이브를 컴퓨터에 복사한 후 내용을 추출하고 에이전트를 구성합니다. 이러한 파일을 사용하여 여러 에이전트를 배치 작업으로 배포하거나 해당 파일을 기본 에이전트 파일 집합의 아카이브로 보관하십시오.

다음 단계를 따르십시오.

1. CA Support의 CA APM 소프트웨어 다운로드 영역에서 다음 설치 아카이브 중 현재 운영 체제에 맞는 설치 아카이브 *하나*를 다운로드합니다.
   - DotNetAgentFiles-NoInstaller.x64.<APM_version_number>.zip
   - DotNetAgentFiles-NoInstaller.x86.<APM_version_number>.zip

   예:

   ```
   DotNetAgentFiles-NoInstaller.x64.9.1.0.0.zip
   ```
2. 원하는 디렉터리에 설치 아카이브의 내용을 추출합니다.
3. .NET 에이전트의 *<Agent_Home>*\wily 디렉터리를 설정합니다.

   참고: .NET 에이전트 설치 관리자 프로그램을 실행할 경우 기본 루트 설치 디렉터리는 C:\Program Files\CA APM\Introscope*<version>*입니다. 설치 관리자는 루트 설치 디렉터리에서 *<Agent_Home>* 디렉터리인 *wily* 디렉터리를 생성합니다.
4. 다음과 같이 wily.Agent.dll 파일을 GAC(전역 어셈블리 캐시)에 등록합니다.
   a. 관리자 권한으로 명령 프롬프트를 엽니다.
   b. *wily\bin* 디렉터리로 이동합니다. 예:
      ```
      explorer <Agent_Home>\wily\bin
      ```
   c. *assembly* 디렉터리로 이동합니다. 예:
      ```
      explorer C:\WINDOWS\assembly
      ```
   d. *<Agent_Home>*\wily\bin 디렉터리에서 assembly 디렉터리로 wily.Agent.dll 을 복사합니다.

5. 다음과 같이 wily.NativeProfiler 파일을 등록합니다.
   a. 명령줄에서 wily\bin 디렉터리로 이동하여 다음 실행 파일을 호출합니다.
      ```
      C:\WINDOWS\SysWOW64\regsvr32.exe
      <Agent_Home>\wily\bin\x86\wily.NativeProfiler.dll
      ```
6. IntroscopeAgent.profile 파일을 텍스트 편집기에서 열고 Enterprise Manager 연결을 다음과 같이 구성합니다.
   a. *introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT* 행을 찾은 다음 에이전트에 대한 Enterprise Manager 의 호스트 이름을 지정합니다. 예:
      ```
      introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=sfcollect01
      ```
   b. Enterprise Manager 와 통신하는 데 필요한 다른 속성을 적절하게 설정합니다.
7. 명령줄에서 PerfMon Collector Service 를 다음과 같이 시작합니다.
   ```
   sc create PerfMonCollectorAgent binPath=
   "<Agent_home>\bin\PerfMonCollectorAgent.exe start= auto DisplayName= "CA APM
   PerfMon Collector Service"
   ```
8. 명령줄에서 *wilypermissions* 유틸리티를 실행하여 사용자가 *<Agent_Home>* 디렉터리에 액세스할 수 있는 권한을 가지고 있는지 확인합니다. 예:
   ```
   wilypermissions.exe c:\WilyAgent9065\wily w3wp.exe
   ```
9. .NET 에이전트 환경 변수를 다음과 같이 구성합니다.
   - com.wily.introscope.agentProfile=*<Agent_Home>*\wily\IntroscopeAgent.profile
   - Cor_Enable_Profiling=0x1
   - COR_PROFILER={5F048FC6-251C-4684-8CCA-76047B02AC98}
10. IIS Admin Service 를 다시 시작하여 설치를 완료합니다.

추가 정보:

에이전트 디렉터리에 대한 기본 사용자 권한 수정 (페이지 40)
Enterprise Manager 및 에이전트 연결을 구성하는 방법 (페이지 41)

중국어 또는 일본어로 .NET 에이전트 설치

.msi 설치 관리자 실행 파일을 실행할 때 언어 값을 지정할 수 있습니다.

참고: 이 정보는 32 비트 및 64 비트 Windows 에만 해당됩니다.

다음 단계를 따르십시오.

1. 명령 프롬프트 창을 엽니다.
2. 다음과 같은 언어 값을 사용하여 .msi 설치 관리자 실행 파일을 호출합니다.
   - 일본어:
     ```
     IntroscopeDotNetAgentInstall64_<release>.msi TRANSFORMS=1041.mst
     ProductLanguage=1041
     ```
   - 중국어:
     ```
     IntroscopeDotNetAgentInstall64_<release>.msi TRANSFORMS=2052.mst
     ProductLanguage=2052
     ```
3. 응용 프로그램 서버를 다시 시작하여 설치를 완료합니다.

   설치 관리자가 .NET 에이전트 루트 설치 디렉터리를 생성하고 에이전트 파일을 설치합니다. .MSI*.LOG 파일이 %temp% 폴더에 생성됩니다.

.NET 에이전트의 디렉터리 구조 이해

.NET 에이전트를 설치하면 루트 설치 디렉터리에 다음과 같은 *<Agent_Home>* 디렉터리 구조가 생성됩니다.

wily

에이전트 작업, 메트릭 데이터 수집 및 계측 프로세스를 제어하는 IntroscopeAgent.profile, ProbeBuilder 지시문 파일(.pbd)과 ProbeBuilder 목록 파일(.pbl)이 포함되어 있습니다. IntroscopeAgent.profile 에 정의되어 있는 속성 중에서 에이전트에서 배포하고 참조하는 속성은 설치 시 선택한 옵션에 따라 다릅니다.

<Agent_Home> 디렉터리 안에 있는 하위 디렉터리는 .NET 에이전트의 다양한 기능을 사용하는 데 필요한 라이브러리와 확장 파일을 제공합니다.

bin

핵심 .NET 에이전트 라이브러리를 포함합니다.

dynamic

동적 계측에 필요한 PBD 파일을 포함합니다.

예

CA APM for SOA 와 같은 선택적 에이전트 확장에 대한 폴더 및 파일이 포함되어 있습니다. 설치 시 확장 기능을 사용하도록 설정하지 않은 경우 이 디렉터리에 있는 파일을 사용하여 나중에 확장을 구성하고 사용하도록 설정할 수 있습니다.

ext

사용하도록 설정된 에이전트 확장 또는 기능에 대한 파일이 포함되어 있습니다. 예를 들어 이 디렉터리에는 응용 프로그램 심사 맵 및 LeakHunter 에 대한 파일이 포함되어 있습니다.

hotdeploy

이 디렉터리는 기본적으로 비어 있습니다. PBD 파일을 이 디렉터리에 두면 IntroscopeAgent.profile 을 편집하거나 응용 프로그램을 재시작할 필요 없이 새 지시문을 배포할 수 있습니다. 그러나 이 디렉터리에 파일을 둘 때는 주의해야 합니다. 사용자 지정 PBD 가 잘못된 구문을 포함하거나 너무 많은 메트릭을 포함하면 계측이 실패하거나 응용 프로그램 성능이 저하될 수 있습니다.

로그

.NET 에이전트 로그 파일을 저장합니다.

readme

이 디렉터리는 설치 시 확장 기능을 사용하도록 설정한 경우에만 생성됩니다. 디렉터리가 있으면, 사용하도록 설정된 확장 기능에 대한 추가 정보가 포함됩니다.

tools

네트워크 인터페이스 통합 유틸리티와 Tagscript 도구 파일인 TagScript.jar, TagScript.bat 및 TagScript.sh 명령줄 스크립트를 포함합니다.

version

선택적 에이전트 확장을 사용하는지 여부에 관계없이 확장에 대한 버전 정보를 포함합니다.

에이전트 디렉터리에 대한 사용자 권한

기본적으로 설치 관리자 프로그램은 <Agent_Home> 디렉터리에서 다음에 대해 모든 사용자에게 읽기 액세스 권한을 부여합니다.

- *<Agent_Home>*\wily 에 대한 읽기 권한
- *<Agent_Home>*\bin 및 *<Agent_Home>*\ext 에 대한 읽기 및 실행 권한
- *<Agent_Home>*\logs 및 *<Agent_Home>*\dynamic 에 대한 읽기 및 쓰기 권한

이 디렉터리에 대한 액세스를 제한하려면 IIS 작업자 프로세스를 실행하는 사용자 계정만 *<Agent_Home>* 디렉터리에 액세스할 수 있도록 기본 권한을 수정할 수 있습니다.

추가 정보:

에이전트 디렉터리에 대한 사용자 권한 확인 (페이지 50)

에이전트 디렉터리에 대한 기본 사용자 권한 수정

명령줄에서 *wilypermissions* 유틸리티를 실행하여 *<Agent_Home>* 디렉터리에 대한 기본 사용자 권한을 수정할 수 있습니다.

사용자를 지정하지 않으면 유틸리티에서 응용 프로그램의 기본 IIS 사용자를 자동으로 확인하여 해당 사용자에게 권한을 부여합니다.

다음 단계를 따르십시오.

1. *<Agent_Home>* 디렉터리로 이동합니다.
2. 명령줄에서 다음과 같이 *wilypermissions.exe* 를 실행합니다.
   `wilypermissions.exe <Agent_Home> [process name]`

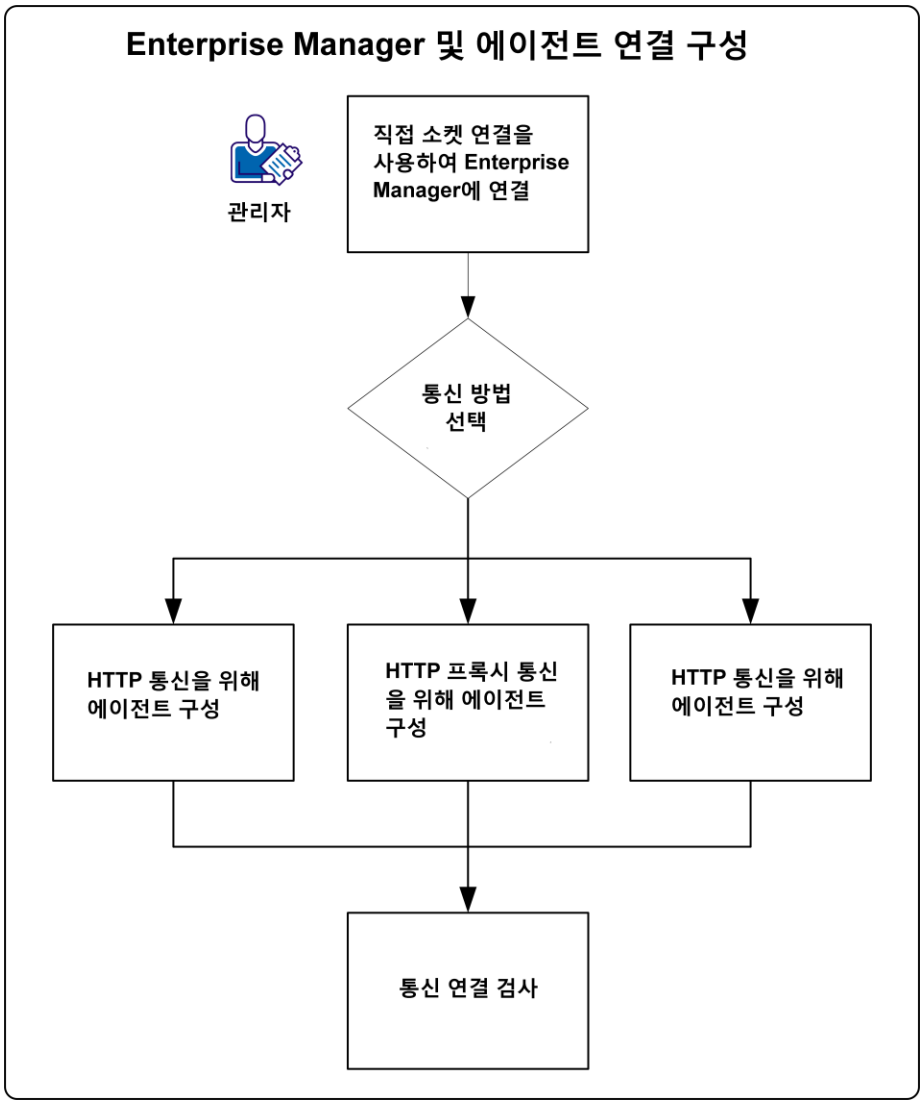
   이 유틸리티는 실행 시 사용자가 성능 모니터 카운터에 액세스할 수 있도록 *<Agent_Home>* 디렉터리에 대한 권한을 사용자에게 부여합니다.

   참고: 프로세스 이름에 파일 확장명을 포함해야 합니다. 프로세스 이름을 지정하지 않으면 IIS 작업자 프로세스 이름이 기본적으로 사용됩니다.

Enterprise Manager 및 에이전트 연결을 구성하는 방법

CA APM 배포에서 에이전트는 웹 응용 프로그램 메트릭과 환경 메트릭을 수집하여 Enterprise Manager 에 릴레이합니다. 에이전트는 메트릭을 보고하기 위해 Enterprise Manager 에 연결해야 합니다. 기본 CA APM 통신 설정을 사용할 경우 에이전트는 로컬 Enterprise Manager 에 연결할 수 있습니다. 관리자는 Enterprise Manager 및 에이전트가 서로 다른 시스템에 있을 때도 연결할 수 있도록 기본 설정을 수정할 수 있습니다. HTTP, HTTP 프록시 또는 HTTPS 의 통신 방법을 사용하도록 설정을 구성할 수 있습니다.

다음 그래픽에서는 관리자가 Enterprise Manager 및 에이전트 연결을 구성하는 방법을 설명합니다.



Enterprise Manager 및 에이전트 연결을 구성하려면 다음 단계를 수행하십시오.

1. 직접 소켓 연결을 사용하여 Enterprise Manager 에 연결합니다. (페이지 42)
2. 통신 방법을 선택합니다. (페이지 43)
   - HTTP 통신에 대한 에이전트 구성 (페이지 43)
   - HTTP 프록시 통신에 대한 에이전트 구성 (페이지 44)
   - HTTPS 통신에 대한 에이전트 구성 (페이지 45)
3. 통신 연결을 확인합니다 (페이지 48).

직접 소켓 연결을 사용하여 Enterprise Manager 에 연결

에이전트는 직접 소켓 연결을 통해 Enterprise Manager 에 연결할 수 있습니다. 가능하면 Enterprise Manager 에 대해 직접 소켓 연결을 사용하는 것이 좋습니다. Enterprise Manager 에 연결할 때 직접 소켓 연결을 사용하도록 통신 속성을 구성할 수 있습니다.

다음 단계를 따르십시오.

1. IntroscopeAgent.profile 파일을 텍스트 편집기에서 엽니다.
2. introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT 속성을 찾은 후 에이전트가 기본적으로 연결할 Enterprise Manager 의 호스트 이름을 지정합니다. 예:
   `introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=sfcollect01`
3. introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT 속성을 Enterprise Manager 수신 포트로 설정합니다. 예:
   `introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=5001`
4. introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT 속성을 Enterprise Manager 에 연결하는 데 사용되는 소켓 팩터리로 설정합니다. 예:
   `introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory`

   참고: 대부분의 경우에는 이 속성의 기본 설정을 사용할 수 있습니다. 다른 소켓 팩터리를 사용하려면 이 속성을 수정하십시오.

5. IntroscopeAgent.profile 파일을 저장하고 닫습니다.
6. IIS Admin Service 를 다시 시작합니다.

   구성이 완료되었습니다.

통신 방법 선택

에이전트를 설치한 후에 다음 통신 방법 중 하나를 사용하도록 기본 에이전트 설정을 수정할 수 있습니다.

- HTTP(Hypertext Transport Protocol)
- HTTP 프록시
- HTTPS(Hypertext Transfer Protocol Secure)

Enterprise Manager 에 대한 에이전트 연결을 구성하려면 구성 방법을 선택하고 다음 태스크 중 해당되는 것을 수행하십시오.

- HTTP 통신에 대한 에이전트 구성 (페이지 43)
- HTTP 프록시 통신에 대한 에이전트 구성 (페이지 44)
- HTTPS 통신에 대한 에이전트 구성 (페이지 45)

HTTP 통신에 대한 에이전트 구성

HTTP 터널링을 사용하면 하나의 네트워크에서 다른 네트워크의 연결을 사용하여 데이터를 안전하게 전송할 수 있습니다. 통신이 HTTP 트래픽만 허용하는 방화벽을 통과할 수 있도록 에이전트 연결을 구성할 수 있습니다.

참고: Enterprise Manager 에서는 기본적으로 HTTP 터널링이 사용되도록 설정됩니다.

다음 단계를 따르십시오.

1. <*Agent_Home*>/wily 디렉터리로 이동합니다.
2. 텍스트 편집기에서 IntroscopeAgent.profile 을 열고 다음 속성을 설정합니다.

   `introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT= Enterprise Manager` *호스트 이름 또는 IP 주소*

3. Enterprise Manager 에 포함된 웹 서버의 HTTP 수신 포트로 다음 속성을 설정합니다. 예:

   ```
   introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=8081
   ```

   Enterprise Manager 의 <EM_Home>/config/IntroscopeEnterpriseManager.properties 파일에 있는 introscope.enterprisemanager.webserver.port 속성 값과 이 값을 비교합니다. 기본적으로 이 포트는 8081 입니다.

4. 다음 속성을 HTTP 터널링 소켓 팩터리로 설정합니다. 예:

   ```
   introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.HttpTunnelingSocketFactory
   ```

5. 파일을 저장하고 닫습니다.
6. 응용 프로그램을 다시 시작합니다.

   Enterprise Manager 웹 서버 포트에 대한 연결은 에이전트가 HTTP 를 사용하여 Enterprise Manager 에 연결한다는 것을 나타냅니다.

HTTP 프록시 통신에 대한 에이전트 구성

HTTP 에이전트가 프록시 서버를 통해 Enterprise Manager 에 연결하도록 구성할 수 있습니다. 전달하는 프록시 서버가 아웃바운드 HTTP 트래픽만 허용하고 에이전트가 방화벽 내에서 실행되는 경우 이 구성이 필요합니다. 프록시 서버 구성은 단일 연결뿐 아니라 에이전트에 구성된 모든 HTTP 터널링 연결에 적용됩니다. 이 설정은 여러 Enterprise Manager 가 각각 HTTP 를 통해 연결되는 경우 Enterprise Manager 간의 장애 조치를 구성할 때 고려해야 합니다.

중요! 프록시 서버는 HTTP Post 를 지원해야 합니다.

중요! 프록시에 연결할 수 없으면 에이전트는 이 프록시를 건너뛰고 Enterprise Manager 와 직접 연결합니다. 프록시에 연결할 수 있지만 인증이 실패하는 경우 에이전트는 이 프록시를 통해 계속 Enterprise Manager 에 연결을 시도합니다.

다음 단계를 따르십시오.

1. <*Agent_Home*>/wily 디렉터리로 이동합니다.
2. IntroscopeAgent.profile 을 텍스트 편집기에서 엽니다.
3. 다음 속성의 주석 처리를 제거하여 해당 속성을 설정합니다.

   ```
   introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=프록시 서버의 호스트 이름 또는 IP 주소
   introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=Enterprise Manager 웹 서버 포트
   introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.HttpTunnelingSocketFactory
   ```

4. 다음 속성의 주석 처리를 제거하고 속성을 프록시 호스트 및 포트로 설정합니다.

   ```
   introscope.agent.enterprisemanager.transport.http.proxy.host=호스트 이름
   introscope.agent.enterprisemanager.transport.http.proxy.port=포트 번호
   ```

5. 다음 속성의 주석 처리를 제거하고 속성을 프록시 사용자 이름 및 암호로 설정합니다.

   ```
   introscope.agent.enterprisemanager.transport.http.proxy.username=사용자 이름
   introscope.agent.enterprisemanager.transport.http.proxy.password=암호
   ```

6. 파일을 저장하고 닫습니다.
7. 응용 프로그램을 다시 시작합니다.

HTTPS 통신에 대한 에이전트 구성

에이전트는 HTTPS(HTTP over Secure Sockets Layer)를 사용하여 Enterprise Manager 에 연결할 수 있습니다. HTTPS 는 사용자 페이지 요청과 웹 서버가 반환하는 페이지를 암호화 및 해독합니다. SSL 을 사용하면 온라인 트랜잭션 중에 웹 사이트가 보호됩니다.

다음 단계를 따르십시오.

1. <*Agent_Home*>/wily 디렉터리로 이동합니다.
2. IntroscopeAgent.profile 을 텍스트 편집기에서 엽니다.
3. 다음 속성의 주석 처리를 제거하고 대상 Enterprise Manager 의 호스트 이름 또는 IP 주소로 설정합니다.

   ```
   introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=Enterprise Manager 의 호스트 이름 또는 IP 주소
   ```

4. 다음 속성의 주석 처리를 제거하고 Enterprise Manager 에 포함된 웹 서버의 HTTPS 수신 포트로 설정합니다. 예:

   ```
   introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=8444
   ```

5. 다음 속성의 주석 처리를 제거하고 HTTPS 터널링 소켓 팩터리를 사용하도록 설정합니다. 예:

   ```
   introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.HttpsTunnelingSocketFactory
   ```

6. IntroscopeAgent.profile 을 저장하고 닫습니다.

7. 에이전트와 Enterprise Manager 에 서로 통신할 수 있는 공통 암호 그룹이 있는지 확인합니다.

   참고: 에이전트 측에서는 호스트 컴퓨터에 어떤 암호가 설치되든 Enterprise Manager 와 통신하는 데 사용됩니다. 호스트 컴퓨터 그룹 정책 설정에 설정된 우선 순위에 따라 암호의 우선 순위가 제어됩니다.

8. wily 디렉터리에 있는 사용자를 pfx 파일에 추가하고 해당 사용자에게 적절한 권한 (페이지 50)을 부여합니다.

9. IntroscopeAgent.profile 을 텍스트 편집기에서 엽니다.

   a. 다음 속성의 주석 처리를 제거하고 다음과 같이 pfx 파일 위치 및 암호를 설정합니다.

      ```
      introscope.agent.enterprisemanager.transport.tcp.keystore.DEFAULT=<pfx 파일의 경로>
      introscope.agent.enterprisemanager.transport.tcp.keypassword.DEFAULT=<pfx 암호>
      ```

   b. 파일을 저장하고 닫습니다.

신뢰할 수 있는 인증서를 Trusted Root Certification Authorities 디렉터리에 설치

클라이언트 그룹과 Enterprise Manager 사이에 SSL(Secure Socket Layer) 연결을 사용하는 경우 에이전트는 유효성 검사를 위해 서명된 인증서를 제공해야 합니다. 인증서 가져오기 마법사를 사용하여 SSL 연결에 사용할 적절한 인증서 파일을 설치하고 해당 경로를 지정합니다.

다음 단계를 따르십시오.

1. 응용 프로그램 서버에서 Console Root\Certificates(로컬 컴퓨터)로 이동합니다.
2. Trusted Root Certification Authorities 폴더를 마우스 오른쪽 단추로 클릭하고 "모든 작업", "가져오기"를 선택합니다.

   그러면 인증서 가져오기 마법사가 나타납니다.
3. "다음"을 클릭합니다.
4. 마법사를 진행하면서 다음 필드를 입력합니다.

   File to Import(가져올 파일)

   <*DVD*>\wilyHome\agentCert 와 같이 가져올 파일을 지정합니다.

   Place all certificates in the following store(모든 인증서를 다음 저장소에 저장)

   모든 인증서를 지정한 저장소에 저장합니다.

   Certificate store(인증서 저장소)

   Trusted Root Certification Authorities 와 같은 인증서 저장소를 지정합니다.
5. "다음"을 클릭하고 "마침"을 클릭합니다.

   인증서를 가져옵니다.

신뢰할 수 있는 인증서를 Personal 디렉터리에 설치

신뢰할 수 있는 인증서를 개인 키를 사용한 보안을 유지하기 위해 Personal 디렉터리에 설치할 수 있습니다.

다음 단계를 따르십시오.

1. 클라이언트에서 Console Root\Certificates(로컬 컴퓨터)로 이동합니다.
2. Personal 폴더를 마우스 오른쪽 단추로 클릭하고 "모든 작업", "가져오기"를 선택합니다.

   그러면 인증서 가져오기 마법사가 나타납니다.
3. "다음"을 클릭합니다.

4. 메시지에 따라 다음 필드를 입력합니다.

   파일 이름

   <*DVD*>\wilyHome\agentCer 과 같이 가져올 경로 및 파일을 지정합니다.

   Place all certificates in the following store(모든 인증서를 다음 저장소에 저장)

   모든 인증서를 한 저장소에 배치합니다.

   Certificate store(인증서 저장소)

   Personal 과 같이 인증서를 저장할 저장소를 지정합니다.

5. "다음"을 클릭하고 "마침"을 클릭합니다.

   인증서를 가져옵니다.

통신 연결 확인

Enterprise Manager 및 에이전트를 구성한 후 통신 연결을 확인하십시오. 예를 들어 포트 8081 연결에 성공한 경우 HTTP 통신을 확인합니다.

다음 단계를 따르십시오.

1. <*Agent_Home*>/wily/logs 디렉터리로 이동합니다.
2. 에이전트 로그를 열고 다음 메시지를 찾습니다.

   ```
   [DEBUG] [IntroscopeAgent.HttpOutgoingConnection] Established client connection: host:hostname, port: port num
   ```

   참고: *Hostname* 은 에이전트가 연결하는 컴퓨터의 이름이고 *port num* 은 에이전트와 Enterprise Manager 가 연결된 포트입니다.

   예를 들어 포트 8444 에 연결한 경우 로그에는 다음 메시지가 포함됩니다.

   ```
   [DEBUG] [IntroscopeAgent.HttpOutgoingConnection] Established client connection: host:hostname, port: 8444
   ```

IIS 작업자 프로세스의 사용자 권한 확인

일반적으로 .NET 웹 응용 프로그램은 IIS 작업자 프로세스에서 실행됩니다. 기본적으로 IIS 작업자 프로세스 이름은 *w3wp.exe* 또는 *aspnet_wp.exe* 입니다. 작업자 프로세스를 실행하는 기본 계정은 *NETWORK SERVICE* 입니다. .NET 에이전트를 설치할 때 설치 관리자는 .NET *<Agent_Home>* 또는 *wily* 라고 하는 디렉터리와 *bin*, *ext* 및 *log* 같은 하위 디렉터리, 그리고 성능 모니터 카운터에 액세스할 수 있는 적절한 권한을 가지고 루트 설치 디렉터리를 생성합니다.

그러나 응용 프로그램 풀 수준에서 작업자 프로세스를 실행하도록 다른 계정을 구성할 수도 있습니다. IIS 작업자 프로세스를 실행할 수 있는 다른 사용자 계정을 구성한 경우 다음을 수행하십시오.

- 응용 프로그램을 실행하는 사용자 계정을 확인합니다.
- 응용 프로그램을 실행하는 데 사용할 모든 계정이 *<Agent_Home>* 디렉터리에 액세스할 수 있는 권한을 가지고 있는지 확인합니다.
- Performance Monitor Collection Agent 를 실행하는 계정이 성능 모니터 카운터에 액세스할 수 있도록 사용자 권한을 설정합니다.

응용 프로그램을 실행하는 사용자 확인

일부 조직에서는 *NETWORK SERVICE* 이외의 계정이 IIS 작업자 프로세스를 실행합니다. 다른 계정으로 실행되는 응용 프로그램을 모니터링하기 위해서는 모니터링할 모든 대상 응용 프로그램의 IIS 작업자 프로세스와 관련된 사용자 이름을 확인해야 합니다. 식별되는 각 사용자 이름에 대해 해당 사용자가 *<Agent_Home>* 디렉터리와 성능 모니터 카운터에 액세스할 수 있는 권한을 가지고 있는지 확인해야 합니다.

다음 단계를 따르십시오.

1. 응용 프로그램이 실행 중인지 확인합니다.
2. "작업 표시줄"을 마우스 오른쪽 단추로 클릭하고 "작업 관리자"를 선택하여 "Windows 작업 관리자"를 엽니다.
3. "프로세스" 탭을 클릭합니다.
4. "이미지 이름" 열에서 아래로 스크롤하여 *w3wp.exe* 또는 *aspnet_wp.exe* 프로세스에 해당하는 항목을 찾습니다.
5. "사용자 이름" 열을 확인하여 해당 프로세스를 실행하는 사용자 계정을 확인합니다.

   예를 들어 *w3wp.exe* 작업자 프로세스의 기본 사용자는 *NETWORK SERVICE* 입니다.IIS 작업자 프로세스를 다른 사용자 계정으로 실행하는 경우에는 해당 계정에 적절한 권한이 부여되었는지 확인할 수 있도록 계정 이름을 기록해 두는 것이 좋습니다.

에이전트 디렉터리에 대한 사용자 권한 확인

.NET 에이전트가 제대로 작동하려면 IIS 작업자 프로세스를 실행하는 사용자 계정이 *<Agent_Home>* 디렉터리 내의 파일에 대한 적절한 사용자 권한과 성능 모니터 카운터에 대한 액세스 권한을 가지고 있어야 합니다. 기본적으로 설치 관리자는 *<Agent_Home>* 폴더에 대해 모든 사용자에게 읽기 액세스 권한을 부여합니다. 따라서 이 디렉터리에 대한 액세스를 제한하려면 IIS 작업자 프로세스를 실행하는 사용자 계정만 *<Agent_Home>* 디렉터리에 액세스할 수 있도록 기본 권한을 수정해야 합니다.

다음 단계를 따르십시오.

1. Windows 탐색기에서 루트 .NET 에이전트 폴더를 마우스 오른쪽 단추로 클릭하고 "속성"을 선택합니다.

   폴더의 "속성" 창이 표시됩니다.
2. "보안" 탭을 클릭합니다.
3. "Add"(추가)를 클릭합니다.

   "사용자 및 그룹 선택" 대화 상자가 표시됩니다.
4. *"선택할 개체 이름을 입력하십시오."*에 사용자 이름 전체 또는 일부를 입력합니다.
5. "이름 확인"을 클릭하여 일치하는 사용자 이름을 하나 이상 검색합니다.

6. 원하는 사용자 계정을 선택하고 "확인"을 클릭합니다.

   "권한 항목" 대화 상자가 표시됩니다.

7. "모든 권한" 아래에서 해당 사용자에 대해 *"허용"*이 선택되어 있는지 확인합니다.

   모든 권한을 부여하면 사용자에게 "수정", "읽기 및 실행", "폴더 내용 보기", "읽기" 및 "쓰기" 권한이 부여됩니다.

8. "고급"을 클릭합니다.

   "고급 보안 설정" 대화 상자가 표시됩니다.

9. "여기에 표시된 권한으로 자식 개체 권한 바꾸기"를 선택하여 권한을 하위 디렉터리에 전파한 후 "확인"을 클릭합니다.

성능 모니터 메트릭에 대한 사용자 권한 설정

Introscope Investigator 에서 성능 모니터 메트릭을 보려면 IIS 작업자 프로세스를 실행하는 사용자 계정이 올바른 권한을 가지고 있어야 합니다. wilypermissions 유틸리티를 사용한 경우에는 이러한 권한이 자동으로 정의됩니다. wilypermissions 유틸리티를 사용하지 않은 경우에는 IIS 작업자 프로세스를 실행하는 사용자 계정의 권한을 수동으로 설정해야 합니다.

다음 단계를 따르십시오.

1. "시작" 메뉴에서 "설정", "제어판", "관리 도구", "로컬 보안 정책", "로컬 정책", "사용자 권한 할당"으로 이동합니다.
2. "프로필 단일 프로세스"를 마우스 오른쪽 단추로 클릭하고 "속성"을 선택합니다.
3. "사용자 또는 그룹 추가"를 클릭합니다.
4. IIS 작업자 프로세스를 실행하는 사용자 계정을 사용자 목록에 추가한 후 "확인"을 클릭합니다.
5. "프로필 시스템 성능"을 마우스 오른쪽 단추로 클릭하고 "속성"을 선택합니다.
6. "사용자 또는 그룹 추가"를 클릭합니다.
7. IIS 작업자 프로세스를 실행하는 사용자 계정을 사용자 목록에 추가한 후 "확인"을 클릭합니다.

추가 정보:

에이전트 디렉터리에 대한 기본 사용자 권한 수정 (페이지 40)

계측 사용자 지정

기본적으로 .NET 에이전트는 NativeProfiler 를 사용하여 하나 이상의 IIS 작업자 프로세스에서 실행되는 모든 응용 프로그램 및 응용 프로그램 풀을 계측합니다. 기본 계측은 .NET 환경과 웹 응용 프로그램에 대해 광범위하고 상세한 모니터링을 수행하지만 필요에 맞게 모니터링 대상을 미세 조정할 수 있습니다.

기본 계측 수정

기본적으로 NativeProfiler 는 모든 IIS 웹 응용 프로그램과 응용 프로그램 풀을 계측하지만 IIS 외부에서 실행되는 다른 독립 실행형 응용 프로그램은 계측하지 않습니다. 환경이 더 복잡해짐에 따라 기본 계측을 수정해야 할 수 있습니다. 예를 들어 다음과 같은 작업을 수행할 수 있습니다.

- 배포되는 기본 PBL 또는 PBD 를 변경합니다.
- IIS 응용 프로그램이 아닌 응용 프로그램도 계측하도록 .NET 에이전트를 구성합니다.
- 특정 프로세스나 응용 프로그램을 계측 대상에서 제외합니다.
- 특정 응용 프로그램 풀만 계측하도록 제한합니다.

이러한 변경 작업을 수행하려면 .NET 에이전트 프로필을 구성해야 합니다.

배포할 기본 ProbeBuilder 지시문 지정

에이전트 프로필은 배포할 ProbeBuilder 지시문 파일(*.pbd*)을 지정하는 속성인 *introscope.autoprobe.directivesFile* 을 제공합니다. 개별 ProbeBuilder 지시문 파일은 타이머와 카운터 같이 코드에 삽입되는 특정 프로브를 제어합니다. 배포할 *.pbd* 파일 모음을 정의하는 ProbeBuilder 목록 파일(*.pbl*)로 ProbeBuilder 지시문 파일을 그룹화할 수 있습니다. 에이전트를 설치하면 *introscope.autoprobe.directivesFile* 은 *default-full.pbl* 에 나열된 파일을 사용하여 응용 프로그램을 계측하도록 구성됩니다. 이 *default-full.pbl* 파일은 모든 .NET 구성 요소에 대해 전체 계측을 수행하는 *.pbd* 파일을 참조합니다. *introscope.autoprobe.directivesFile* 속성을 수정하면 다른 *.pbl* 파일을 사용하거나, 특정 *.pbd* 파일을 나열하거나, 배포할 *.pbl* 에 나열된 *.pbd 파일* 목록을 수정할 수 있습니다.

default-typical.pbl 파일을 사용하도록 *introscope.autoprobe.directivesFile* 속성을 수정하는 것이 배포된 기본 ProbeBuilder 지시문을 변경하는 가장 일반적인 방법입니다. *default-typical.pbl* 파일은 모니터링할 구성 요소의 하위 집합을 계측하는 *.pbd* 파일을 참조합니다.

전체 계측에서 표준 계측으로 변경하려면

1. IIS 를 중지합니다.
2. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
3. *introscope.autoprobe.directivesFile* 속성을 찾습니다.
4. *default-full.pbl* 을 *default-typical.pbl* 로 변경합니다. 예:
   `introscope.autoprobe.directivesFile=default-typical.pbl,hotdeploy`
5. 파일을 저장하고 닫습니다.
6. IIS 를 다시 시작합니다.

ProbeBuilder 지시문 및 .NET 에이전트가 모니터링하는 기본 구성 요소에 대한 자세한 내용은 "기본 데이터 수집 사용자 지정" 장을 참조하십시오. ProbeBuilder 지시문 파일에 사용되는 구문 및 모니터링 사용자 지정에 대한 자세한 내용은 ProbeBuilder 지시문 작업 (페이지 91)을 참조하십시오.

IIS 외부에서 실행되는 프로세스와 응용 프로그램 계측

IIS 외부에서 실행되는 응용 프로그램을 모니터링하려면 모니터링하려는 응용 프로그램을 포함하도록 .NET 에이전트 프로필을 수정하십시오. 프로필을 수정하기 전에 포함할 응용 프로그램 실행 파일의 정확한 이름을 확인하십시오.

IIS 외부에서 실행되는 프로세스나 응용 프로그램을 계측하려면

1. IIS 를 중지합니다.
2. *IntroscopeAgent.profile* 을 텍스트 편집기에서 엽니다.
3. *Restricted Instrumentation* 섹션을 찾습니다.
4. 다음 속성에 응용 프로그램 이름을 추가합니다.
   `introscope.agent.dotnet.monitorApplications`

   기본적으로 *w3wp.exe* 와 *aspnet_wp.exe* 는 이 속성에 이미 나열되어 있습니다. 쉼표로 각각을 구분하여 여러 다른 응용 프로그램 이름을 추가할 수 있습니다. 예:
   `introscope.agent.dotnet.monitorApplications=w3wp.exe,aspnet_wp.exe,RandomApp.exe,testapp.exe,readloop.exe`

   중요: 속성 목록은 대/소문자를 구분합니다. 상대 경로와 와일드카드는 지원되지 않습니다. 전체 경로 지정은 지원되지 않으며 응용 프로그램 이름만 사용해야 합니다.
5. *IntroscopeAgent.profile* 에 다음 속성을 추가하고 *false* 로 설정합니다.
   `introscope.agent.dotnet.runInRestrictedMode=false`
6. 파일을 저장하고 닫습니다.
7. IIS 를 다시 시작합니다.

응용 프로그램에 대한 모니터링을 사용하지 않으려면 해당 응용 프로그램을 *introscope.agent.dotnet.monitorApplications* 속성 목록에서 제거하십시오. 제거된 응용 프로그램의 CLR 프로파일러는 활성화된 상태로 유지됩니다. NativeProfiler 가 해제되고 해당 응용 프로그램에 연결된 .NET 에이전트는 Enterprise Manager 에 연결하거나 메트릭을 보고하지 않습니다.

특정 응용 프로그램 풀 계측

기본적으로 모든 응용 프로그램 풀이 계측됩니다. 그러나 오버헤드를 줄이거나 가장 중요한 리소스의 모니터링에만 초점을 두기 위해 계측 대상 IIS 응용 프로그램을 제한할 수 있습니다. 모니터링 대상 응용 프로그램을 제한하려면 계측할 특정 응용 프로그램 풀을 에이전트 프로필에 반드시 지정해야 합니다.

특정 응용 프로그램 풀을 계측하려면

1. IIS 를 중지합니다.
2. *IntroscopeAgent.profile* 을 텍스트 편집기에서 엽니다.
3. *Restricted Instrumentation* 섹션을 찾습니다.

4. 다음 속성의 주석 처리를 제거합니다.
   `introscope.agent.dotnet.monitorAppPools=`
5. 계측할 응용 프로그램 풀을 쉼표로 구분된 목록으로 속성에 추가합니다. 예:
   `introscope.agent.dotnet.monitorAppPools="NULL","DefaultAppPool","AppPool1","AppPool2"`

   참고: IIS 5 는 응용 프로그램 풀 없이도 실행할 수 있습니다. IIS 5 에서 실행되는 응용 프로그램을 계측하는 경우에는 "Null" 값을 사용하십시오.
6. 파일을 저장하고 닫습니다.
7. IIS 를 다시 시작합니다.

응용 프로그램 유휴 시간 구성

Introscope 배포에서 모니터링할 대상 응용 프로그램을 실행하는 각 시스템에 .NET 에이전트를 설치합니다. 에이전트를 설치한 후에는 Microsoft IIS(인터넷 정보 서비스)가 .NET 에이전트를 제어합니다. 계측된 응용 프로그램에서 사용자 작업이 감지되지 않으면 IIS 는 응용 프로그램 프로세스를 중지합니다. 미사용으로 인해 IIS 가 응용 프로그램 프로세스를 중지하면 Introscope Investigator 에서 .NET 에이전트에 해당하는 노드가 사용할 수 없는 상태가 됩니다.

IIS(인터넷 정보 서비스) 관리자를 사용하면 응용 프로그램이 유휴 상태로 표시될 때까지의 시간을 구성할 수 있습니다. 유휴 시간을 구성하면 Introscope Investigator 에서 응용 프로그램 메트릭을 사용할 수 있는 기간을 제어할 수 있습니다.

다음 단계를 따르십시오.

1. "시작", "모든 프로그램", "관리 도구", "인터넷 정보 서비스 관리자"로 이동합니다.
2. 구성할 응용 프로그램을 마우스 오른쪽 단추로 클릭한 다음 "속성"을 선택합니다.
3. "가상 디렉터리" 탭을 클릭합니다.
4. 이 탭의 "응용 프로그램 설정" 영역에서 "구성"을 클릭합니다.
5. "옵션" 탭을 클릭합니다.
6. "세션 상태 사용" 옵션을 확인합니다.
7. 유휴 시간을 분 단위로 설정한 다음 "확인"을 클릭하여 "응용 프로그램 구성" 대화 상자를 닫습니다.
8. "확인"을 클릭하여 "속성" 대화 상자를 닫습니다.

   구성이 설정되었습니다.

.NET 에이전트 제거

운영 체제에서 .NET 에이전트를 제거하려면 다음 방법 중 *하나*를 사용하십시오.

- 대화식으로 제거 (페이지 57)
- 자동 제거 (페이지 58)
- 수동 제거 (페이지 58)

참고: 운영 체제에 맞는 제거 프로그램을 사용하십시오. 예를 들어 .exe 설치 관리자 프로그램을 사용한 경우에는 해당되는 .exe 제거 프로그램을 사용하여 에이전트를 제거해야 합니다. 그렇지 않으면 프로세스가 에이전트를 제거할 수 없습니다.

대화식으로 .NET 에이전트 제거

.NET 에이전트는 계측된 응용 프로그램이 실행 중인 동안은 항상 활성 상태입니다. 에이전트 DLL 파일을 삭제하거나 수정하려면 먼저 계측된 모든 응용 프로그램을 중지해야 합니다. 그런 후 "제어판"의 "프로그램 추가/제거"를 사용하여 로컬에서 에이전트 파일을 제거하거나 제거 프로그램을 사용하여 에이전트 파일을 자동으로 제거할 수 있습니다.

다음 단계를 따르십시오.

1. IIS 서비스를 중지합니다. 그러면 계측된 응용 프로그램이 모두 중지됩니다.
2. "시작", "설정", "제어판", "프로그램 추가/제거"로 이동합니다.
3. 현재 설치되어 있는 프로그램 목록에서 다음 중에서 프로그램 *하나*를 선택합니다.
   - CA APM .NET 에이전트<*release*>(32 비트)
   - CA APM .NET 에이전트<*release*>(64 비트)
4. "제거"를 클릭합니다.

   에이전트를 제거할지 여부를 확인하는 메시지가 표시됩니다.
5. "예"를 클릭하여 에이전트를 제거합니다.

   .NET 에이전트 DLL 의 등록이 취소되고, 관련 환경 변수가 제거되며, 에이전트 파일이 제거됩니다.
6. "프로그램 추가/제거"를 닫습니다.

7. IIS Admin Service 를 재시작하거나 컴퓨터를 재부팅합니다.
8. 에이전트의 루트 설치 디렉터리를 선택하고 마우스 오른쪽 단추를 클릭한 다음 "삭제"를 선택합니다.

   .NET 에이전트가 제거됩니다.

자동 모드로 .NET 에이전트 제거

.NET 에이전트와 관련 파일을 자동 모드로 제거할 수 있습니다.

다음 단계를 따르십시오.

1. IIS 서비스를 중지하여 계측된 모든 응용 프로그램을 중지합니다.
2. 명령 프롬프트에서 다음 명령 중 *하나*를 실행합니다.

   ```
   IntroscopeDotNetAgentInstall*.exe /s /x /v"qn"
   IntroscopeDotNetAgentInstall*.msi /x /qn
   ```

   .NET 에이전트가 제거됩니다.

수동으로 .NET 에이전트 제거

.NET 에이전트와 관련 파일을 수동으로 제거할 수 있습니다.

참고: .NET 에이전트는 계측된 응용 프로그램이 실행 중인 동안은 항상 활성 상태입니다. 에이전트 DLL 파일을 삭제하거나 수정하려면 먼저 계측된 모든 응용 프로그램을 중지해야 합니다.

다음 단계를 따르십시오.

1. C:\WINDOWS\assembly 디렉터리로 이동합니다.
2. wily.Agent.dll 을 마우스 오른쪽 단추로 클릭하고 메뉴에서 "Uninstall"(제거)을 선택합니다.

   에이전트 파일이 제거됩니다.
3. 관리자 권한으로 명령 프롬프트를 엽니다.
   a. 다음과 같이 NativeProfiler 파일을 제거합니다.
      - C:\WINDOWS\system32\regsvr32.exe /u <*Agent_Home*>\bin\wily.NativeProfiler.dll
      - C:\WINDOWS\SysWOW64\regsvr32.exe /u <*Agent_Home*>\bin\x86\wily.NativeProfiler.dll

b. 다음 명령을 실행하여 PerfMon Collector Service 를 제거합니다.
```
sc delete PerfMonCollectorAgent
```

c. 아래의 .NET 에이전트 환경 변수를 다음과 같이 설정합니다.

- com.wily.introscope.agentProfile=<*Agent_Home*>\IntroscopeAgent.profile
- Cor_Enable_Profiling=0x1
- COR_PROFILER={5F048FC6-251C-4684-8CCA-76047B02AC98}

4. IIS 를 재시작하여 제거 프로세스를 완료합니다.

제 3 장: 에이전트 속성 구성

대부분의 에이전트 작업은 *IntroscopeAgent.profile* 파일 내의 속성을 사용하여 구성됩니다. 이 단원에서는 가장 일반적으로 설정하는 에이전트 속성에 대해 설명합니다. 사용 환경에 따라 추가 속성도 적합할 수 있습니다. 각 에이전트 버전에 대해 서로 다른 속성을 설정하거나 서로 다른 기본값을 사용할 수도 있습니다.

이 섹션은 다음 항목을 포함하고 있습니다.

백업 Enterprise Manager 및 장애 조치 속성을 구성하는 방법

에이전트를 설치할 때는 에이전트에서 기본적으로 연결할 Enterprise Manager 의 호스트 이름 및 포트 번호를 지정합니다. 필요한 경우 백업 Enterprise Manager 를 하나 이상 지정할 수도 있습니다. 에이전트와 기본 Enterprise Manager 사이의 연결이 끊어지면 에이전트는 백업 Enterprise Manager 에 연결을 시도할 수 있습니다.

에이전트가 백업 Enterprise Manager 에 연결할 수 있게 하려면 에이전트 프로필에 Enterprise Manager 의 통신 속성을 지정하십시오. 기본 Enterprise Manager 를 사용할 수 없으면 에이전트는 허용되는 연결 목록의 다음 Enterprise Manager 에 연결을 시도합니다. 첫 번째 백업 Enterprise Manager 와 연결할 수 없으면 목록의 다음 Enterprise Manager 에 연결을 시도합니다. 에이전트는 연결에 성공할 때까지 이런 식으로 목록의 각 Enterprise Manager 에 대해 연결을 시도하며, 어떠한 Enterprise Manager 에도 연결할 수 없는 경우에는 10 초 동안 기다렸다가 다시 시도합니다.

다음 단계를 따르십시오.

1. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
2. 각 백업 Enterprise Manager 에 대해 다음 속성을 에이전트 프로필에 추가하여 대체 Enterprise Manager 통신 채널을 하나 이상 지정합니다.

   ```
   introscope.agent.enterprisemanager.transport.tcp.host.NAME
   introscope.agent.enterprisemanager.transport.tcp.port.NAME
   Introscope.agent.enterprisemanager.transport.tcp.socketfactory.NAME
   ```

   여기서 *NAME* 은 새 Enterprise Manager 채널의 식별자로 바꿉니다. 새 채널을 생성할 때는 DEFAULT 나 기존 채널 이름을 사용하지 마십시오. 다음은 백업 Enterprise Manager 를 두 개 생성하기 위한 예입니다.

   ```
   introscope.agent.enterprisemanager.transport.tcp.host.BackupEM1=paris
   introscope.agent.enterprisemanager.transport.tcp.port.BackupEM1=5001
   Introscope.agent.enterprisemanager.transport.tcp.socketfactory.BackupEM1=com.
   custom.postofficehub.link.net.DefaultSocketFactory
   introscope.agent.enterprisemanager.transport.tcp.host.BackupEM2=voyager
   introscope.agent.enterprisemanager.transport.tcp.port.BackupEM2=5002
   introscope.agent.enterprisemanager.transport.tcp.socketfactory.BackupEM2=com.
   wily.isengard.postofficehub.link.net.DefaultSocketFactory
   ```

3. *introscope.agent.enterprisemanager.connectionorder* 속성을 찾아 기본 및 백업 Enterprise Manager 의 식별자를 포함하는 쉼표로 구분된 목록으로 설정합니다. 목록 내의 식별자 순서에 따라 Enterprise Manager 의 연결 순서가 정의됩니다. 예:

   ```
   introscope.agent.enterprisemanager.connectionorder=DEFAULT,BackupEM1,BackupEM
   2
   ```

4. *introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds* 속성을 찾아 에이전트가 기본 Enterprise Manager 에 연결을 시도할 빈도를 지정합니다. 기본 간격은 120 초(2 분)입니다. 예:

   ```
   introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds=120
   ```

5. 변경 내용을 저장하고 *IntroscopeAgent.profile* 파일을 닫습니다.
6. 응용 프로그램을 다시 시작합니다.

응용 프로그램에 특정 에이전트 프로필을 사용하는 방법

기본적으로 웹 응용 프로그램을 모니터링하는 .NET 에이전트는 응용 프로그램의 가상 디렉터리나 컨텍스트 경로에 기반하여 이름이 자동으로 할당됩니다. 웹에 기반하지 않는 응용 프로그램을 모니터링하는 에이전트는 응용 프로그램 도메인 이름에 기반하여 이름이 할당됩니다. 가능하면 자동 에이전트 이름 지정 기능을 사용하는 것이 좋습니다. 그러나 필요한 경우 에이전트에 이름을 명시적으로 할당할 수도 있습니다. 예를 들어 .NET 에이전트 인스턴스가 여러 개 있는 경우에는 인스턴스를 개별적으로 관리하고 수동으로 이름을 할당할 수 있습니다.

에이전트 이름을 수동으로 할당하려면 다음을 수행하십시오.

- 각 에이전트 인스턴스와 응용 프로그램에 대해 별도의 에이전트 프로필을 만듭니다.
- 각 프로필에 프로세스 및 에이전트 이름 속성을 정의합니다.
- 특정 응용 프로그램 및 에이전트 인스턴스의 에이전트 프로필 위치를 설정합니다.

각 응용 프로그램에 대해 별도의 프로필 생성

특정 .NET 응용 프로그램을 모니터링하는 에이전트 인스턴스에 이름을 명시적으로 할당하려면 각 응용 프로그램에 대해 별도의 *IntroscopeAgent.profile* 을 만들어야 합니다. 각 에이전트 인스턴스와 응용 프로그램에 별도의 프로필을 사용하면 프로세스와 에이전트 이름을 제어하고 다른 속성을 필요에 맞게 사용자 지정할 수 있습니다.

다음 단계를 따르십시오.

1. *<Agent_Home>* 디렉터리에 있는 기존 IntroscopeAgent.profile 파일을 복사합니다.
2. 복사한 프로필을 새 디렉터리에 붙여 넣습니다.
3. (선택 사항) 에이전트 인스턴스 프로필로 식별할 수 있도록 프로필 이름을 변경합니다. 예를 들면 *IntroscopeAgentCalc1.profile* 과 같이 변경합니다.

에이전트 이름 정의

사용자 지정 프로세스 이름과 에이전트 이름을 사용하면 특정 응용 프로그램을 모니터링하는 에이전트 인스턴스를 식별할 수 있습니다. 필요에 따라 인스턴스의 다른 속성을 수정할 수도 있습니다.

다음 단계를 따르십시오.

1. 에이전트 인스턴스와 응용 프로그램에 대해 만든 *IntroscopeAgent.profile* 을 텍스트 편집기에서 엽니다. 예를 들어 이름이 *IntroscopeAgentCalc1.profile* 인 사용자 지정 프로필을 *C:\NetApps\wily* 디렉터리에 만든 경우 해당 파일을 텍스트 편집기에서 엽니다.
2. Custom Process Name 섹션을 찾습니다.
3. *introscope.agent.customProcessName* 속성을 사용하여 사용자 지정 프로세스 이름이 표시되도록 지정합니다. 예:
   `introscope.agent.customProcessName=ArcadeCaclcProcess`
4. *introscope.agent.AutoNamingEnabled* 속성을 사용하여 에이전트 이름 자동 지정을 사용하지 않도록 설정합니다. 예:
   `introscope.agent.agentAutoNamingEnabled=false`
5. *introscope.agent.agentName* 속성을 사용하여 사용자 지정 에이전트 이름이 표시되도록 지정합니다. 예:
   `introscope.agent.agentName=ArcadeCalcAgent`
6. 업데이트된 사용자 지정 프로필을 저장하고 닫습니다.

특정 응용 프로그램 및 에이전트 인스턴스의 에이전트 프로필 위치 설정

기본적으로 IntroscopeAgent.profile 파일은 *<Agent_Home>*\wily 디렉터리에 설치됩니다. 예를 들어 기본 위치는 C:\Program Files\CA APM\Introscope<version>\wily 입니다. 특정 응용 프로그램을 모니터링하는 에이전트 인스턴스에 대해 개별 프로필을 만들면 해당 응용 프로그램과 에이전트 인스턴스의 프로필 위치를 지정하는 구성 파일을 만들어야 합니다.

다음 단계를 따르십시오.

1. 텍스트 편집기에서 *<Agent_Home>\Sample.exe.config* 파일을 엽니다.
2. *Sample.exe.config* 파일에서 샘플 *<configSections>* 및 *<com.wily.introscope.agent>* 섹션을 모두 복사합니다.

3. .NET 응용 프로그램의 *web.config* 파일 또는 IIS 가 아닌 응용 프로그램 실행 파일의 독립 실행형 구성 파일을 엽니다.

   응용 프로그램 실행 파일의 구성 파일을 만드는 데 대한 자세한 내용은 응용 프로그램별 구성 (페이지 275)을 참조하십시오.

4. *<env.parameters>* 섹션에서 에이전트 프로필 위치를 수정합니다. 예:

```
<env.parameters>
   <add key="com.wily.introscope.agentProfile"
value="C:\\NETApps\\wily\\IntroscopeAgentCalc1.profile" />
</env.parameters>
```

5. *web.config* 또는 독립 실행형 구성 파일을 저장합니다.
6. 관리되는 응용 프로그램을 다시 시작합니다.

성능 모니터 데이터의 수집 및 사용자 지정 방법

기본적으로 .NET 에이전트는 별도의 Windows 서비스인 Performance Monitor Collection Agent 를 배포하여 모든 Windows 성능 모니터 개체, 카운터 및 인스턴스로부터 메트릭을 수집합니다. Windows 서비스는 IIS 서버에서 실행되는 모든 에이전트 인스턴스와 프로세스를 위해 이 정보를 보고합니다. 설치를 진행하는 동안 이 서비스를 자동으로 시작할지 아니면 수동으로 시작할지 여부를 지정할 수 있습니다. 설치 후에는 서비스 제어판을 사용하여 서비스를 관리할 수 있습니다. 예를 들면 서비스 제어판을 통해 성능 모니터 카운터의 수집을 일시 중지하고 계속하거나 시작 유형을 변경할 수 있습니다.

에이전트 프로필에 포함된 속성을 수정하면 Performance Monitor Collection Agent 가 수집하는 데이터를 사용자 지정할 수 있습니다. 예를 들어 다음을 수행할 수 있습니다.

- 수집할 특정 메트릭을 정규식을 통해 필터링합니다.
- 반환되는 총 성능 모니터 메트릭 수를 제어합니다.
- 성능 모니터 카운터를 확인하는 빈도를 제어합니다.
- Performance Monitor Collection Agent 가 새 성능 모니터 개체를 확인하는 빈도를 제어합니다.
- Performance Monitor Collection Agent 가 새 성능 모니터 개체를 확인하지 않도록 방지합니다.

참고: Investigator 에서 데이터를 사용할 수 있으려면 Performance Monitor Collection Agent 를 실행하는 계정이 성능 모니터 카운터에 액세스할 수 있는 권한을 가지고 있어야 합니다. 기본적으로 Performance Monitor Collection Agent 서비스는 성능 모니터 카운터에 대한 액세스 권한이 있는 로컬 시스템 계정으로 실행됩니다. 그러나 서비스 제어판을 통해 서비스를 실행할 다른 사용자 계정과 암호를 지정할 수 있습니다. 적절한 권한 설정에 대한 자세한 내용은 IIS 작업자 프로세스의 사용자 권한 확인 (페이지 49)을 참조하십시오.

정규식을 사용하여 메트릭 수집 필터링

.NET 에이전트 속성인 *perfmon.metric.filterPattern* 은 에이전트가 읽는 성능 모니터 카운터를 지정합니다. 기본 설정은 다음과 같습니다.

```
introscope.agent.perfmon.metric.filterPattern=|Processor|*|*,|.NET Data
Provider*|*|*,|.NET CLR*|{osprocessname}|*,|.NET CLR
Data|*|*,|Process|{osprocessname}|*,|ASP.NET|*
```

필터는 */Object|Instance|Counter* 또는 */Object|Counter*(인스턴스가 없는 경우) 형식이며 각각 다음을 의미합니다.

- *Object* 는 Memory, Processor 또는 Process 같은 성능 모니터 범주를 나타냅니다.
- *Instance* 는 지정한 개체의 특정 인스턴스를 나타냅니다. Memory 같은 일부 개체에는 인스턴스가 없습니다.
- *Counter* 는 *Object|Instance* 에 대해 수집할 특정 메트릭 유형을 나타냅니다. 예를 들어 .NET CLR Memory 성능 모니터 개체의 카운터로는 # Bytes in all heaps, Gen 0 heap size, # GC handles 및 % time in GC 가 있습니다.

기본 필터에는 *{osprocessname}* 자리 표시자도 포함됩니다. Investigator 에서 *{osprocessname}* 자리 표시자는 모니터링되는 독립 실행형 응용 프로그램의 인스턴스 또는 *w3wp(BusinessServiceAppPool)* 같이 IIS 작업자 프로세스의 응용 프로그램 풀 이름을 나타내도록 대체됩니다.

중요: |*|* 필터를 사용하는 것은 모든 카운터를 인스턴스 없이 열거하도록 성능 모니터에 지정하는 것과 같으며, 이 경우 특정 카운터에서 문제가 발생할 수 있습니다.

introscope.agent.perfmon.metric.filterPattern 속성의 값을 수정하면 .NET 에이전트가 수집하는 성능 모니터 데이터를 조정할 수 있습니다. 예를 들어 기본 필터를 수정하여 보고되는 데이터를 늘리거나 줄일 수 있습니다. 응용 프로그램에 사용자 지정 성능 모니터 카운터를 정의한 경우, 해당 카운터를 포함할 수도 있습니다.

일부 성능 모니터 메트릭은 Microsoft 에서 향후 구현하기 위해 예약되어 있습니다. 이러한 메트릭은 성능 모니터에서 "NotDisplayed"로 태그가 지정되어 있습니다. Introscope Investigator 에서 이러한 메트릭을 보면 자리 표시자 태그가 표시됩니다.

총 메트릭 수의 제한 설정

기본적으로 Performance Monitor Collection Agent 는 사용 가능한 모든 성능 모니터 개체, 인스턴스 및 카운터에 대해 성능 모니터 데이터를 수집합니다. 필요한 경우 보고되는 총 성능 모니터 메트릭 수의 상한을 설정하여 데이터의 전체 범위를 제한할 수 있습니다. 성능 모니터 메트릭의 최대 수를 설정하면 모니터링 대상 서버에서 Performance Monitor Collection Agent 의 오버헤드를 줄일 수 있습니다.

다음 단계를 따르십시오.

1. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
2. *perfmon.metric.limit* 속성을 찾은 후 각 간격당 허용할 성능 모니터 메트릭의 최대 수로 설정합니다. 예:
   ```
   introscope.agent.perfmon.metric.limit=100
   ```
3. *IntroscopeAgent.profile* 파일을 저장하고 닫습니다.

성능 모니터 데이터를 수집하는 빈도 제어

기본적으로 Performance Monitor Collection Agent 는 모든 성능 모니터 개체, 인스턴스 및 카운터에서 15 초마다 메트릭 값을 확인합니다. 이 폴링 간격을 사용하면 이미 발견된 성능 모니터 개체에 대해 최신 데이터가 보고됩니다. Performance Monitor Collection Agent 는 데이터를 수집할 새로운 성능 모니터 개체도 주기적으로 확인합니다. 기본적으로 이와 같은 검색은 오버헤드 없이 사용 가능한 개체를 검색하고 오래된 개체를 제거할 수 있도록 10 분 간격으로 실행됩니다. 이러한 두 가지 간격 모두 에이전트 프로필에서 수정할 수 있습니다.

다음 단계를 따르십시오.

1. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
2. *introscope.agent.perfmon.metric.pollIntervalInSeconds* 속성을 찾은 후 Performance Monitor Collection Agent 의 폴링 간격을 구성합니다. 여기에 설정하는 값은 Performance Monitor Collection Agent 가 메트릭 값을 확인하는 빈도를 제어합니다. 예:
   ```
   introscope.agent.perfmon.metric.pollingIntervalInSeconds=20
   ```
3. *introscope.agent.perfmon.category.browseIntervalInSeconds* 속성을 찾은 후 Performance Monitor Collection Agent 의 검색 간격을 구성합니다. 여기에 설정하는 값은 Performance Monitor Collection Agent 가 새 성능 모니터 개체나 사용되지 않는 성능 모니터 개체를 확인하는 빈도를 제어합니다. 예:
   ```
   introscope.agent.perfmon.category.browseIntervalInSeconds=900
   ```
4. *IntroscopeAgent.profile* 파일을 저장하고 닫습니다.

성능 모니터 개체를 탐색하지 않도록 설정

Performance Monitor Collection Agent 는 성능 모니터 개체를 주기적으로 쿼리하여 사용 가능한 모든 서비스가 포함되었는지 확인합니다. 이를 통해 에이전트는 새로운 카운터를 발견하거나, 필요한 경우 사용하지 않는 카운터를 제거할 수 있습니다. 이 기능은 기본적으로 사용하도록 설정되며 기본 쿼리 간격은 600 초(10 분)로 설정됩니다.

시스템 오버헤드를 줄이기 위해 Performance Monitor Collection Agent 가 새 카운터나 사용되지 않는 카운터를 확인하지 않도록 할 수도 있습니다.

다음 단계를 따르십시오.

1. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
2. *introscope.agent.perfmon.category.browseEnabled* 속성을 찾아 *false* 로 설정합니다. 예:
   `introscope.agent.perfmon.category.browseEnabled=false`
3. *IntroscopeAgent.profile* 파일을 저장하고 닫습니다.

성능 모니터 데이터의 수집 시작

.NET 에이전트는 CA APM PerfMon Collector Service 라는 외부 서비스를 배포하여 성능 모니터 메트릭을 수집하고 보고합니다. 이 서비스를 사용하면 각 .NET 에이전트가 아니라 컴퓨터 수준에서 메트릭을 수집할 수 있습니다. .NET 에이전트의 이 인스턴스만 PerfMon 시스템 리소스를 사용합니다.

참고: 이 서비스는 응용 프로그램 서버에 .NET 에이전트를 배포한 후에 실행하십시오.

다음 단계를 따르십시오.

1. Windows 관리 콘솔에 관리자로 로그인합니다.
2. CA APM PerfMon Collector Service 로 이동합니다.
3. 서비스를 마우스 오른쪽 단추로 클릭하고 "시작"을 클릭합니다.

   데이터 수집이 시작됩니다.

성능 모니터 데이터의 수집 중지

.NET 응용 프로그램 관리에 필요하지 않은 성능 모니터 데이터를 수집하지 않도록 모든 성능 모니터 카운터 수집을 중지할 수 있습니다.

다음 단계를 따르십시오.

1. 서비스 제어판을 엽니다.
2. 서비스 이름 목록에서 Performance Monitor Collector Service 를 찾습니다.
3. Performance Monitor Collector Service 를 선택한 후 마우스 오른쪽 단추를 클릭하고 "속성"을 선택합니다.
4. "중지"를 클릭합니다.
5. 시작 유형으로 "수동" 또는 "사용 안 함"을 선택합니다.
6. "확인"을 클릭합니다.

시작 시간을 제어하는 방법

IIS 를 사용하는 대부분의 조직에서는 각 응용 프로그램 도메인에 대해 .NET 응용 프로그램 풀을 재사용하기 위해 IIS 서비스를 주기적으로 다시 시작합니다. IIS 를 재시작할 때마다 .NET 에이전트도 함께 호출되어 각 응용 프로그램 풀에 속해 있는 응용 프로그램을 계측합니다. 초기 시작 시간은 모니터링하는 응용 프로그램과 클래스의 수, 에이전트 프로필 구성 및 사용자 지정 PBD 파일 존재 여부에 따라 달라질 수 있습니다.

NativeProfiler 를 통해 계측되는 에이전트의 기본 설정을 사용하면 에이전트와 응용 프로그램 서버를 시작하는 데 적절한 시간이 소요됩니다. 몇 가지 선택적인 단계를 수행하여 시작 성능을 개선할 수 있습니다.

.NET 에이전트의 시작 시간을 단축하려면 다음 작업을 수행하십시오.

- 에이전트 프로필에서 *introscope.nativeprofiler.directivematching.cache.max.size* 속성의 값을 변경합니다.

  기본적으로 에이전트는 이전에 찾은 지시문 그룹의 캐시를 메모리 내에 생성하는데, 여기에는 모니터링되는 클래스가 포함됩니다. 에이전트는 사용자가 IIS 를 시작할 때 이전에 발견된 클래스의 캐시를 생성합니다. 이 캐시는 응용 프로그램 코드가 새로운 클래스를 모니터링함에 따라 시간이 지나면서 크기가 증가합니다. 메모리 내 캐시에는 기본적으로 최대 5000 개의 클래스 이름이 저장됩니다. 캐시 크기가 이 제한에 도달하면 에이전트는 캐시가 가득 찼음을 나타내는 항목을 NativeProfiler 로그 파일에 기록합니다.

  IntroscopeAgent.profile 파일에서 *introscope.nativeprofiler.directivematching.cache.max.size* 속성을 사용하여 캐시 크기를 늘리거나 줄일 수 있습니다. 캐시에 저장된 클래스 이름이 5000 개를 초과하는 경우 값을 늘리면 시작 시간을 단축시킬 수 있습니다. 하지만 값을 늘리면 에이전트의 메모리 오버헤드가 증가할 수 있습니다. 속성 값을 줄이면 에이전트의 메모리 오버헤드도 감소합니다. 모니터링하는 클래스 수가 5000 개 미만이거나 크기가 큰 지시문 그룹의 모니터링을 중지할 경우에는 이 값을 줄이는 것이 좋습니다.

- 사용자 지정 PBD 파일에서 클래스와 지시문을 식별한 방법을 확인합니다.

  같은 그룹에 속해 있는 클래스에 *IdentifyInheritedAs* 지시문을 사용하면 에이전트가 상속 계층을 가장 효과적으로 사용할 수 있습니다.

In-Process Side-by-Side 실행을 사용하는 방법

.NET Framework 4 는 서로 다른 .NET Framework 버전을 사용하는 응용 프로그램을 같은 프로세스에서 실행할 수 있습니다. 이 경우 이전 구성 요소는 이전 .NET Framework 버전을 계속 사용하고 새 구성 요소는 새로운 .NET Framework 버전을 사용합니다.

.NET 에이전트는 기본적으로 호스트 프로세스에 먼저 로드되는 응용 프로그램에서 사용하는 .NET Framework 를 계측합니다. 예를 들어, 시작하는 첫 번째 응용 프로그램이 .NET Framework 2.0 을 사용하면 기본적으로 .NET Framework 2.0 구성 요소만 계측됩니다.

com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs 속성을 사용하면 .NET 에이전트가 같은 프로세스에서 .NET Framework 버전 여러 개의 구성 요소를 계측할 수 있습니다.

com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs 속성을 사용하면 모니터링할 .NET Framework 인스턴스를 지정할 수 있습니다. 예를 들어 .NET Framework 4 만 모니터링하도록 이 속성을 설정하면 .NET Framework 4 에서 실행되는 구성 요소의 메트릭만 보고됩니다. *com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs* 속성을 사용하면 In-Process Side-by-Side 실행을 통해 .NET Framework 4 및 .NET Framework 2 응용 프로그램 둘 모두를 모니터링하고 메트릭을 보고할 수 있습니다.

다음 단계를 따르십시오.

1. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
2. *com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs* 속성을 찾습니다.
3. 모니터링할 .NET Framework 버전을 표시하기 위해 다음 예와 같이 이 속성을 쉼표로 구분하여 설정합니다.
   ```
   com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs=v2,v4
   ```
4. 파일을 저장하고 닫습니다.
5. 관리되는 응용 프로그램을 다시 시작합니다.

에이전트 부하 분산 구성

작업 부하가 에이전트에 의해 보고되는 기본 메트릭인 클러스터에서는 MOM 에이전트 부하 분산을 구성하여 전체 클러스터 수용 능력을 최적화할 수 있습니다.

참고: MOM 에이전트 부하 분산에 대한 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

분포 통계 메트릭을 수집하도록 에이전트를 구성하는 방법

"평균 응답 시간" 메트릭을 생성하기 위해 BlamePointTracer 가 분석한 응답 시간 분포를 수집하도록 에이전트를 구성할 수 있습니다. 분포 통계 메트릭은 특정 작업이 어떻게 다른지 설명하는 자세한 설명 정보를 제공합니다. 모니터링된 응답 시간 값의 통계적 분포에 대한 추가적인 데이터는 getExtendedMetricData 웹 서비스를 통해 얻을 수 있습니다.

CA APM 에이전트는 특정 작업에 대한 응답 시간 정보를 수집하도록 구성할 수 있습니다. 응답 시간 정보는 선택한 작업에 대한 평균 응답 시간 메트릭과 쌍을 이루어 분포 통계 메트릭에 저장됩니다.

다음 단계를 따르십시오.

1. <*Agent_Home*>/wily/core/config 디렉터리에 있는 *IntroscopeAgent.profile* 파일을 엽니다.
2. introscope.agent.distribution.statistics.components.pattern 의 주석을 해제하고 편집하여 쌍을 이룬 분포 통계 메트릭을 생성할 평균 응답 시간 메트릭을 지정합니다. 예를 들어 이 식은 다음과 같습니다.

   ```
   ASP\\.NET\\|login_aspx:.*
   ```

   login.aspx 응답 시간에 대한 분포 통계가 생성되었습니다.
3. (선택 사항) 다음 지침을 따라 식을 만듭니다.
   a. 세로 막대와 마침표는 정규식 내에서 특수한 의미를 가지므로 메트릭 노드 구분 기호 앞에는 백슬래시를 사용합니다.
   b. 백슬래시는 에이전트 프로필에서 특수한 의미를 가지므로 백슬래시 앞에는 또 다른 백슬래시를 사용합니다.

      정규식은 에이전트 로그에서 특수 문자 뒤에 표시됩니다. 예:

      ```
      "ASP\.NET\|login_aspx:.*"
      ```
   c. 요약 수준 및 개별 메트릭 정규식을 매칭합니다. 메트릭이 매칭되지 않으면 요약 수준에 대한 분포 통계가 요약되거나 생성되지 않습니다.

      다음 예는 매칭하는 식을 나타냅니다.

      `ASP\\.NET(\\|.*):.*`는 ASP.NET 요약 수준 및 모든 개별 ASP 페이지에 매칭됩니다.

      또한 `ASP\\.NET.*"`는 문자 "ASP.NET"으로 시작하는 다른 메트릭 경로가 없는 한 사용할 수 있습니다.
4. IntroscopeAgent.profile 파일을 저장하고 닫습니다.
5. 에이전트를 다시 시작합니다.

분포 통계 메트릭의 예

구성 속성을 설정한 방법에 따라 다음에 대해 분포 통계를 수집할 수 있습니다.

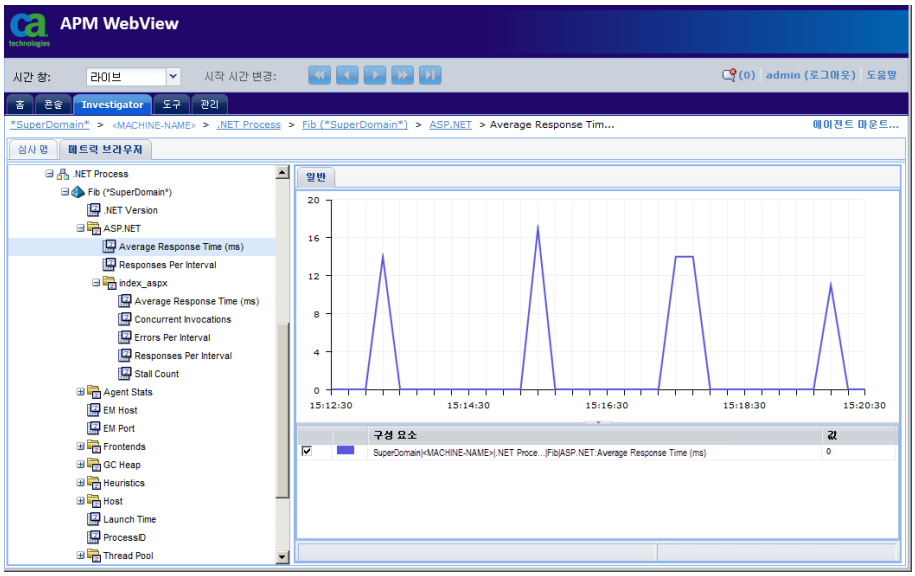
- ASP 페이지 및 요약 수준 분포 메트릭 (페이지 74)
- ASP 페이지 수준 분포 메트릭 (페이지 75)

ASP 페이지 및 요약 수준 분포 메트릭의 예

다음 패턴을 사용하여 ASP 페이지 및 요약 수준 분포 메트릭을 수집하십시오.

```
introscope.agent.distribution.statistics.components.pattern=ASP\\.NET(\\|.*):.*
```

다음 이미지는 ASP 페이지에 대한 분포 통계 메트릭과 .NET 에이전트 노드에 대한 예약 수준을 수집하는 Investigator 를 보여 줍니다.

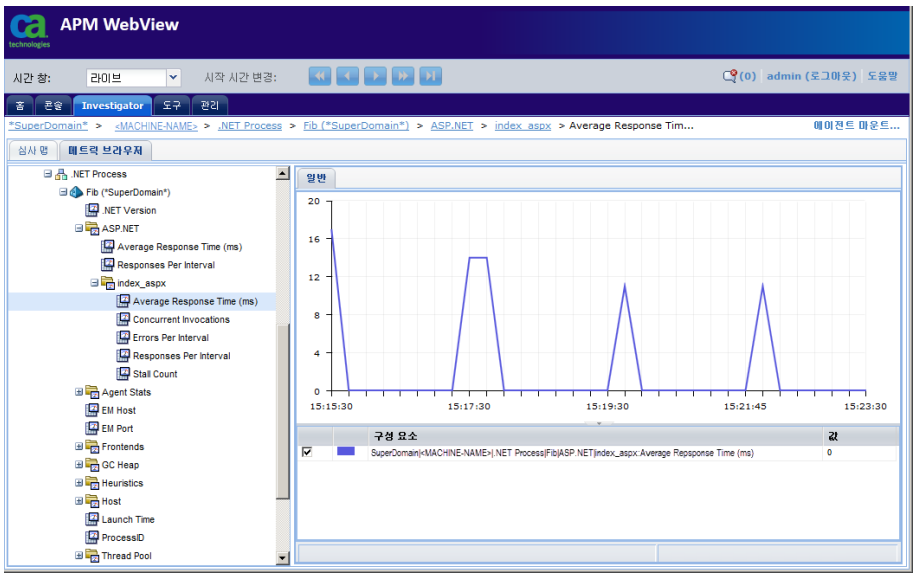


ASP 페이지 분포 메트릭의 예

이 패턴을 사용하여 ASP 페이지 분포 메트릭만 수집하십시오.

```
introscope.agent.distribution.statistics.components.pattern=ASP\\,NET\\|.*
```

다음 이미지는 ASP 페이지 수준 분포 통계 메트릭 .NET 에이전트 노드를 수집하는 Investigator 를 보여 줍니다.



가상 트랜잭션 감지 구성

가상 트랜잭션 모니터링의 구성 설정은 introscope.agent.synthetic.header.names 매개 변수를 사용하여 지정할 수 있습니다.

introscope.agent.synthetic.header.names 매개 변수 값은 모니터링된 HTTP 요청이 가상 트랜잭션의 일부인지 여부를 확인하는 데 사용되는 HTTP 헤더 매개 변수를 나열합니다. 개별 매개 변수 이름은 쉼표로 구분됩니다. 이 매개 변수를 정의하지 않거나 값을 비워 두면 가상 트랜잭션이 감지되지 않습니다. HTTP 헤더 매개 변수 이름을 여러 개 정의할 경우, 매개 변수를 지정한 순서대로 검토가 이루어집니다. 값이 지정된 첫 번째 HTTP 매개 변수가 가상 트랜잭션 정의에 사용됩니다.

가상 트랜잭션이 보고되는 노드는 다음과 같이 각 트랜잭션을 감지하는 데 사용되는 특정 HTTP 헤더 매개 변수에 따라 결정됩니다.

- 매개 변수 값이 *lisaframeid* 또는 *x-wtg-info* 이외의 값이면 HTTP 매개 변수 값 자체가 노드 이름으로 사용됩니다. 유효한 노드 이름이 사용되도록 적절한 수정 작업이 수행됩니다.
- 매개 변수 값이 *lisaframeid* 인 경우 가상 노드 이름은 CA LISA 입니다.
- 매개 변수 값이 *x-wtg-info* 인 경우 HTTP 헤더 매개 변수 값에 이름 및 값 쌍 시퀀스가 포함된 것으로 간주됩니다. 각 쌍은 앰퍼샌드 기호로 구분되며 각 쌍 내에서 특성 이름과 값은 등호로 구분됩니다. 가상 트랜잭션 노드 이름은 *group, name, ipaddress* 및 *request_id* 의 값을 | 노드 구분 기호로 구분하여 구성됩니다.

예를 들어 매개 변수가 다음과 같은 경우:

```
introscope.agent.synthetic.header.names=Synthetic_Transaction,x-wtg-info,lisaframeid
```

다음의 *x-wtg-info* 헤더를 사용할 경우 SampleGroup|sample|192.168.193.1|start 라는 노드 아래에 메트릭이 보고됩니다.

```
clear
synthetic=true&instance=ewing&name=sample&group=SampleGroup&version=4.1.0&ipaddress=192.168.193.1&sequencenumber=1&request_id=start&executiontime=1226455047
```

x-wtg-info HTTP 헤더 매개 변수 값에 정의되지 않은 모든 특성에는 다음과 같은 기본값이 사용됩니다.

- *group*=unknownGroup
- *name*=unknownScript
- *ipaddress*=0.0.0.0
- *request_id*=Action

introscope.agent.synthetic.header.names 를 정의하지 않으면 다음과 같은 구성 매개 변수가 무시됩니다.

```
introscope.agent.synthetic.node.name=Synthetic Users
```

가상 트랜잭션으로 인식된 트랜잭션이 보고되는 노드입니다. 이 노드는 *Frontends(프런트엔드)|Apps(응용 프로그램)|<WebAppName>*에 위치하며, 여기서 *<WebAppName>*은 웹 응용 프로그램 이름입니다. 이 값은 기본적으로 Synthetic Users 입니다.

`introscope.agent.non.synthetic.node.name=Real Users`

가상 트랜잭션으로 인식되지 않은 트랜잭션이 보고되는 노드입니다. 이 노드는 *Frontends(프런트엔드)|Apps(응용 프로그램)|<WebAppName>*에 위치하며, 여기서 *<WebAppName>*은 웹 응용 프로그램 이름입니다. 정의하지 않을 경우 *<WebAppName>* 아래에 노드가 추가로 생성되지 않습니다.

`introscope.agent.synthetic.user.name=Synthetic_Trace_By_Vuser`

해당 값이 가상 사용자 이름으로 사용되는 HTTP 헤더 매개 변수의 이름입니다. 가상 사용자 이름은 서로 다른 가상 트랜잭션을 구분하기 위해 사용됩니다. 각 가상 사용자 이름에 해당하는 노드가 *Synthetic User* 노드 아래에 생성됩니다. 이 구성 매개 변수가 정의되고 동일한 이름의 HTTP 헤더 매개 변수가 존재하면 가상 트랜잭션 메트릭이 보고됩니다. 트랜잭션은 *<Synthetic Users>|<Synthetic User>* 노드 아래에 보고됩니다.

- *introscope.agent.synthetic.node.name* 구성 매개 변수가 *<Synthetic Users>* 노드 이름을 결정합니다.
- HTTP 헤더 매개 변수 값이 *<Synthetic User>* 노드 이름을 결정합니다.

참고: 이러한 속성의 변경 내용은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

TagScript 유틸리티 사용

CA TagScript 유틸리티를 HP Vugen 과 함께 사용하여 가상 사용자 정보의 추출을 지정할 수 있습니다.

TagScript 유틸리티를 사용하려면

1. TagScript 유틸리티를 엽니다.

   Windows:

   `<Agent_Home>\wily\tools\TagScript.bat`

   UNIX:

   `<Agent_Home>/wily/tools/TagScript.sh`

   어느 환경을 대상으로 스크립트를 수정할지 묻는 메시지가 표시됩니다.

2. 다음 옵션 중 하나를 클릭합니다.
   - "Performance Testing"(성능 테스트) - HP Loadrunner 스크립트
   - "프로덕션" - HP Business Process Monitor 또는 Sitescope 스크립트
   - "Un-tag"(태그 해제) - 태그 지정 프로세스 전환
3. HP Vugen 스크립트가 저장되어 있는 디렉터리로 이동한 후 각 .c 스크립트를 두 번 클릭하여 엽니다.

   HP Vugen 스크립트 .c 파일이 모두 백업되고 수정된 버전으로 대체됩니다.
4. HP Vugen 이 열려 있고 유틸리티가 실행 중이면 수정된 스크립트를 다시 로드하라는 메시지가 나타납니다. 이 메시지가 표시되면 "모두 예"를 클릭하십시오.
5. TagScript 유틸리티를 닫거나, 파일 선택 대화 상자에서 "취소" 단추를 클릭합니다. TagScript 유틸리티를 반드시 닫아야 하는 것은 아니며, 대부분의 사용자는 HP Vugen 을 사용하는 동안 이 유틸리티를 열어 둡니다. 스크립트가 수정되거나 새로운 스크립트가 생성된 경우, 유틸리티를 열어 두면 프로세스가 더 간단합니다.
6. 다음과 같은 위치에서 스크립트 태그가 지정되었는지 확인합니다.
   - 각 스크립트의 시작 부분에 HP Vugen 코드의 신규 단락이 삽입되었는지 확인합니다.
   - 모든 lr_start_transaction 및 lr_end_transaction 전과 스크립트 맨 마지막에 태그가 표시되는지 확인합니다.
7. (선택 사항) 개별 Blame 스택 집합을 사용하여 각 가상 사용자를 HP Loadrunner 성능 테스트를 통해 추적합니다. 각 사용자를 추적하려면 스크립트 시작 부분의 선언 단락에서 다음 행에 대한 주석 처리를 제거합니다.

   ```
   web_add_auto_header("Synthetic_Trace_By_Vuser",vuserOverview)
   ```

   참고: "프로덕션" 태그 지정 스크립트에 대해 이 옵션의 주석 처리를 제거하면 각 요소 또는 가상 생성기에 대해 개별 Blame 스택 집합이 생성됩니다.

제 4 장: 기본 데이터 수집 사용자 지정

에이전트를 설치하면 몇 가지 기본적인 ProbeBuilder 지시문 파일(*.pbd*)이 함께 설치되고 .NET Framework 의 여러 일반적인 구성 요소와 ASP.NET 응용 프로그램에 대해 모니터링이 설정됩니다. 이 단원에서는 기본적으로 제공되는 모니터링 기능 및 사용자 지정 *.pbd* 파일을 작성하지 않고 기본 모니터링을 수정하는 방법을 설명합니다.

이 섹션은 다음 항목을 포함하고 있습니다.

기본 ProbeBuilder 파일 정보

ProbeBuilder 지시문 파일(*.pbd*)은 삽입할 프로브 유형과 코드에서 프로브를 배치할 위치를 지정합니다. 타이머와 카운터 같은 프로브는 에이전트가 Introscope Enterprise Manager 에 보고하는 메트릭을 제어합니다. 사용자 지정 작업 없이 에이전트가 메트릭을 기본 설정 그대로 수집할 수 있도록 Introscope 에는 다음과 같이 사전 정의된 기본 ProbeBuilder 파일 집합이 포함되어 있습니다.

- *ProbeBuilder 지시문 파일(PBD)*에는 프로브 삽입 및 응용 프로그램 메트릭 검색에 대한 지침이 들어 있습니다.
- *ProbeBuilder 목록 파일(PBL)*에는 구성 요소 전체 또는 일부를 모니터링하기 위해 함께 배포해야 하는 PBD 파일 목록이 들어 있습니다.

참고: 모든 메트릭은 시스템 클록에 의해 설정된 시간을 사용하여 계산됩니다. 트랜잭션 중에 시스템 클록을 재설정하면 해당 트랜잭션에 대해 보고되는 경과 시간이 정확하지 않을 수 있습니다.

기본 ProbeBuilder 파일을 사용하면 .NET 환경의 가장 일반적인 구성 요소를 모니터링할 수 있습니다. 기본 파일 집합의 설정을 미세 조정하여 일반 구성 요소에 대한 모니터링을 사용자의 환경에 맞게 사용자 지정할 수 있습니다.

기본 PBD 로 추적되는 구성 요소

기본 Introscope PBD 파일이 추적하는 .NET 구성 요소는 다음과 같습니다.

- .NET Directory Services
- .NET Messaging
- .NET Remoting
- ADO.NET
- ASP.NET
- 엔터프라이즈 서비스
- 네트워크 소켓
- SMTP 메일
- 웹 서비스

기본 ProbeBuilder 지시문 파일(PBD)

다음 표에는 .NET 에이전트와 함께 설치되는 기본 PBD 파일에 대한 설명이 나와 있습니다.

PBD 파일 이름	설명
appmap.pbd	이 파일은 응용 프로그램 심사 맵을 계측하기 위한 추적 프로그램 지시문을 제공합니다.
appmap-soa.pbd	이 파일은 설치 시 CA APM for SOA 를 사용하도록 설정했는지 여부에 따라 SOAP 스택 또는 .NET Framework 클래스 라이브러리의 응용 프로그램 심사 맵을 위한 추적 프로그램 지시문을 제공합니다.
appmap-soa.spm.pbd	이 파일은 .NET Framework 에 지원되는 SOAP 스택의 응용 프로그램 심사 맵을 위한 추적 프로그램 지시문을 제공합니다. 이 파일은 설치 시 CA APM for SOA 를 사용하지 않도록 설정한 경우에만 설치됩니다. 설치 시 CA APM for SOA 를 사용하도록 설정한 경우에는 이 파일 이름이 *appmap-soa.pbd* 로 변경되고 기본 에이전트 *appmap-soa.pbd* 의 이름이 *appmap-soa.core.pbd* 로 변경됩니다.
bizrecording.pbd	이 파일은 에이전트 비즈니스 기록을 설정하기 위한 추적 프로그램 정의와 지시문을 제공합니다.

biz-trx-http.pbd	이 파일은 비즈니스 중심의 HTTP 계측을 위한 추적 프로그램 지시문을 제공합니다.
dotnet.pbd	이 파일은 .NET Framework 클래스 라이브러리를 지원하기 위한 지시문을 제공합니다.
errors.pbd	이 파일은 심각한 오류로 간주되는 코드 수준의 이벤트를 지정하여 ErrorDetector 를 구성합니다. 기본적으로 프런트엔드 오류와 백엔드 오류만 심각한 오류로 간주됩니다. 즉, 사용자에게 오류 페이지로 표시되거나, ADO.NET, 메시징 등의 백엔드 시스템 문제를 나타내는 오류만 심각한 오류로 간주됩니다.
httpheaderdecorator.pbd	이 파일은 CA CEM 과의 통합 솔루션의 일부인 HTTP Header Decorator 를 사용하도록 설정하는 데 사용됩니다.
leakhunter.pbd	이 파일은 Introscope LeakHunter 를 위한 계측 설정을 제공합니다. 대개 이 파일의 내용은 수정할 필요가 없습니다.
sharepoint-full.pbd	이 파일은 Microsoft SharePoint 데이터 추적 기능의 설정/해제 스위치(*TurnOn* 지시문 형식)를 제공합니다. 대부분의 추적 프로그램 그룹은 설정되어 있습니다.
sharepoint-typical.pbd	이 파일은 Microsoft SharePoint 데이터 추적 기능의 설정/해제 스위치(*TurnOn* 지시문 형식)를 제공합니다. 추적 프로그램 그룹의 일부만이 설정되어 있습니다.
skips.pbd	이 파일은 다시 시작 문제로 인해 특정 어셈블리, 클래스 및 메서드를 건너뛰도록 NativeProfiler 에 지시합니다.
spm-correlation.pbd	이 파일은 구성 요소 간 트랜잭션 추적의 상관 관계를 제어하는 지시문을 제공합니다. 이 파일은 CA APM for SOA 를 사용할 경우 프로세스 간 트랜잭션 추적을 사용하는 데 필요합니다.
sqlagent.pbd	이 파일은 SQL 에이전트 구성 파일로, ADO.NET 공급업체 라이브러리(.dll)를 계측하는 데 사용합니다. 일반적으로 이 파일은 편집할 필요가 없습니다.
toggles-full.pbd	이 파일은 다른 지시문 파일에 제공되는 추적 기능에 대한 설정/해제 스위치를 *TurnOn* 지시문 형식으로 제공합니다. 대부분의 추적 프로그램 그룹은 설정되어 있습니다.
toggles-typical.pbd	이 파일은 다른 지시문 파일에 제공되는 추적 기능에 대한 설정/해제 스위치를 *TurnOn* 지시문 형식으로 제공합니다. 추적 프로그램 그룹의 일부만이 설정되어 있습니다.

webservices.pbd	이 파일은 .NET 웹 서비스 모니터링을 지원하기 위한 지시문을 제공합니다. 이 파일의 내용은 설치 시 CA APM for SOA 를 사용하도록 설정했는지 여부에 따라 달라집니다. 사용하도록 설정한 경우에는 .NET 웹 서비스와 SOAP 스택이 모니터링되고 사용하지 않도록 설정한 경우에는 .NET 웹 서비스만 모니터링됩니다.

추가 정보:

기본 추적 프로그램 그룹 및 toggles 파일 (페이지 83)
추적 프로그램 그룹 설정 또는 해제 (페이지 84)

이전 릴리스의 기본 PBD 파일

에이전트에서는 기본적으로 현재 릴리스의 PBD 및 PBL 파일을 사용합니다. 그러나 제품에서는 <*Agent_Home*>/wily/examples/legacy 디렉터리에 이전 릴리스의 PBD 및 PBL 파일을 제공합니다. 이 디렉터리의 각 파일 이름에는 -legacy 접미사가 사용됩니다(예: default-full-legacy.pbl).

기본 PBL(ProbeBuilder 목록) 파일

각 에이전트와 함께 사용할 수 있는 .pbl 파일 집합에는 다음 두 가지가 있습니다.

- **default-full.pbl**: 이 파일이 기본값입니다. 대부분의 추적 프로그램 그룹이 설정된 PBD 파일을 참조합니다. Introscope 는 기본적으로 이 집합을 사용하여 전체 Introscope 기능을 보여 줍니다.
- **default-typical.pbl**: 참조되는 .pbd 파일의 추적 프로그램 그룹 중 일부가 설정되어 있습니다. 이 표준 집합에는 일반적인 설정이 포함되어 있으며 특정 환경에 맞게 이 집합을 사용자 지정할 수 있습니다.

추적 프로그램 그룹은 PBD 파일에 있으며 PBL 파일에서 참조됩니다. 추적 프로그램 그룹은 일련의 클래스에 대한 정보가 보고되도록 합니다. .pbd 파일에서 "flag"라는 용어는 추적 프로그램 그룹 정보를 나타냅니다. 예를 들어 *TraceOneMethodIfFlagged* 또는 *SetFlag* 는 추적 프로그램 그룹 정보를 정의합니다.

기본 추적 프로그램 그룹 및 toggles 파일

추적 프로그램 그룹은 일련의 클래스에 적용되는 일련의 추적 프로그램으로 구성됩니다. 예를 들면 모든 시스템 메시징 클래스의 응답 시간과 속도를 보고하는 추적 프로그램 그룹이 있습니다.

일부 추적 프로그램 그룹을 설정하거나 해제하여 시스템에 대한 메트릭 수집을 미세 조정할 수 있습니다. 이 경우 추적 프로그램 그룹의 구성 방식에 따라 오버헤드 사용이 늘어나거나 줄어드는 영향이 있습니다.

추적 프로그램 그룹은 기본 *default-full.pbl* 및 *default-typical.pbl* 파일이 참조하는 *toggles-full.pbd* 및 *toggles-typical.pbd* 파일에서 수정합니다. *toggles-full.pbd* 파일에서는 모든 기본 추적 프로그램 그룹의 추적을 사용하도록 설정하여 검색된 모든 .NET 구성 요소에 대한 추적을 사용합니다.

toggles-typical.pbd 파일에서는 추적 프로그램 그룹의 일부에 대해 추적 기능을 해제합니다. *toggles-typical.pbd* 파일에서는 기본적으로 다음과 같은 기본 추적 프로그램 그룹에 대해 추적 기능이 해제됩니다.

- 네트워크 구성: SocketTracing
- 트랜잭션 유틸리티 추적: ContextUtilTracing
- 런타임 원격 추적: RemotingWebServiceTracing
- 시스템 웹 메일 추적: WebMailTracing

대부분의 경우 기본 *toggles-full.pbd* 파일과 *toggles-typical.pbd* 파일은 편집 없이 그대로 사용할 수 있습니다. 그러나 일부 추적 프로그램 그룹을 설정하거나 해제하여 메트릭 수집을 미세 조정할 수는 있습니다. 예를 들어 *toggles-typical.pbd* 파일을 사용하는 경우 *Network Configuration* 섹션을 찾은 후 *TurnOn* 문의 주석 처리를 제거하여 *SocketTracing* 그룹을 사용하도록 설정하면 소켓 메트릭에 대해 추적 기능을 추가할 수 있습니다.

```
TurnOn: SocketTracing
```

이와 마찬가지로 *TurnOn* 문의 주석 처리를 제거하여 특정 추적 프로그램 그룹 추적을 중지함으로써 시스템 오버헤드를 줄일 수 있습니다. 예를 들어 *toggles-full.pbd* 파일을 사용하고 있지만 SQL 에이전트 작업을 추적하지 않으려는 경우 *SQL Agent Tracing* 섹션을 찾은 후 SQL 에이전트 추적 프로그램 그룹에 대해 *TurnOn* 문의 주석 처리를 제거하여 해당 작업에 대한 추적을 중지할 수 있습니다.

```
#TurnOn: SQLAgentCommands
#TurnOn: SQLAgentDataReaders
#TurnOn: SQLAgentTransactions
#TurnOn: SQLAgentConnections
```

기존 추적 프로그램 그룹에 클래스를 추가하여 추적 기능을 사용자 지정할 수도 있습니다.

추적 프로그램 그룹 설정 또는 해제

일부 추적 프로그램 그룹을 설정하거나 해제하여 시스템에 대한 메트릭 수집을 미세 조정할 수 있습니다.

다음 단계를 따르십시오.

1. *default-full.pbl* 또는 *default-typical.pbl* 에 따라 *toggles-full.pbd* 또는 *toggles-typical.pbd* 중 어느 파일 유형을 사용 중인지에 따라 적절한 파일을 찾아서 엽니다. 이러한 파일은 *<Agent_Home>* 디렉터리에서 찾을 수 있습니다.
2. 설정하거나 해제할 추적 프로그램 그룹을 찾습니다.

3. 행 시작 부분에 파운드 기호(#)를 추가하거나 제거하여 해당 행을 주석 처리하거나 주석 처리를 제거합니다. 다음 예의 지시문은 설정된 추적 프로그램 그룹과 해제된 추적 프로그램 그룹을 보여 줍니다.

   ```
   TurnOn: SocketTracing
   ```

   이 추적 프로그램 그룹은 설정되어 있습니다. 행의 주석 처리가 제거되어 있습니다.

   ```
   #TurnOn: SocketTracing
   ```

   이 추적 프로그램 그룹은 해제되어 있습니다. 행은 주석 처리되어 있습니다.

4. *toggles-full.pbd* 또는 *toggles-typical.pbd* 를 저장합니다.

   구성이 설정되었습니다.

에이전트 연결 메트릭 구성

기본적으로 Introscope 에서는 Enterprise Manager 에 연결된 에이전트의 연결 상태에 대해 모니터링 가능한 메트릭을 생성합니다. 에이전트 연결 메트릭을 모니터링하여 에이전트와 Enterprise Manager 간의 현재 연결 상태를 확인할 수 있습니다.

에이전트 연결 메트릭은 Workstation Investigator 에서 다음과 같이 Enterprise Manager 프로세스(사용자 지정 메트릭 호스트) 아래에 나타납니다.

```
Custom Metric Host (Virtual) \ Custom Metric Process(Virtual) \ Custom Metric Agent (Virtual) (*SuperDomain*) \ Agents \ <HostName> \ <Agent Process Name> \ <Agent Name> \ ConnectionStatus
```

연결 메트릭의 값은 다음과 같습니다.

- 0 - 에이전트에 대한 데이터를 사용할 수 없음
- 1 - 에이전트가 연결됨
- 2 - 에이전트 보고 시간이 느림
- 3 - 에이전트의 연결이 끊어짐

에이전트의 연결이 끊어진 경우 "주목할 사항" 이벤트도 생성됩니다. 사용자는 다른 이벤트와 마찬가지로 기록 쿼리 인터페이스를 사용하여 에이전트 연결 끊김을 쿼리할 수 있습니다. 에이전트 연결 끊김 이벤트는 Workstation 콘솔의 "개요" 탭에서 응용 프로그램 건전성을 평가하는 데 사용되는 데이터의 일부입니다.

에이전트와 Enterprise Manager 의 연결이 끊기고 나면 Introscope 에서는 에이전트가 시간 초과될 때까지 계속해서 연결 끊김 상태 메트릭을 생성합니다. 에이전트가 시간 초과되면 더 이상 연결 메트릭이 생성되거나 Enterprise Manager 에 보고되지 않습니다.

다음 단계를 따르십시오.

1. Enterprise Manager 가 설치된 컴퓨터에서 *<Introscope_Home>/config* 디렉터리에 있는 *IntroscopeEnterpriseManager.properties* 파일을 엽니다.
2. 다음 속성을 수정합니다.
   `introscope.enterprisemanager.agentconnection.metrics.agentTimeoutInMinutes`
   시간 단위는 분입니다.
3. *IntroscopeEnterpriseManager.properties* 를 저장합니다.

참고: Enterprise Manager 속성에 대한 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

소켓 메트릭 해제

기본적으로 .NET 에이전트는 개별 소켓의 입력 및 출력 대역폭 메트릭을 수집하도록 구성됩니다. 그러나 소켓별 대역폭을 추적하는 메트릭은 상당한 오버헤드가 발생할 수 있습니다. 소켓 수준의 네트워크 메트릭을 수집하는 데 프로세서 또는 I/O 시간이 많이 소요되는 경우에는 이러한 소켓 메트릭의 수집 기능을 완전히 해제할 수 있습니다.

소켓 메트릭 보고를 해제하려면

1. *IntroscopeAgent.profile* 파일을 텍스트 편집기에서 엽니다.
2. Agent Socket Rate Metrics 섹션을 찾습니다.
3. *introscope.agent.sockets.reportRateMetrics* 속성을 *false* 로 설정합니다.
   예:
   `introscope.agent.sockets.reportRateMetrics=false`
4. *IntroscopeAgent.profile* 을 저장하고 닫습니다.

.NET 에이전트 로깅 옵션 구성

다음 단원에서는 .NET 에이전트를 세부 정보 표시 모드에서 실행하고 에이전트의 로그 파일 옵션을 설정하는 방법을 설명합니다. Introscope 용 .NET 에이전트는 *Log4net* 기능을 사용하여 이와 같은 기능을 수행합니다. 다른 *Log4net* 기능을 사용하려면 http://logging.apache.org/log4net/release/features.html 페이지에서 Log4net 문서를 참조하십시오.

세부 정보 표시 모드로 .NET 에이전트 실행

.NET 에이전트를 세부 정보 표시 모드로 실행하면 많은 세부 정보가 로그 파일에 기록되어 디버그할 때 도움이 됩니다.

세부 정보 표시 모드로 .NET 에이전트를 실행하려면

1. .NET 에이전트를 중지합니다.
2. *logging.config.xml* 파일을 엽니다.
3. *level value* 특성을 **VERBOSE** 로 변경합니다. 기본값은 *INFO* 입니다.

```
<root>
<level value="VERBOSE" />
<appender-ref ref="logfile" />
<appender-ref ref="console" />
</root>
```

4. *logging.config.xml* 파일을 저장하고 .NET 에이전트를 다시 시작합니다.

.NET 에이전트 로그 파일 위치 변경

로그 파일은 기본적으로 *C:\Program Files\CA Wily\Introscope<version>\wily\logs* 아래의 *<Agent_Home>\logs* 디렉터리에 기록되며 여기에서 *<version>*은 설치되어 있는 Introscope 버전입니다. 로그 파일을 좀 더 쉽게 사용할 수 있도록 .NET 에이전트 로그 파일의 위치를 변경할 수 있습니다.

.NET 에이전트 로그 파일의 위치를 변경하려면

1. .NET 에이전트를 중지합니다.
2. *logging.config.xml* 파일을 엽니다.
3. 다음과 같이 **file value** 특성을 로그 파일의 원하는 위치로 변경합니다.
   ```
   <file value="c:\introscope_logs\IntroscopeAgent.log" />
   ```
4. *logging.config.xml* 파일을 저장합니다.
5. .NET 에이전트를 다시 시작합니다.

에이전트의 이름을 구성한 경우에는 이름이 지정된 에이전트 로그가 기본 위치 또는 방금 지정한 새 위치의 *logs* 디렉터리에 기록됩니다.

.NET 에이전트 로그 파일 및 자동 에이전트 이름 지정

기본적으로 .NET 에이전트의 이름은 자동으로 지정됩니다. .NET 에이전트 이름이 자동으로 검색되면 해당 에이전트와 관련된 로그 파일의 이름도 동일한 정보를 사용하여 자동으로 지정됩니다. 에이전트가 생성하는 로그 파일에는 계측 프로세스 동안 삽입된 프로브 및 사용된 PBD 에 대한 정보가 기록됩니다. 기본적으로 자동 이름 지정 기능을 사용하면 파일 이름에 타임스탬프를 사용하여 로그 파일이 처음 생성됩니다. 예:

```
AutoProbe20060928-175024.log
```

그런 후 에이전트 이름을 사용할 수 있게 되면 해당 에이전트 이름을 포함하도록 로그 파일 이름이 변경됩니다. 예를 들어 에이전트 이름이 *MyDomain//MyAgent* 이고, 여기에서 *MyDomain* 이 도메인이고 *MyAgent* 가 인스턴스인 경우 로그 파일 이름은 다음과 같습니다.

```
AutoProbeMyDomain_MyStuff.log
```

로그의 실제 이름이 아니라 타임스탬프 이름이 사용된 로그 파일이 있는 경우에는 에이전트 이름을 확인하기 전에 프로세스 시간이 초과된 것일 수 있습니다. 또한 고급 Log4Net 기능을 사용할 경우에는 자동 이름 지정 기능이 제대로 작동하지 않을 수 있습니다.

참고: 클래스 경로에 있는 리소스에서 .NET 에이전트 프로필을 로드하면 *IntroscopeAgent.profile* 파일이 리소스 내에 위치하기 때문에 NativeProfiler 가 로그 파일에 정보를 기록할 수 없습니다.

로그 파일 자동 이름 지정 기능을 사용하지 않으려면 에이전트 프로필에서 introscope.agent.disableLogFileAutoNaming (페이지 182) 속성을 true 로 설정하십시오.

기본 도메인 로그

기본 도메인은 메트릭을 보고하기 위해 Enterprise Manager 에 연결하지 않으며 자체적으로 응용 프로그램을 실행하지도 않습니다. 그러나 기본 도메인에 속한 .NET 에이전트가 기본 도메인에 호스트되는 모든 응용 프로그램 도메인에 대한 모든 바이트 코드 계측을 처리하기 때문에 로그 파일은 생성됩니다. 이러한 파일 중 하나인 *AutoProbe.DefaultDomain.log* 에는 기본 도메인 내에서 발생하는 바이트 코드 계측에 대한 정보가 포함됩니다. 모든 바이트 코드 계측은 기본 도메인에서 이루어지기 때문에 이러한 로그 파일에는 계측과 관련된 중요 정보가 들어 있습니다.

기본 도메인에서는 .NET 에이전트에 대해 *IntroscopeAgent.DefaultDomain.log* 파일도 생성됩니다.

제 5 장: ProbeBuilder 지시문 작업

에이전트는 .NET 응용 프로그램의 구성 요소 대부분에 대해 모니터링 기능을 기본적으로 제공합니다. 그러나 Introscope 를 가장 효과적으로 사용하려면 사용 중인 응용 프로그램에 맞는 클래스와 메서드를 몇 가지만이라도 계측하는 것이 좋습니다. 이 단원에서는 ProbeBuilder 지시문 키워드로 작업하고 사용자 지정 PBD 파일을 만드는 작업을 소개합니다.

중요! PBD 와 PBL 은 ASCII 문자만 지원합니다. PBD 또는 PBL 에 유니코드 문자 같은 다른 문자를 사용하면 문제가 발생할 수 있습니다.

이 섹션은 다음 항목을 포함하고 있습니다.

기존 추적 프로그램 그룹에 클래스 추가

기존 추적 프로그램 그룹에 클래스를 추가하여 특정 클래스에 대한 추적을 설정할 수 있습니다. 클래스를 추적 프로그램 그룹의 일부로 식별하려면 식별 키워드 중 하나를 사용하십시오.

예를 들어 *System.EnterpriseServices.ServicedComponent* 클래스를 추적 프로그램 그룹 *ServicedComponentTracing* 에 추가하려면 다음과 같이 지정하십시오.

```
IdentifyClassAs:
System.EnterpriseServices.ServicedComponent ServicedComponentTracing
```

사용자 지정 추적 프로그램 생성

사용자 지정 .pbd 파일을 생성하여 메트릭 수집을 보다 세부적으로 조정할 수 있습니다. 응용 프로그램별 측정을 추적하는 추적 프로그램을 생성하여 사용자 지정 지시문을 생성하려면 특정 구문 및 키워드를 사용해야 합니다.

사용자 지정 추적 프로그램을 작성하려면 다음을 정의해야 합니다.

- 지시문 유형(일반적으로 추적할 클래스 또는 메서드 수를 나타냄)
- 추적할 특정 클래스 또는 메서드
- 클래스 또는 메서드에서 추적할 정보 유형(예: 시간, 속도 또는 개수)
- 이 정보를 나타낼 정규화된 메트릭 이름(리소스 경로 포함)

사용자 지정 .pbd 가 생성되면 Introscope 는 이를 기본 제공 .pbd 인 것처럼 처리합니다. 생성된 메트릭에 대한 경고를 설정하거나, 이 메트릭을 SmartStor 에 저장하거나, Introscope Workstation 에서 사용자 지정 대시보드를 생성할 때 이 메트릭을 사용할 수 있습니다.

참고: 추적되는 메서드가 많을수록 오버헤드가 증가하므로 추적할 메서드는 신중하게 선택해야 합니다.

이 단원에서 다루는 항목

공통 사용자 지정 추적 프로그램 예제

BlamePointTracer 는 가장 많이 사용되는 추적 프로그램입니다. 이 추적 프로그램은 연결된 메서드나 클래스에 대해 다음과 같은 다섯 개의 개별 메트릭을 생성합니다.

- 평균 응답 시간(ms)
- 동시 호출
- 간격당 오류 수
- 간격당 응답 수
- 중단 수

다음은 *BlamePointTracer* 의 예제입니다. *BlamePointTracer* 가 "petshop.catalog.Catalog" 클래스에서 "search"라는 이름의 메서드에 대해 설정되었습니다. "MSPetShop|Catalog|search"는 BlamePoint 메트릭이 표시될 Introscope Investigator 의 노드 이름입니다.

```
TraceOneMethodOfClass: petshop.catalog.Catalog search BlamePointTracer
"MSPetShop|Catalog|search"
```

추적 프로그램 구문

PBD 파일에는 추적 프로그램을 그룹에 연결하거나 그룹이 사용 또는 사용되지 않도록 설정하는 단순 키워드 외에 추적 프로그램 정의도 포함됩니다. Introscope 가 추적 프로그램을 인식하고 처리하도록 하려면 사용자 지정 추적 프로그램을 구성할 때 특정 구문을 사용해야 합니다. 추적 프로그램은 지시문과 추적할 메서드 또는 클래스에 대한 정보로 구성되며 형식은 다음과 같습니다.

```
<directive>: [arguments]
```

여기서 *[arguments]*는 목록으로, 지시문에 따라 다릅니다. 추적 지시문에 사용되는 인수에는 *<Tracer-Group>*, *GHOST*, *<method>*, *<Tracer-name>* 및 *<metric-name>*이 있습니다.

참고: 사용되는 지시문에 따라 이러한 매개 변수 중 일부만이 필요합니다.

<directive>

사용자 지정 추적에 사용할 수 있는 6 개의 기본 지시문이 있습니다.

- **TraceOneMethodOfClass** - 지정된 클래스의 지정된 메서드를 추적합니다.
- **TraceAllMethodsOfClass** - 지정된 클래스의 모든 메서드를 추적합니다.
- **TraceOneMethodIfInherits** - 지정된 클래스나 인터페이스의 모든 직접 서브클래스 또는 직접 인터페이스 구현에서 메서드 하나를 추적합니다.
- **TraceAllMethodsIfInherits** - 지정된 클래스나 인터페이스의 모든 직접 서브클래스 또는 직접 인터페이스 구현에 포함된 모든 메서드를 추적합니다.

참고: 구체적으로 구현된 메서드만이 추적 가능하며 실행 중에 메트릭 데이터를 보고할 수 있습니다. 사용자 지정 추적 프로그램에 추상 메서드를 지정하면 메트릭 데이터가 보고되지 않습니다.

- **TraceOneMethodIfFlagged** - 지정된 클래스가 *TurnOn* 키워드로 사용하도록 설정된 추적 프로그램 그룹에 포함된 경우 메서드 하나를 추적합니다.
- **TraceAllMethodsIfFlagged** - 지정된 클래스가 *TurnOn* 키워드로 사용하도록 설정된 추적 프로그램 그룹에 포함된 경우 모든 메서드를 추적합니다.

<Tracer-Group>

추적 프로그램을 연결할 그룹입니다.

GHOST

추적할 클래스 또는 인터페이스의 정규화된 이름입니다. 정규화된 클래스 이름에는 다음과 같이 클래스 이름뿐 아니라 클래스의 전체 어셈블리 이름도 포함됩니다.

```
[MyAssembly]com.mycompany.myassembly.MyClass
```

참고: 어셈블리 이름은 대괄호([])로 묶어야 합니다.

<method>

- 메서드 이름(예: *MyMethod*)

또는

- 반환 유형 및 매개 변수가 포함된 전체 메서드 서명(예: *myMethod;[mscorlib]System.Void([mscorlib]System.Int32)* 메서드 서명에 대한 자세한 내용은 서명 구별 (페이지 99)을 참조하십시오.

<Tracer-name>

사용할 추적 프로그램 유형을 지정합니다. 예를 들어 *BlamePointTracer* 입니다. 추적 프로그램 이름과 추적 프로그램이 추적하는 항목에 대한 설명은 추적 프로그램 이름 (페이지 95)을 참조하십시오.

<metric-name>

수집된 데이터가 Introscope Workstation 에 표시되는 방식을 제어합니다.

다음 예에서는 메트릭 트리의 서로 다른 수준에서 메트릭 이름 및 위치를 지정하는 세 가지 방법에 대해 설명합니다.

- **metric-name** - 메트릭이 에이전트 노드 내에 바로 표시됩니다.
- **resource:metric-name** - 메트릭이 에이전트 노드에서 한 수준 아래의 리소스(폴더) 내에 표시됩니다.
- **resource|sub-resource|sub-sub-resource:metric-name** - 메트릭이 에이전트 노드에서 두 수준 이상 아래의 리소스(폴더)에 표시됩니다. 파이프 문자(|)를 사용하여 리소스를 구분합니다.

추적 프로그램 이름

다음은 추적 프로그램 이름과 추적 프로그램이 추적하는 항목입니다.

BlamePointTracer

Blame 관련 구성 요소의 평균 응답 시간, 간격당 수, 동시성, 중단 및 오류를 포함하는 표준 메트릭 집합을 제공합니다.

ConcurrentInvocationCounter

시작되었지만 아직 완료되지 않은 메서드의 횟수를 보고합니다. 결과는 Investigator 트리에서 추적 프로그램에 지정된 메트릭 이름 *<metric-name>* 아래에 보고됩니다. 예를 들어 이 추적 프로그램을 사용하여 동시 데이터베이스 쿼리의 수를 셀 수 있습니다.

DumpStackTraceTracer

이 추적 프로그램은 적용되는 메서드에 대해 계측된 응용 프로그램의 표준 오류로 스택 추적을 덤프합니다. 덤프 스택 추적 프로그램에서 throw 되는 예외 스택 추적은 실제 예외가 아니라 메서드 스택 추적을 출력하기 위한 메커니즘입니다.

이 기능은 메서드의 호출 경로를 확인하는 데 유용합니다.

중요! 이 기능은 과다한 시스템 오버헤드를 발생시킵니다. 따라서 이 추적 프로그램은 급격한 오버헤드 증가가 허용되는 진단 컨텍스트에서만 사용하는 것이 좋습니다.

MethodTimer

평균 메서드 실행 시간(밀리초)으로, 메트릭 트리에서 추적 프로그램에 지정된 메트릭 이름 *<metric-name>* 아래에 보고됩니다.

PerIntervalCounter

간격당 호출 수입니다. 이 간격은 데이터 소비자(예: Investigator 의 뷰 창)의 표시 기간에 따라 변경됩니다. 결과는 Investigator 트리에서 추적 프로그램에 지정된 메트릭 이름 *<metric-name>* 아래에 보고됩니다.

사용자 지정 메서드 추적 프로그램 예제

사용자 지정 추적 프로그램에는 공백이 있는 메트릭 이름이 포함될 수 있습니다. 사용자 지정 메트릭 이름에 공백을 사용할 경우에는 모든 메트릭 이름을 따옴표("")로 묶는 것이 좋습니다.

중요! *클래스* 이름은 따옴표로 묶지 마십시오. 사용자 지정 추적 프로그램이 오작동하는 원인이 될 수 있습니다. 예:

올바른 이름

IdentifyClassAs: My.Name.Space.MyClass MyTracers

올바르지 않은 이름

IdentifyClassAs: "My.Name.Space.MyClass" MyTracers

클래스 이름이 포함된 메트릭 이름을 생성하는 경우에는 전체 메트릭 이름을 따옴표로 묶어야 합니다. 메트릭 이름에 공백을 사용할 수 있으며, 이 경우 메트릭 이름의 모든 공백은 따옴표 내에 포함되어야 합니다. 예를 들어 메트릭 이름 "{classname}|Test One Node"는 다음과 같이 나타내야 합니다.

올바른 이름

TraceOneMethodIfFlagged: MyTracers AMethod BlamePointTracer "{classname}|Test One Node"

올바르지 않은 이름

TraceOneMethodIfFlagged: MyTracers AMethod BlamePointTracer {classname}|Test One Node

다음 단원에서는 메서드 추적 프로그램의 예를 보여 줍니다. 다음 예에서는 메트릭 이름을 따옴표("")로 묶었습니다. 사용자 지정 메트릭 이름을 생성할 때는 모든 메트릭 이름을 따옴표로 묶는 것이 좋습니다.

평균 추적 프로그램 예제

이 추적 프로그램은 지정된 메서드의 평균 실행 시간을 밀리초 단위로 추적합니다.

```
TraceOneMethodOfClass: petshop.catalog.Catalog search BlamedMethodTimer
"MSPetShop|Catalog|search:Average Method Invocation Time (ms)"
```

속도 추적 프로그램 예제

이 추적 프로그램은 초당 메서드 호출 횟수를 계산하여 지정된 메트릭 이름 아래에 이 속도를 보고합니다.

```
TraceOneMethodOfClass: petshop.catalog.Catalog search BlamedMethodRateTracer
"MSPetShop|Catalog|search:Method Invocations Per Second"
```

간격당 카운터 추적 프로그램 예제

이 메서드 추적 프로그램은 간격당 메서드 호출 횟수를 계산하여 지정된 메트릭 이름 아래에 이 간격당 수를 보고합니다.

```
TraceOneMethodOfClass: petshop.catalog.Catalog search PerIntervalCounter
"MSPetShop|Catalog|search:Method Invocations Per Interval"
```

이 간격은 그래프 빈도와 같은 Enterprise Manager 의 모니터링 로직에 의해 결정됩니다.

Introscope Investigator 에서 미리 보기 창은 기본적으로 15 초 간격입니다.

카운터 추적 프로그램 예제

이 추적 프로그램은 총 메서드 호출 횟수를 계산합니다.

```
TraceOneMethodOfClass: petshop.cart.ShoppingCart placeOrder
BlamedMethodTraceIncrementor "MSPetShop|ShoppingCart|placeOrder:Total Order Count"
```

결합된 카운터 추적 프로그램 예제

이러한 추적 프로그램은 수를 계산하기 위해 incrementor 및 decrementor 추적 프로그램을 결합합니다.

```
TraceOneMethodOfClass: petshop.account.LoginComponent login MethodTraceIncrementor
"MSPetShop|Account:Logged In Users"
TraceOneMethodOfClass: petshop.account.LogoutComponent logout
MethodTraceDecrementor "MSPetShop |Account:Logged In Users"
```

고급 사용자 지정 추적 프로그램 만들기

다음 단원에는 단일 메트릭 추적 프로그램, 건너뛰기, 결합된 사용자 지정 추적 프로그램 등 고급 사용자 지정 추적 프로그램을 생성하는 방법이 자세히 나와 있습니다.

이 단원에서 다루는 항목

고급 단일 메트릭 추적 프로그램

지시문 및 추적 프로그램은 메서드, 클래스 및 클래스 집합을 추적합니다. Introscope 가 추적할 수 있는 최소 단위는 메서드이고, 단일 메트릭 추적 프로그램은 특정 메서드에 대한 특정 메트릭을 보고합니다. 메서드 서명을 사용하거나, 키워드를 대체하거나, 메트릭 이름 매개 변수를 조작하는 등 여러 방법으로 단일 메트릭 추적 프로그램을 만들 수 있습니다.

서명 구별

메서드 서명을 기준으로 메서드에 추적 프로그램을 적용할 수 있습니다.

특정 서명으로 메서드의 단일 인스턴스를 추적하려면 내부 메서드 설명자 형식을 사용하여 지정된 메서드 이름에 서명을 추가합니다(반환 유형 포함).

예를 들어 *myMethod;[mscorlib]System.Void([mscorlib]System.Int32)*는 *int* 인수와 *void* 반환 유형을 사용하는 메서드의 인스턴스를 추적합니다.

메트릭 이름 키워드 기반 대체

키워드 기반 대체를 사용하면 런타임에 값을 메트릭 이름으로 대체할 수 있습니다.

추적 프로그램에서 메트릭 이름의 매개 변수가 런타임에 실제 값으로 대체되어 메트릭 이름이 됩니다. 이 기능은 모든 지시문에서 사용할 수 있습니다. 다음 표에는 매개 변수와 해당 런타임 대체가 나와 있습니다.

매개 변수	런타임 대체
{method}	추적할 메서드 이름
{classname}	추적할 클래스의 런타임 클래스 이름
{namespacename}	추적할 클래스의 런타임 네임스페이스 이름
{namespaceandclassname}	추적할 클래스의 런타임 네임스페이스 및 클래스 이름
{assemblyname}	추적할 어셈블리 이름
{fullclassname}	어셈블리 이름을 포함한 전체 클래스 이름을 보고합니다.

참고: Introscope 가 *네임스페이스*가 없는 클래스를 처리하는 경우 *{namespacename}*을 문자열 *"<Unnamed Namespace>"*로 대체합니다.

키워드 기반 대체: 예제 1

.pbd 파일에서 추적 프로그램의 메트릭 이름이 다음과 같다고 가정합니다.

```
"{namespacename}|{classname}|{method}:Response Time (ms)"
```

그리고 추적 프로그램이 *myNamespace* 에 있는 *myClass* 의 런타임 클래스가 있는 메서드 *myMethod* 에 적용되는 경우 결과 메트릭 이름은 다음과 같이 됩니다.

```
"myNamespace|myClass|myMethod:Response Time (ms)"
```

키워드 기반 대체: 예제 2

.pbd 파일에서 다음과 같은 메트릭 이름을 갖는 추적 프로그램이 있다고 가정합니다.

```
"{namespaceandclassname}|{method}:Response Time (ms)"
```

이 추적 프로그램을 예제 1 과 동일한 메서드에 적용할 경우 결과 메트릭 이름은 다음과 같이 됩니다.

```
"myNamespace.myClass|myMethod:Response Time(ms)"
```

참고: 이 예제에서는 *네임스페이스*와 *클래스* 사이에 첫 번째 예제의 |(파이프 기호) 대신 마침표(.)가 사용되었습니다.

메트릭 이름 기반 매개 변수

메트릭 이름을 생성하는 단일 메서드 추적 프로그램을 생성할 수 있습니다. 이 메트릭 이름은 TraceOneMethodWithParametersofClass 키워드를 사용하여 다음 형식으로 메서드에 전달된 매개 변수를 기반으로 합니다.

```
TraceOneMethodWithParametersOfClass: GHOST <method> <Tracer-name> <metric-name>
```

메트릭 이름에 매개 변수를 사용할 수 있습니다. 메트릭 이름의 자리 표시자 문자열을 매개 변수의 값으로 대체합니다. "{#}" 문자열을 자리 표시자로 사용합니다. 여기서 #은 대체할 매개 변수의 인덱스입니다. 인덱스는 0 부터 시작됩니다. 대체 매개 변수는 원하는 수만큼 원하는 순서대로 사용할 수 있습니다. 모든 매개 변수는 메트릭 이름으로 대체되기 전에 문자열로 변환됩니다. 문자열 이외의 개체 매개 변수는 ToString() 메서드를 사용하여 변환되므로 주의해서 사용해야 합니다.

중요! 매개 변수가 어떤 문자열로 변환될지 확실히 알 수 없는 경우에는 매개 변수를 메트릭 이름에 사용하면 안 됩니다.

메트릭 이름 기반 예제

웹 사이트에서는 *order* 라는 클래스와 *process* 라는 메서드를 사용합니다. 이 메서드에는 서로 다른 종류의 주문(*책* 또는 *음악*)에 대한 매개 변수가 있습니다.

다음과 같은 추적 프로그램을 생성할 수 있습니다.

```
TraceOneMethodWithParametersOfClass: order process;
[mscorlib]System.Void([mscorlib]System.Int32) MethodTimer "Order|{0}Order:Average
Response Time (ms)"
```

이 추적 프로그램은 다음과 같은 메트릭을 생성합니다.

```
Order
    BookOrder
        평균 응답 시간(ms)
    MusicOrder
        평균 응답 시간(ms)
```

TraceOneMethodWithParametersIfInherits 키워드를 사용할 수도 있습니다.

건너뛰기 지시문

건너뛰기 지시문을 사용하여 특정 패키지, 클래스 또는 메서드를 계측하지 않도록 설정할 수 있습니다. 건너뛰기 지시문을 사용하면 ProbeBuilder 가 네임스페이스, 클래스 또는 어셈블리를 건너뜁니다.

일치 비율을 낮게 설정하여 계측되는 패키지, 클래스 또는 메서드의 양을 제한할 수도 있습니다. 그러기 위해서는 *IdentifyAllClassAs* 및/또는 *TraceAllMethodsIfFlagged* 지시문을 더 좁은 범위의 값으로 대체해야 합니다.

사용자 지정 추적 프로그램 결합

동일한 메트릭에 적용되는 여러 추적 프로그램을 사용하여 결합 효과를 낼 수 있습니다. 이러한 결합은 대개 incrementor 및 decrementor 추적 프로그램과 함께 사용됩니다.

이 예제에서는 *Logged-in users* 라는 이름의 메트릭을 생성합니다. 클래스 *user* 와 메서드 *login* 및 *logout* 을 사용하여 다음 추적 프로그램을 생성하십시오.

```
TraceOneMethodOfClass user login MethodTraceIncrementor "Logged-in Users"
TraceOneMethodOfClass user logout MethodTraceDecrementor "Logged-in Users"
```

이 추적 프로그램은 사용자가 로그인하면 *Logged-in Users* 메트릭을 증가시키고, 사용자가 로그아웃하면 *Logged-in Users* 메트릭을 감소시킵니다.

특정 추적 프로그램에 대한 참고 사항

다음 식별자와 추적 프로그램에는 .NET 환경과 관련된 작업이 있습니다.

- *IdentifyAnnotatedClassAs*: <attribute-class-name> <Tracer-group>

  지정된 특성 클래스로 주석 처리된 모든 클래스를 지정된 추적 프로그램 그룹에 연결합니다.

  일부 클래스는 클래스에 대한 추가적인 기능을 제공하기 위해 특성 클래스로 주석 처리될 수 있습니다. 아래 예제에서는 *System.EnterpriseServices.Transaction* 이라는 특성 클래스가 *ServicedComponent* 클래스에 추가되었습니다.

  ```
  [Transaction]
  Public class ServicedComponent {
  }
  ```

  .pbd 파일에서 다음과 같이 작성할 수 있습니다.

  ```
  IdentifyAnnotatedClassAs: System.EnterpriseServices.MyTracerGroup
  ```

  이 코드는 *ServicedComponent* 를 포함한 *[Transaction]* 어노테이션이 있는 모든 클래스를 식별합니다.

참고: 이 식별자를 사용하면 Introscope .NET 에이전트가 상속된 특성을 추적하지 않고 기본 클래스에 적용된 특성을 추적합니다.

- *TraceAnnotatedMethodsIfFlagged*: <Tracer-group> <attribute-class-name> <Tracer-name> <metric-name>

  지정된 추적 프로그램 그룹과 연결된 클래스에 대해 지정된 클래스로 주석 처리된 모든 메서드를 추적합니다.

명시적 인터페이스 구현

.NET 에이전트는 .pbd 파일에서 명시적 인터페이스 구현을 사용합니다. 클래스의 메서드를 추적할 때 다른 클래스에 이름이 같은 메서드가 사용되는 경우 메서드와 해당 메서드가 속한 인터페이스의 이름을 명시적으로 지정해야 합니다. 예를 들어 *InterfaceA* 및 *InterfaceB* 에 모두 *MethodX* 라는 메서드가 있는 경우 *InterfaceA* 의 *MethodX* 를 호출할 때 *InterfaceA.MethodX* 와 같이 인터페이스와 메서드의 이름을 모두 지정해야 합니다.

다음은 명시적 인터페이스를 구현하여 클래스의 메서드를 추적하는 예제입니다.

```
SetFlag: customInterfaceTracing
TurnOn: customInterfaceTracing
IdentifyInheritedAs: EdgeCaseInterface customInterfaceTracing
TraceOneMethodIfFlagged: customInterfaceTracing EdgeCaseInterface.method2
BlamePointTracer "Interface|{namespaceandclassname}|{method}"
```

계측 및 상속

Introscope 는 클래스 계층에서 더 깊은 수준에 있는 클래스를 자동으로 계측하지 않습니다. 예를 들어 다음과 같이 *ClassB* 가 *ClassA* 를 확장하고 *ClassC* 가 *ClassB* 를 확장하는 클래스 계층 구조가 있다고 가정합니다.

```
Interface\ClassA
    ClassB
        ClassC
```

ClassA 를 계측하면 *ClassB* 는 *ClassA* 를 명시적으로 확장하므로 함께 계측됩니다. 그러나 *ClassC* 는 *ClassA* 를 명시적으로 확장하지 않으므로 *ClassC* 는 계측되지 *않습니다*. *ClassC* 를 계측하려면 *ClassC* 를 명시적으로 식별해야 합니다.

ProbeBuilder 지시문 적용

ProbeBuilder 지시문 파일을 구현할 준비를 마쳤으면 다음 세 가지 방법 중 하나를 사용하여 새 파일을 구현할 수 있습니다.

hotdeploy 디렉터리 사용

hotdeploy 디렉터리를 사용하면 Introscope 관리자가 *IntroscopeAgent.profile* 을 편집하거나 응용 프로그램을 재시작하지 않고도 새 지시문을 보다 빠르고 손쉽게 배포할 수 있습니다. 그러나 이 기능을 사용할 때는 상당한 주의가 요구됩니다. 사용자 지정 PBD 파일이 잘못된 구문을 포함하거나 메트릭을 너무 많이 수집하도록 구성되어 있으면 그 영향이 더 빠르게 나타납니다. 잘못된 PBD 를 사용하면 NativeProfiler 가 종료될 수 있으며 PBD 가 너무 많은 메트릭을 수집하는 경우에는 응용 프로그램 성능이 저하될 수 있습니다. 이러한 문제를 해결하기 위해 다음과 같이 하는 것이 좋습니다.

- 모든 지시문을 QA 및 성능 환경에서 테스트하고 검증한 이후에 프로덕션 환경에서 사용합니다.
- 새로운 PBD 배포 옵션을 반영하도록 서버 환경의 변경 제어 프로세스를 업데이트합니다.

새 PBD 를 hotdeploy 디렉터리에 배치하면 .NET 에이전트가 해당 PBD 를 자동으로 배포합니다. 그러나 이미 실행 중인 클래스와 응용 프로그램의 경우에는 해당 응용 프로그램을 재시작해야만 새 PBD 나 변경된 PBD 의 내용이 적용됩니다. 새 PBD 를 이 디렉터리에 배치하면 새 PBD 나 변경된 PBD 를 적용하기 위해 IntroscopeAgent.profile 을 편집하지 않아도 됩니다.

hotdeploy 디렉터리를 사용하여 .pbd 를 적용하려면

- 사용자 지정 파일 또는 수정한 파일(.pbd 및 .pbl)을 *<Agent_Home>\wily\hotdeploy* 디렉터리에 복사합니다.

<Agent_Home>/wily 디렉터리 사용

새로 만들거나 변경한 PBD 와 PBL 을 배포하려면 해당 파일을 *introscope.autoprobe.directivesFile* 속성에 포함하고 *IntroscopeAgent.profile* 파일과 같은 디렉터리 또는 *IntroscopeAgent.profile* 파일의 위치에 상대적인 디렉터리에 배치해야 합니다.

그 외의 디렉터리에 파일을 둘 경우에는 *introscope.autoprobe.directivesFile* 속성을 설정할 때 파일의 전체 경로를 지정해야 합니다.

새로 만들거나 변경한 .pbd 및 .pbl 파일을 배포하려면

1. 사용자 지정 파일 또는 수정한 파일(PBD 및 PBL)을 *<Agent_Home>* 디렉터리에 복사합니다.
2. 새 파일의 이름을 쉼표로 구분하여 포함하도록 *IntroscopeAgent.profile* 파일의 *introscope.autoprobe.directivesFile* 속성을 업데이트합니다.

   예를 들어 사용자 지정 *petstore.pbd* 파일을 다음과 같이 속성에 추가합니다.

   ```
   introscope.autoprobe.directivesFile=default-full.pbl,petstore.pbd,hotdeploy
   ```
3. *IntroscopeAgent.profile* 을 저장하고 닫습니다.
4. 응용 프로그램 또는 IIS 서비스를 다시 시작합니다.

참고: 기존 .pbl 또는 .pbd 파일을 통해 제어되는 모니터링 기능을 해제하려는 경우가 아니면 이 속성에 정의된 기존 *.pbl* 또는 *.pbd* 파일을 제거하지 마십시오.

사용자 지정 위치 및 권한

위에 설명되어 있는 것처럼 *hotdeploy* 디렉터리 또는 *wily* 디렉터리를 사용하는 것 외에, 이 두 디렉터리 중 하나에 없는 선택한 사용자 지정 위치에 PBD 를 저장할 수 있습니다.

.pbd 파일을 사용자 지정 위치에 배치하는 경우 *IntrosocopeAgent.profile* 에서 .pbd 파일 위치를 지정해야 합니다. 예를 들어 *C:* 드라이브의 사용자 지정 위치에 *leakhunter.pbd* 를 배치한 경우 다음과 같은 방법으로 *introscope.autoprobe.directivesFile* 속성을 업데이트할 수 있습니다.

```
introscope.autoprobe.directivesFile=default-full.pbl,C:\\sw\\leakhunter.pbd
```

사용자 지정 위치에 .pbd 를 저장한 경우 IIS 프로세스를 시작하는 사용자에게 해당 사용자 지정 위치(위 예제의 경우 *C:\\sw*)에 대한 적절한 권한이 있어야 합니다. IIS 프로세스를 시작하는 사용자에게 이 위치에 대한 사용 권한이 없는 경우 기본 도메인 로그에 오류 메시지가 보고되고 사용자 지정 위치에 있는 .pbd 는 적용되지 않습니다.

중요! CA Technologies 는 PBD 를 *hotdeploy* 디렉터리에 배치할 것을 적극 권장합니다.

트랜잭션 추적 및 동적 계측

Introscope Workstation 에서 트랜잭션 추적을 실행할 경우 동적 계측을 수행하도록 선택할 수 있습니다. 동적 계측을 사용하면 PBD 파일을 만들거나 에이전트 프로필을 수정하지 않고 런타임에 계측 대상 메서드를 하나 이상 선택할 수 있습니다. 동적 계측을 사용하면 다음과 같은 작업을 수행할 수 있습니다.

- 트랜잭션 구성 요소에 속한 호출된 메서드 하나, 둘 이상 또는 모두 보기 및 계측.
- 임시 계측된 메서드에 대한 추적 뷰.
- 임시 계측된 메서드에 대해 수집된 메트릭 보기.
- 임시 계측을 영구 계측으로 저장.

동적 계측은 CPU 를 많이 사용하는 작업입니다. 따라서 계측되는 클래스를 최소화하는 구성을 사용하는 것이 좋습니다.

참고: .NET 운영 환경에서는 기능이 제한적으로 지원됩니다. 자세한 내용은 *CA APM Workstation 사용자 안내서*에서 "트랜잭션 추적"을 참조하십시오.

추가 정보:

동적 계측 (페이지 217)

제 6 장: LeakHunter 구성

Introscope LeakHunter 는 시간이 지남에 따라 크기가 늘어나는 컬렉션 인스턴스(즉, 컬렉션에 저장된 개체 수가 시간이 지남에 따라 늘어나는 경우)를 조사하여 잠재 메모리 누수의 원인을 파악할 수 있도록 설계된 추가 기능 구성 요소입니다.

몇 분 또는 몇 시간 정도로 짧게 실행되는 프로그램에서 발생하는 메모리 누수는 큰 문제가 되지 않을 수 있습니다. 그러나 웹 사이트와 같이 하루 24 시간 실행되는 응용 프로그램의 경우에는 사소한 메모리 누수라도 곧바로 큰 문제로 확대될 수 있습니다.

Introscope LeakHunter 를 설정하고 컬렉션 클래스를 검색한 다음 정보가 수집된 후에는 Introscope LeakHunter 를 해제하는 방법으로 발견된 메모리 누수에 대한 정보를 추적할 수 있습니다. 이 방법으로 LeakHunter 를 사용하면 일시적으로만 약간의 오버헤드가 발생합니다.

이 섹션은 다음 항목을 포함하고 있습니다.

LeakHunter 작동 방식

LeakHunter 를 사용하도록 설정한 후 LeakHunter 가 새 잠재 누수를 찾는 동안 적용되는 시간 초과 기간을 정의할 수도 있습니다. 그런 후에 관리되는 응용 프로그램을 다시 시작합니다.

LeakHunter 는 시간에 지남에 따라 크기가 증가하는 컬렉션을 찾은 경우 다음 작업을 수행합니다.

- 컬렉션을 고유 ID 로 식별합니다.
- 컬렉션에 대한 정보를 Enterprise Manager 에 메트릭 데이터로 보고합니다.
- 컬렉션에 대한 정보를 에이전트 컴퓨터의 로그 파일에 보고합니다.
- 추적을 계속하여 해당 컬렉션에 대한 데이터를 보고합니다.

LeakHunter 는 컬렉션에 더 이상 누수가 없는 것으로 확인될 경우 해당 사실을 Enterprise Manager 와 로그 파일 모두에 보고하지만, 해당 컬렉션에 대한 추적과 데이터 보고는 계속합니다.

LeakHunter 는 시간 초과 기간이 만료될 때까지 계속해서 잠재 누수를 찾고 이미 식별된 잠재 누수를 모니터링합니다. 시간 초과 기간이 만료되면 LeakHunter 는 새로 할당된 컬렉션에서 잠재 누수 찾기를 중지하고 이미 잠재 누수로 식별된 컬렉션만 계속 검사합니다. 따라서 LeakHunter 오버헤드가 크게 줄어들며 잠재 누수를 추가적으로 모니터링할 수 있습니다. LeakHunter 는 관리되는 응용 프로그램이 종료될 때까지 식별된 잠재 누수를 계속 모니터링합니다.

메모리 누수의 원인을 찾으려면 Introscope Investigator 에서 메트릭 데이터를 탐색하거나 로그 파일을 확인하면 됩니다.

LeakHunter 가 .NET 에서 추적하는 항목

다음과 같은 .NET 특정 컬렉션을 추적합니다.

단원 정보

특정 컬렉션 (페이지 109)
특정 인터페이스 (페이지 109)
일반 컬렉션 (페이지 110)
일반 인터페이스 (페이지 111)

특정 컬렉션

- System.Collections.ArrayList
- System.Collections.BitArray
- System.Collections.CollectionBase
- System.Collections.DictionaryBase
- System.Collections.Hashtable
- System.Collections.Queue
- System.Collections.SortedList
- System.Collections.Stack
- System.Collections.Specialized.HybridDictionary
- System.Collections.Specialized.ListDictionary
- System.Collections.Specialized.NameObjectCollectionBase
- System.Collections.Specialized.NameValueCollection
- System.Collections.Specialized.StringCollection
- System.Collections.Specialized.StringDictionary
- System.Collections.Specialized.OrderedDictionary

특정 인터페이스

- System.Collections.ICollection
- System.Collections.IDictionary
- System.Collections.IList
- System.Collections.Specialized.IOrderedDictionary

LeakHunter 는 System.Collections 에서 .NET IList, ICollection 및 IDictionary 구현을 추적합니다.

ICollection, IDictionary 및 IList 에 대한 자세한 내용은 다음을 참조하십시오.

- IList 구현: http://msdn2.microsoft.com/en-us/library/system.collections.ilist(VS.71).aspx
- ICollection 구현: http://msdn2.microsoft.com/en-us/library/system.collections.icollection(VS.71).aspx
- IDictionary 구현: http://msdn.microsoft.com/en-us/library/system.collections.idictionary%28VS.71%29.aspx

LeakHunter 는 System.Collections.Specialized 의 IOrderedDictionary 인스턴스도 추적할 수 있습니다. 그러나 .NET Framework 에는 이 항목에 대한 구현이 없으므로 LeakHunter 는 IOrderedDictionary 응용 프로그램 구현만 추적합니다.

일반 컬렉션

- System.Collections.Generic.List
- System.Collections.Generic.SortedList
- System.Collections.Generic.Dictionary
- System.Collections.Generic.SortedDictionary
- System.Collections.Generic.LinkedList
- System.Collections.Generic.Queue
- System.Collections.Generic.Stack
- System.Collections.Generic.SynchronizedKeyedCollection
- System.Collections.Generic.SynchronizedCollection
- System.Collections.Generic.KeyedByTypeCollection
- System.Collections.Generic.HashSet
- System.Collections.ObjectModel.KeyedCollection
- System.Collections.ObjectModel.Collection
- System.Collections.ObjectModel.ObservableCollection

일반 인터페이스

- System.Collections.Generic.IList
- System.Collections.Generic.ICollection
- System.Collections.Generic.IDictionary

LeakHunter 가 추적하지 않는 항목

Introscope LeakHunter 는 다음을 추적하지 않습니다.

- 컬렉션에 의해 발생하지 않은 누수
- 참조 수가 증가하는 사용자 지정 컬렉션 구현 또는 기타 데이터 구조
- 계측되지 않는 누수 컬렉션

LeakHunter 를 사용 또는 사용하지 않도록 설정

LeakHunter 는 에이전트 확장으로 실행되므로 클래스 경로를 업데이트할 필요가 없습니다. 기본적으로 LeakHunter 는 설치 후 사용하도록 설정되지 않습니다. 따라서 해당 기능을 사용하려면 LeakHunter 를 사용하도록 설정해야 합니다.

LeakHunter 를 사용하려면

1. 에이전트 프로필 *IntroscopeAgent.profile* 을 엽니다.
2. *LeakHunter Configuration* 제목 아래에서 *introscope.agent.leakhunter.enable* 속성을 찾아 값을 *true* 로 입력합니다.
3. 에이전트 프로필을 저장합니다.
4. *IntroscopeAgent.profile* 속성인 *introscope.autoprobe.directivesFile* 에 나열한 **.typical.pbl* 또는 **.full.pbl* 파일을 엽니다.
5. *leakhunter.pbd* 의 주석 처리를 제거합니다.

6. **.typical.pbl* 또는 **.full.pbl* 파일을 저장하고 닫습니다.
7. 응용 프로그램을 다시 시작합니다.

참고: 기본적으로 LeakHunter 와 같은 에이전트 확장 기능은 *<Agent_Home>\ext* 디렉터리에 위치하며 이 디렉터리에서 참조됩니다. 그러나 에이전트 프로필에서 에이전트 확장 디렉터리의 위치를 변경할 수 있습니다. 확장 기능 디렉터리의 위치를 변경할 때는 디렉터리의 콘텐츠도 함께 이동해야 합니다.

LeakHunter 를 사용하지 않으려면

1. 에이전트 프로필 *IntroscopeAgent.profile* 을 엽니다.
2. *LeakHunter Configuration* 제목 아래에서 *introscope.agent.leakhunter.enable* 속성을 찾아 값을 *false* 로 입력합니다.
3. 에이전트 프로필을 저장합니다.
4. 응용 프로그램을 다시 시작합니다.

LeakHunter 속성 구성

LeakHunter 구성 속성은 *<Agent_Home>/wily* 디렉터리의 *IntroscopeAgent.profile* 에이전트 프로필에 있습니다.

LeakHunter 를 구성하려면

1. 에이전트 프로필 *IntroscopeAgent.profile* 을 엽니다.
2. 다음 LeakHunter 속성을 원하는 대로 구성합니다.
   - introscope.agent.leakhunter.logfile.location
   - introscope.agent.leakhunter.logfile.append
   - introscope.agent.leakhunter.leakSensitivity
   - introscope.agent.leakhunter.timeoutInMinutes
   - introscope.agent.leakhunter.collectAllocationStackTraces
   - introscope.agent.leakhunter.ignore.0

   LeakHunter 속성에 대한 자세한 내용은 LeakHunter 속성 (페이지 223)을 참조하십시오.

   참고: *IntroscopeAgent.profile* 에는 LeakHunter 에서 무시되는 패키지를 제어하는 기본 속성이 포함되어 있습니다. 대부분의 경우 이러한 속성은 성능 저하의 원인이 되는 컬렉션을 무시하도록 구성됩니다. 이러한 속성은 기본적으로 사용되도록 설정되어 있습니다. 이러한 속성을 주석으로 처리할 경우 예외가 에이전트 로그에 보고되지 않습니다.

성능 저하의 원인이 되는 컬렉션 무시

컬렉션의 size() 메서드가 수집의 개체 수에 비례하는 시간 동안 실행되면 성능이 저하됩니다. 즉, 컬렉션의 size() 메서드가 실행되는 시간이 오래 걸릴수록(예: 잘못 구현된 LinkedList 에서 목록의 각 요소를 트래버스하여 목록의 크기 및 요소의 수를 가져오는 경우) 응용 프로그램 성능에 부정적인 영향을 줍니다.

이러한 컬렉션은 IntroscopeAgent.profile 의 무시 속성을 사용하여 무시되어야 합니다.

LeakHunter 실행

LeakHunter 는 에이전트 확장으로 실행됩니다.

LeakHunter 를 실행하려면

1. *IntroscopeAgent.profile* 을 텍스트 편집기에서 엽니다.
2. *introscope.agent.leakhunter.enable* 속성을 true 로 설정하고, 필요에 따라 다른 LeakHunter 속성도 설정합니다.
3. 응용 프로그램을 다시 시작합니다.

컬렉션 ID 를 사용하여 잠재 누수 확인

LeakHunter 는 Investigator 트리의 메트릭 데이터를 로그 파일의 데이터와 연결하는 데 사용되는 고유한 컬렉션 ID 로 잠재 누수 각각을 식별합니다. 컬렉션 ID 는 응용 프로그램 간에 안정된 이름을 제공하는 역할도 합니다.

컬렉션 ID 의 형식은 다음과 같습니다.

```
<method>-<4 digit hash code>#<unique number>
```

- <method>: 컬렉션이 할당된 메서드의 이름
- <4 digit hash code>: 메서드를 포함하는 클래스의 전체 이름을 나타내는 해시 코드
- #<unique number>: 에이전트를 실행하는 동안 컬렉션 ID 를 고유하게 식별할 수 있도록 메서드 및 해시 코드가 동일한 잠재 누수에 추가되는 번호

다음은 이러한 컬렉션 ID 의 예입니다.

```
theLookupTable-6314#1
getLoginID-1234#1
getLoginID-1234#2
getLoginID-1234#3
verifyCart-5678#1
verifyCart-0012#1
```

LeakHunter 로그 파일

LeakHunter 로그 파일에는 관리되는 응용 프로그램에서 Introscope LeakHunter 에 의해 식별된 잠재 누수 관련 정보가 들어 있습니다. 로그 파일의 각 항목은 잠재 누수 관련 정보를 포함합니다. LeakHunter 로그 파일에는 네 가지 시나리오에 대한 항목이 들어 있습니다.

- 잠재 누수가 처음 식별된 경우 - 처음 식별된 잠재 누수 로그 항목 (페이지 115) 참조
- 식별된 누수가 누수를 멈춘 것으로 나타나는 경우 - 누수를 멈춘 식별된 잠재 누수 로그 항목 (페이지 116) 참조
- 이전에 식별된 누수가 다시 누수되기 시작한 것으로 나타나는 경우 - 다시 누수되기 시작한 식별된 잠재 누수 로그 항목 (페이지 117) 참조
- LeakHunter 시간 만료가 발생한 경우 - LeakHunter 시간 만료 로그 항목 (페이지 117) 참조

처음 식별된 잠재 누수 로그 항목

이 유형의 LeakHunter 로그 항목에는 처음 식별된 잠재 누수에 대한 다음과 같은 정보가 포함됩니다.

- 현재 타임스탬프(로그에 기록된 시간)
- 컬렉션 ID
- 컬렉션 클래스
- 컬렉션 할당 방법
- 컬렉션 할당 시간
- 컬렉션 할당 스택 추적
- 컬렉션이 할당된 필드 이름
- 컬렉션의 현재 크기

참고: LeakHunter 로그 파일에 기록된 누수된 컬렉션의 현재 크기는 동적으로 업데이트되지 않습니다. 로그 파일에서는 누수가 처음 식별되었을 때의 누수 크기를 식별할 수 있습니다. 누수된 연결의 크기에 대한 최신 정보를 확인하려면 Introscope Workstation 에서 "누수" 탭을 클릭하십시오.

.NET 로그 파일 예제: 잠재 누수가 처음 식별된 경우

이 예에서는 계측 메서드로 AutoProbe 를 사용한다고 가정합니다.

```
10/18/2007 11:54:47 AM
Potential leak identified
Assigned ID: createNewInstance-2975#1
Collection Class:
System.Collections.Generic.LinkedList`1[[com.wily.tools.munger.ContainedObject,
munger, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null]]
Allocation Method:
[munger]com.wily.tools.munger.MungerFactory.createNewInstance[mscorlib,
Version=2.0.0.0,
PublicKeyToken=b77a5c561934e089]System.Object([mscorlib]System.String)
Allocation Timestamp: 10/18/2007 11:54:47 AM
Allocation Stack Trace:
Field Name(s):
    Unknown
Current Size: 10500
```

누수를 멈춘 식별된 잠재 누수 로그 항목

이 유형의 LeakHunter 로그 항목에는 누수를 멈춘 잠재 누수에 대한 다음과 같은 정보가 포함됩니다.

- 현재 타임스탬프(로그에 기록된 시간)
- 컬렉션 ID
- 컬렉션 클래스
- 컬렉션의 현재 크기

.NET 로그 파일 예제: 누수를 멈춘 것으로 나타나는 잠재 누수

```
10/18/2007 11:55:47 AM
Potential leak no longer appears to be leaking
Assigned ID: createNewInstance-2975#1
Collection Class:
System.Collections.Generic.LinkedList`1[[com.wily.tools.munger.ContainedObject,
munger, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null]]
Current Size: 28000
```

다시 누수되기 시작한 식별된 잠재 누수 로그 항목

이 유형의 항목에는 다시 누수를 시작한 잠재 누수에 대한 다음과 같은 정보가 포함됩니다.

- 현재 타임스탬프(로그에 기록된 시간)
- 컬렉션 ID
- 컬렉션 클래스
- 컬렉션의 현재 크기

.NET 로그 파일 예제: 누수가 다시 시작된 것으로 나타나는 잠재 누수

```
10/18/2007 11:57:47 AM
Potential leak appears to be leaking again
Assigned ID: createNewInstance-2975#1
Collection Class:
System.Collections.Generic.LinkedList`1[[com.wily.tools.munger.ContainedObject,
munger, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null]]
Current Size: 35000
```

LeakHunter 시간 만료 로그 항목

이 유형의 LeakHunter 로그 항목에는 계속 추적할 잠재 누수의 수가 포함됩니다.

로그 파일 예제: 시간 만료가 발생한 경우

```
LeakHunter timeout occurred at 4/27/04 1:32:12 PM PDT
LeakHunter will only continue to track the 3 potential leaks
```

LeakHunter 사용

LeakHunter 를 사용하는 방법에 대한 자세한 내용은 *CA APM Workstation 사용자 안내서*를 참조하십시오.

제 7 장: ErrorDetector 구성

이 섹션은 다음 항목을 포함하고 있습니다.

ErrorDetector 개요

Introscope ErrorDetector 를 통해 응용 프로그램 지원 담당자는 사용자가 웹 트랜잭션을 완료하는 데 문제가 되는 심각한 오류를 확인하고 오류 원인을 진단할 수 있습니다. 이러한 종류의 응용 프로그램 가용성 문제가 발생하면 대개 "404 찾을 수 없음"을 비롯한 오류 메시지가 표시되지만 이러한 오류 메시지에는 IT 담당자가 문제의 근본적인 원인을 찾는 데 필요한 구체적인 정보가 들어 있지 않습니다. Introscope ErrorDetector 를 사용하면 이러한 심각한 오류를 응용 프로그램에서 실시간으로 모니터링하고, 오류의 빈도와 원인을 확인할 수 있으며, 근본적인 원인에 대한 구체적인 정보를 개발자에게 전달할 수 있습니다.

ErrorDetector 는 심각한 응용 프로그램 오류의 근본적인 원인 분석을 통해 뛰어난 트랜잭션 무결성을 제공하고 최고의 사용자 환경을 보장하는 유일한 응용 프로그램 관리 솔루션입니다.

Introscope ErrorDetector 를 통해 IT 팀은 다음과 같은 작업을 수행할 수 있습니다.

- 비정상적인 트랜잭션의 빈도 확인
- 기록된 예외가 사용자에게 영향을 주는지 여부 확인
- 트랜잭션 경로 중 오류가 발생한 정확한 위치 확인
- 심각한 오류를 재현, 진단 및 제거하는 데 필요한 정보 파악

Introscope ErrorDetector 는 Introscope 와 통합되어 Introscope Workstation 에서 오류를 모니터링할 수 있는 기능을 제공합니다. 응용 프로그램 오류가 발생하면 라이브 오류 뷰어를 사용하여 각 오류에 대한 상세한 정보를 검토할 수 있습니다.

단원 정보

오류의 유형 (페이지 120)
ErrorDetector 작동 방식 (페이지 120)

오류의 유형

CA Technologies 는 .NET 사양에 포함된 정보에 기초하여 "심각한" 오류를 기술하는 일련의 조건을 정의했습니다. ErrorDetector 는 오류와 예외를 모두 오류로 간주합니다. 가장 일반적인 유형의 오류는 throw 된 예외입니다.

일반적인 오류의 예는 다음과 같습니다.

- HTTP 오류(404, 500 등)

  경우에 따라 HTTP 404 오류는 응용 프로그램 서버가 아니라 웹 서버에서 발생합니다. 이 경우 ErrorDetector 는 에이전트를 통해 웹 서버 오류를 감지하지 않습니다.

- SQL 문 오류
- 네트워크 연결 오류(시간 만료 오류)
- 백엔드 오류(예: JMS 를 통해 메시지를 보낼 수 없거나 메시지 큐에 메시지를 쓸 수 없는 오류)

CA Technologies 에서 중요한 오류라고 간주하는 항목이 사용자가 중요한 오류라고 생각하는 항목과 다를 수 있습니다. ErrorDetector 가 추적하는 오류 중에 중요하지 않다고 간주하는 오류가 있는 경우 무시하도록 선택할 수 있습니다. 추적하고자 하는 추가 오류가 있는 경우 오류 추적 프로그램을 사용하여 추가 오류를 추적하는 새 지시문을 만들 수 있습니다.

ErrorDetector 작동 방식

Introscope 에는 에이전트 설치 시 설치되는 *errors.pbd* 라는 PBD(ProbeBuilder 지시문) 파일이 포함되어 있습니다. 이 PBD 의 추적 프로그램은 심각한 오류를 캡처합니다.

ErrorDetector 는 에이전트 설치 시 자동으로 설치됩니다. ErrorDetector 가 설치된 후 Introscope 에서 *errors.pbd* 를 사용하도록 구성하고 ErrorDetector 기능이 사용되도록 설정하십시오.

Introscope Agent 는 *errors.pbd* 파일에 정의된 대로 오류 정보를 수집합니다.

Workstation 에서는 다음을 볼 수 있습니다.

- Investigator 에서 오류 메트릭 데이터를 볼 수 있습니다.
- "라이브 오류 뷰어"에서 라이브 오류를 볼 수 있습니다.
- 오류 발생 과정에 대한 구성 요소 수준의 정보를 보여 주는 새 "오류 스냅숏"에서 오류 세부 정보를 볼 수 있습니다.

ErrorDetector 는 트랜잭션 추적 프로그램과 통합되어 있으므로 트랜잭션 경로의 컨텍스트 내에서 심각한 오류가 발생한 원인 및 과정을 정확히 확인할 수 있습니다. 또한 모든 오류 및 트랜잭션이 CA Wily 의 트랜잭션 이벤트 데이터베이스에 보관되므로 기록 데이터를 분석하여 추세를 파악할 수 있습니다.

Introscope 는 트랜잭션을 서비스에 대한 호출 또는 처리로 정의합니다. 웹 응용 프로그램 컨텍스트에서 트랜잭션은 웹 브라우저에서 보낸 URL 에 대한 호출 및 처리입니다. 웹 서비스 컨텍스트에서 트랜잭션은 SOAP 메시지에 대한 호출 및 처리입니다.

.NET 에이전트에서 ErrorDetector 사용

기본적으로 .NET 에이전트는 오류 데이터를 캡처하도록 설정됩니다. *IntroscopeAgent.profile* 에서 *introscope.agent.errorsnapshots.enable* 속성을 *false* 로 설정하여 오류 데이터 캡처 기능을 사용하지 않도록 설정할 수 있습니다. *errors.pbd* 파일은 .NET 에이전트의 *default-full.pbl* 파일과 *default-typical.pbl* 파일에 기본적으로 나열되기 때문에 별도의 구성이 필요하지 않습니다.

ErrorDetector 옵션 구성

추적하지 않을 오류를 무시하도록 에이전트를 구성할 수 있습니다. 오류를 "태그 지정"하기 위해 지정하는 정보는 정확한 오류 메시지이거나 "와일드카드" 별표 문자와 메시지의 일부일 수 있습니다.

특정 오류를 무시하려면(선택 사항)

1. 에이전트 프로필 *IntroscopeAgent.profile* 을 엽니다.
2. *introscope.agent.errorsnapshots.ignore* 속성 값으로 원하는 오류 유형을 식별할 수 있는 정보를 정의합니다.

   예를 들어 다음 ignore 속성은 오류 메시지에 "IOException"이라는 용어가 포함된 모든 오류를 무시합니다.

   ```
   introscope.agent.errorsnapshots.ignore.0=*IOException*
   ```
3. 추가 오류를 무시하려면 추가 ignore 속성을 순차적으로 추가합니다. 예를 들어 두 가지 오류 유형을 무시하려면 다음과 같이 속성을 설정합니다.

   ```
   introscope.agent.errorsnapshots.ignore.0=*IOException*
   introscope.agent.errorsnapshots.ignore.1=*HTTP Error Code *500*
   ```
4. 에이전트 프로필의 변경 사항을 저장합니다.

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

고급 오류 데이터 캡처

ErrorDetector 는 기본적으로 일반적인 여러 오류 유형을 캡처하지만 CA Technologies 에서는 필요에 맞게 오류 감지 메커니즘을 사용자 지정하기 위한 옵션을 제공합니다.

다음의 오류 관련 추적 프로그램을 사용하여 오류를 캡처하는 PBD(ProbeBuilder 지시문)를 생성할 수 있습니다.

- **ExceptionErrorReporter** - 표준 예외를 보고합니다.
- **ThisErrorReporter** - 현재 개체를 오류로 보고합니다.
- **HTTPErrorCodeReporter** - HTTP 오류 코드와 관련 오류 메시지를 캡처합니다.
- **MethodCalledErrorReporter** - 특정 메서드가 호출되었는지 여부를 보고합니다.

새로 생성하는 지시문은 이러한 추적 프로그램과 함께 *<Agent_Home>*/wily 디렉터리의 *errors.pbd* 파일에 포함해야 합니다.

새 오류 유형 정의

오류는 PBD 를 사용하여 ErrorDetector 에 대해 정의됩니다. 오류의 존재 여부를 검사하고 오류 메시지를 캡처(구성)하는 여러 특수 추적 프로그램이 있습니다. 오른쪽에 이러한 추적 프로그램을 배치하면 응용 프로그램 또는 그 인프라의 새 오류 유형에 대해 ErrorDetector 를 학습시킬 수 있습니다.

ExceptionErrorReporter

ExceptionErrorReporter 추적 프로그램을 사용하여 계측된 메서드에서 throw 되는 예외를 확인할 수 있습니다. 예외가 throw 되면 이 추적 프로그램은 이를 오류로 처리하고 예외에서 오류 메시지를 가져옵니다. 이 메시지는 오류에 대한 가장 일반적인 정의입니다.

오류 메시지를 캡처하려면 ExceptionErrorReporter 추적 프로그램을 "...WithParameters" 지시문과 함께 사용해야 합니다. 예:

TraceOneMethodWithParametersOfClass: com.bank.CustomerAccount getBalance ExceptionErrorReporter "CustomerAccount:Errors Per Interval"

이 지시문은 *CustomerAccount* 의 *getBalance()* 메서드에서 throw 되는 예외가 오류로 간주되도록 지정합니다.

간격 메트릭별로 오류로 계측하려면 "...WithParameters" 지시문을 사용해야 하지만 임의의 메서드에 대해 "...WithParameters" 지시문을 한 번만 지정하면 해당 메서드에 대한 모든 추적 프로그램이 이 매개 변수를 사용할 수 있게 됩니다. 예를 들어 다음을 지정할 수 있습니다.

TraceOneMethodWithParametersOfClass: com.myClass myMethod BlamePointTracer

이 지시문은 *com.myClass myMethod* 메서드에 대한 매개 변수를 *ExceptionErrorReporter* 추적 프로그램을 포함한 다른 추적 프로그램에서 사용할 수 있도록 합니다.

MethodCalledErrorReporter

MethodCalledErrorReporter 추적 프로그램은 메서드에 사용되며, 이 메서드가 호출되는 것 자체가 오류가 발생했음을 의미합니다. 예:

TraceOneMethodOfClass: com.bank.CheckingAccount cancelCheck MethodCalledErrorReporter "CustomerAccount:Canceled Checks Per Interval"

이 지시문은 cancelCheck() 메서드가 (어떠한 이유로 인해) 호출될 때마다 이것을 오류로 지정합니다. 오류 메시지는 단순히 호출된 클래스 및 메서드가 기술합니다.

예외 또는 오류를 throw 하는 메서드를 모르는 경우 ThisErrorReporter 추적 프로그램을 사용해 보십시오.

ThisErrorReporter

ThisErrorReporter 추적 프로그램은 *MethodCalledErrorReporter* 와 유사하지만 계측된 개체에 대해 *toString()*을 호출하여 오류 메시지를 생성합니다. 이는 예외 클래스의 생성자에 사용하는 것이 가장 효과적입니다. 예:

```
TraceOneMethodWithParametersOfClass: ezfids.util.exception.EasyFidsException .ctor
ThisErrorReporter "Exceptions|{packageandclassname}:Errors Per Interval"
```

참고: 오류 메시지를 캡처하려면 *ThisErrorReporter* 추적 프로그램을 "...WithParameters" 지시문과 함께 사용해야 합니다.

이 지시문은 *InvalidPINException* 의 생성자("init" 또는 ".ctor")가 호출될 때마다 오류를 발생시킵니다. 오류 메시지는 *InvalidPINException* 에 *toString()*을 호출하여 결정되며, 일반적으로 응용 프로그램 개발자가 지정한 오류 메시지가 반환됩니다.

이 추적 프로그램은 고유한 예외 유형에 기반한 사용자 지정 오류 관리 시스템을 사용하는 경우에 유용합니다.

HTTPErrorCodeReporter

HTTPErrorCodeTracer 추적 프로그램은 ASP.NET 페이지의 오류 코드를 보고합니다. 이것은 다음의 인시던트를 카운트하는 간격당 카운터입니다.

- 400 이상의 HTTP 응답 코드
- .NET 환경에서 400 이상의 코드에 대한 ProcessRequest 의 *System.Web.IHttpHandler* 서브클래스 호출

사용법 예제는 *errors.pbd* 를 참조하십시오.

주의

이전 단원에 설명되어 있는 것처럼 추적 프로그램을 사용할 때는 주의를 기울여야 합니다. 가장 좋은 방법은 오류 추적과 관련된 오버헤드를 고려하여 백엔드 시스템에서 발생하는 복구 불가능한 문제와 같이 심각한 문제만 보고하는 것입니다.

기본 *errors.pbd* 는 가능한 한 오버헤드를 최소화하면서 심각한 오류를 보고하도록 설계되었습니다. 모니터링되는 모든 메서드에 ExceptionErrorReporter 를 적용하는 경우처럼 오류 추적을 과도하게 사용하면 오탐지의 양이 크게 증가할 수 있습니다. 예를 들어 사용자가 숫자 필드에 "California"를 입력하면 심각한 문제로 보고되지 않아도 되는 *NumberFormatException* 이 발생할 수 있습니다.

ErrorDetector 사용

ErrorDetector 사용 방법에 대한 자세한 내용은 *APM Workstation 안내서*를 참조하십시오.

제 8 장: 경계 Blame 구성

이 섹션은 다음 항목을 포함하고 있습니다.

경계 Blame 이해

Introscope 의 Blame 기술을 사용하면 관리되는 .NET 응용 프로그램의 응용 프로그램 계층, 즉 응용 프로그램의 프런트엔드와 백엔드에서 메트릭을 볼 수 있습니다. 경계 Blame 이라고 하는 이 기능을 통해 사용자는 문제를 응용 프로그램 프런트엔드 또는 백엔드에 심사할 수 있습니다.

이 단원에서는 Introscope 경계 Blame 기능의 구성 옵션에 대해 설명합니다.

Introscope 는 SQL 에이전트의 SQL 문 모니터링 기능을 활용하여 백엔드를 자동으로 확인합니다. SQL 에이전트를 사용할 수 없는 경우 Introscope 는 클라이언트/서버 데이터베이스와 같은 백엔드로 소켓 호출을 자동 확인하거나, 소켓을 통해 액세스하는 LDAP 서버를 확인합니다.

Introscope Investigator 에 경계 Blame 이 표시되는 방법에 대한 자세한 내용은 Introscope Workstation 사용자 안내서를 참조하십시오.

URL 그룹 사용

URL 그룹은 URL 경로 접두사를 사용하여 정의한 명명된 트랜잭션 그룹입니다. Introscope 는 각 URL 그룹에 대한 메트릭을 집계하고 해당 메트릭을 Introscope Investigator 의 *Frontends|Apps|<ApplicationName>|URLs* 노드 아래에 표시합니다.

경로 접두사는 호스트 이름 뒤에 오는 URL 부분입니다. 예를 들어 다음과 같은 URL 을 가정합니다.

```
http://burger1.com/TestWar/burgerServlet?ViewItem&category=
11776&item=5550662630&rd=1
```

여기서 경로 접두사는 다음과 같습니다.

```
/TestWar
```

URL 의 경로 접두사에서 파생될 수 있는 모든 유용한 요청 범주를 위한 URL 그룹을 정의할 수 있습니다. 예를 들어 응용 프로그램 URL 의 형식에 따라 응용 프로그램이 지원하는 각 고객에 대한 URL 그룹, 각 주요 응용 프로그램에 대한 URL 그룹, 하위 응용 프로그램에 대한 URL 그룹을 정의할 수 있습니다. 이러한 URL 그룹을 사용하면 커밋된 서비스 수준의 컨텍스트 또는 업무상 중요한 응용 프로그램 부분에서 성능을 모니터링할 수 있습니다.

기본적으로 모든 URL 이 *기본값*이라는 URL 그룹에 할당됩니다. 이를 통해 404 오류를 발생시키는 URL 같은 잘못된 URL 이 고유한 1 회성 메트릭을 생성하지 않을 경우에 발생하는 오버헤드를 방지할 수 있습니다. 의미 있는 URL 그룹을 구성하면 사용자가 하위 응용 프로그램 수준에서 성능을 모니터링할 수 있습니다.

관련 정보

URL 그룹 속성 (페이지 128)
샘플 URL 그룹 (페이지 128)
URL 그룹 정의 (페이지 129)
URL 그룹에 대한 고급 명명 기술(선택 사항) (페이지 130)

URL 그룹 속성

다음과 같은 속성을 사용하여 *IntroscopeAgent.profile* 파일에서 URL 그룹을 정의합니다.

- *introscope.agent.urlgroup.***keys**
- *introscope.agent.urlgroup.***group.default.pathprefix**
- *introscope.agent.urlgroup.***group.default.format**

샘플 URL 그룹

아래 목록은 에이전트 프로필에서 가져온 것으로, URL 그룹이 정의되는 방식을 설명합니다. 목록 다음에 나오는 단원에서는 필요한 속성을 구성하는 지침을 제공합니다.

```
introscope.agent.urlgroup.keys=alpha,beta,gamma
introscope.agent.urlgroup.group.alpha.pathprefix=/testWarintroscope.agent.urlgrou
p.group.alpha.format=foo {host} bar {protocol} baz {port} quux {query_param:foo} red
{path_substring:2:5} yellow {path_delimited:/:0:1} green{path_delimited:/:1:4} blue
{path_substring:0:0}
introscope.agent.urlgroup.group.beta.pathprefix=/nofilterWarintroscope.agent.urlg
roup.group.beta.format=nofilter foo {host} bar {protocol} baz {port}
quux{query_param:foo} red {path_substring:2:5} yellow{path_delimited:/:0:1} green
{path_delimited:/:1:4} blue{path_substring:0:0}
introscope.agent.urlgroup.group.gamma.pathprefix=/examplesWebAppintroscope.agent.
urlgroup.group.gamma.format=Examples Web App
```

URL 그룹 정의

다음 단원에서는 URL 그룹을 구성하는 속성에 대한 정보를 제공합니다.

단원 정보

URL 그룹에 대한 키 정의 (페이지 129)
각 URL 그룹의 구성원 자격 정의 (페이지 129)
URL 그룹의 이름 정의 (페이지 130)

URL 그룹에 대한 키 정의

introscope.agent.urlgroup.keys 속성을 사용하여 모든 URL 그룹의 ID 또는 키 목록을 정의합니다. URL 그룹의 특성을 선언하는 다른 속성 정의에서 URL 그룹의 키를 참조합니다. 이 예제에서는 URL 그룹 세 개에 대한 키를 정의합니다.

```
introscope.agent.urlgroup.keys=alpha,beta,gamma
```

URL 그룹의 키를 지정하는 순서가 중요합니다. 자세한 내용은 URL 그룹화 (페이지 272)를 참조하십시오.

각 URL 그룹의 구성원 자격 정의

*introscope.agent.urlgroup.group.*GroupKey.*pathprefix* 를 사용하여 URL 의 경로 접두사가 일치하는지 확인할 패턴을 지정함으로써 URL 그룹에 포함되는 요청을 정의합니다.

예제 1

이 속성 정의는 URL 의 경로 부분이 */TestWar* 로 시작하는 모든 요청을 키가 *alpha* 인 URL 그룹에 할당합니다.

introscope.agent.urlgroup.group.alpha.pathprefix=/TestWar

지정된 *pathprefix* 와 일치하는 요청에는 다음이 포함됩니다.

http://burger1.com/TestWar/burgerServlet?ViewItem&category=11776&item=5550662630&rd=1

http://burger1.com/TestWar/burgerServlet?Command=Order&item=5550662630

예제 2

콜 센터 서비스를 제공하는 회사가 각 응용 프로그램 기능에 대한 URL 그룹을 설정하여 기능 영역별로 응답 시간을 모니터링할 수 있습니다. 고객이 다음 URL 을 사용하여 서비스에 액세스한다고 가정합니다.

http://genesystems/us/*application_function*/

여기서 *application_function* 은 *OrderEntry*, *AcctService*, *Support* 등과 같은 응용 프로그램에 해당하며, 각 URL 그룹에 대한 *pathprefix* 속성은 적절한 *application_function* 을 지정해야 합니다.

참고: *pathprefix* 속성에서 별표 기호(*)를 와일드카드로 사용할 수 있습니다.

URL 그룹의 이름 정의

introscope.agent.urlgroup.group.GroupKey.format 을 사용하여 키가 *GroupKey* 인 URL 그룹에 대한 응답 시간 메트릭이 Workstation 에 표시되는 이름을 결정할 수 있습니다.

일반적으로 URL 에 대한 이름으로 텍스트 문자열을 할당하는 데 *format* 속성을 사용합니다. 이 예제에서는 키가 *alpha* 인 URL 그룹에 대한 메트릭이 *Alpha Group* 이란 이름으로 Workstation 에 표시되도록 합니다.

introscope.agent.urlgroup.group.alpha.format=*Alpha Group*

URL 그룹에 대한 고급 명명 기술(선택 사항)

필요한 경우 서버 포트, 프로토콜 같은 요청 요소 또는 요청 URL 의 부분 문자열에서 URL 그룹 이름을 파생시킬 수 있습니다. 요청을 검사하는 방법으로 응용 프로그램 모듈이 쉽게 구분되는 경우 이 방법이 유용합니다. 다음 단원에서는 *format* 속성의 고급 형식에 대해 설명합니다.

단원 정보

URL 그룹 이름으로 호스트 사용

요청과 연결된 HTTP 서버의 호스트 이름을 나타내는 이름 아래에서 URL 그룹의 메트릭을 구성하려면 다음과 같이 *format* 매개 변수를 정의합니다.

```
introscope.agent.urlgroup.group.alpha.format={host}
```

*format={host}*인 경우 다음 요청의 통계는 각각 메트릭 이름 *us.mybank.com* 및 *uk.mybank.com* 으로 표시됩니다.

```
https://us.mybank.com/mifi/loanApp...
https://uk.mybank.com/mifi/loanApp...
```

URL 그룹 이름으로 프로토콜 사용

요청과 연결된 프로토콜을 나타내는 이름 아래에서 URL 그룹의 통계를 구성하려면 다음과 같이 *format* 매개 변수를 정의합니다.

```
introscope.agent.urlgroup.group.alpha.format={protocol}
```

*format={protocol}*인 경우 Investigator 에서 요청 URL 의 프로토콜 부분에 해당하는 메트릭 이름 아래에 통계가 그룹화됩니다. 예를 들어 다음과 같은 요청의 통계는 메트릭 이름 *https* 아래에 나타납니다.

```
https://us.mybank.com/cgi-bin/mifi/scripts.....
https://uk.mybank.com/cgi-bin/mifi/scripts.....
```

URL 그룹 이름으로 포트 사용

요청과 연결된 포트를 나타내는 이름으로 URL 그룹의 통계를 구성하려면 다음과 같이 *format* 매개 변수를 정의하십시오.

```
introscope.agent.urlgroup.group.alpha.format={port}
```

*format={port}*인 경우 요청 URL 의 포트 부분에 해당하는 이름 아래에 통계가 그룹화됩니다. 예를 들어 다음과 같은 요청의 통계는 이름 *9001* 아래에 나타납니다.

```
https://us.mybank.com:9001/cgi-bin/mifi/scripts.....
https://uk.mybank.com:9001/cgi-bin/mifi/scripts.....
```

URL 그룹 이름으로 매개 변수 사용

Investigator 에서 요청과 연결된 매개 변수의 값을 나타내는 메트릭 이름 아래에 URL 그룹의 통계를 구성하려면 다음과 같이 format 매개 변수를 정의합니다.

```
introscope.agent.urlgroup.group.alpha.format={query_param:param}
```

*format={query_param:param}*인 경우, Investigator 에서 지정된 매개 변수의 값에 해당하는 메트릭 이름으로 통계가 그룹화됩니다. 매개 변수가 없는 요청은 *<empty>* 아래에 나열됩니다. 예를 들어 다음과 같은 매개 변수 정의를 사용할 수 있습니다.

```
introscope.agent.urlgroup.group.alpha.format={query_param:category}
```

이 경우 다음과 같은 요청의 통계는 메트릭 이름 "734" 아래에 나타납니다.

```
http://ubuy.com/ws/shopping?ViewItem&category=734&item=3772&tc=photo
http://ubuy.com/ws/shopping?ViewItem&category=734&item=8574&tc=photo
```

요청 경로의 부분 문자열을 URL 그룹 이름으로 사용

요청 URL 의 경로 부분과 일치하는 부분 문자열을 나타내는 이름 아래에서 URL 그룹의 통계를 구성하려면 다음과 같이 *format* 매개 변수를 정의합니다.

```
introscope.agent.urlgroup.group.alpha.format={path_substring:m:n}
```

여기서 *m* 은 첫 번째 문자의 인덱스이고, *n* 은 마지막 문자의 인덱스에 1 을 더한 숫자입니다. 예를 들어 다음과 같이 설정되었다고 가정해 보겠습니다.

```
introscope.agent.urlgroup.group.alpha.format={path_substring:0:3}
```

다음 요청에 대한 통계는 메트릭 노드 "*/ht*" 아래에 표시됩니다.

```
http://research.com/htmldocu/WebL-12.html
```

요청 경로의 구분된 부분을 URL 그룹 이름으로 사용

문자로 구분된 부분의 요청 URL 경로를 나타내는 이름 아래에 URL 그룹의 통계를 구성하려면 *format* 매개 변수를 다음과 같이 정의합니다.

```
introscope.agent.urlgroup.group.alpha.format=
{path_delimited:delim_char:m:n}
```

여기서 *delim_char* 는 경로의 세그먼트를 구분하는 문자이고, *m* 은 선택할 첫 번째 세그먼트의 인덱스이며, *n* 은 선택할 마지막 세그먼트의 인덱스보다 1 이 큰 값입니다. 예를 들어 다음과 같이 설정되었다고 가정해 보겠습니다.

```
introscope.agent.urlgroup.group.alpha.format={path_delimited:/:2:4}
```

또한 다음 형식의 요청에 대한 통계가 있다고 가정해 보겠습니다.

```
http://www.buyitall.com/userid,sessionid/pageid
```

이 통계는 메트릭 이름 */pageid* 로 나타납니다.

다음 사항에 유의하십시오.

- 구분 기호 문자는 세그먼트 단위로 계산됩니다.
- 세그먼트 번호는 0 부터 시작합니다.

이 표에서는 위 예제의 세그먼트를 슬래시 문자로 구분하여 보여 줍니다.

세그먼트 인덱스	0	1	2	3
세그먼트 문자열	/	*userid,sessionid*	/	*pageid*

여러 구분 기호를 필요한 대로 지정할 수 있습니다. 예를 들어 다음과 같이 설정되었다고 가정해 보겠습니다.

```
introscope.agent.urlgroup.group.alpha.format={path_delimited:/,:3:4}
```

위에 표시된 형식의 요청에 대한 통계는 메트릭 이름 *sessionid* 아래에 나타납니다.

이 표에서는 위 예제의 세그먼트를 슬래시 및 쉼표 문자로 구분하여 보여 줍니다.

세그먼트 인덱스	0	1	2	3	4	5
세그먼트 문자열	/	*userid*	,	*sessionid*	/	*pageid*

URL 그룹에 대한 다중 명명 방법 사용

아래에 표시된 것처럼 여러 개의 이름 지정 방법을 단일 *format* 문자열에 결합할 수 있습니다.

```
introscope.agent.urlgroup.group.alpha.format=red {host} orange {protocol} yellow
{port} green {query_param:foo} blue {path_substring:2:5} indigo
{path_delimited:/:0:1} violet {path_delimited:/:1:4} ultraviolet
{path_substring:0:0} friend computer
```

Blame 추적 프로그램을 사용하여 Blame 지점 표시

Introscope Blame 기술을 사용하면 .NET 응용 프로그램의 성능을 추적하여 응용 프로그램의 프런트엔드 메트릭과 백엔드 메트릭을 볼 수 있습니다. 경계 Blame 이라고 하는 이 기능을 통해 사용자는 문제를 응용 프로그램 프런트엔드 또는 백엔드에서 심사할 수 있습니다.

다음 단원에서는 추적 프로그램을 사용하여 응용 프로그램의 프런트엔드 및 백엔드를 명시적으로 표시하는 방법에 대해 설명합니다.

프런트엔드 및 백엔드 메트릭 정보를 사용하여 응용 프로그램 심사 맵에서 응용 프로그램 관련 정보를 표시할 수도 있습니다.

이 단원에서 다루는 항목

Blame 추적 프로그램

Introscope 는 프런트엔드 및 백엔드 메트릭을 캡처하기 위한 추적 프로그램인 FrontendTracer 및 BackendTracer 를 제공합니다. 이러한 추적 프로그램은 각각 프런트엔드와 백엔드를 명시적으로 표시합니다.

FrontendTracer 와 BackendTracer 를 사용하여 백엔드에 액세스하는 코드와 같은 사용자 고유의 코드를 계측하면 Introscope 가 사용자 지정 구성 요소에 대한 메트릭을 캡처하여 나타냅니다.

구성된 FrontendTracer 가 없는 경우 Blame 스택의 첫 번째 구성 요소가 기본 프런트엔드가 됩니다. BackendTracer 가 구성되어 있지 않은 경우 Introscope 는 백엔드를 유추합니다. 즉, 명시적으로 표시되지 않은 경우 클라이언트 소켓을 여는 모든 구성 요소가 기본 백엔드가 됩니다.

특정 클래스가 프런트엔드로 표시되지 않도록 하려면 PBD 매개 변수 **is.frontend.unless** 를 지정합니다. 자세한 내용은 사용자 지정 FrontendMarker 지시문 (페이지 135)을 참조하십시오.

Introscope 가 백엔드로 감지하는 항목에 원하는 이름을 지정하거나 Introscope 가 계측하지 않는 사용자 지정 소켓을 표시하려는 경우에는 BackendTracer 를 사용하는 것이 좋습니다.

FrontendTracer 와 BackendTracer 는 평균 응답 시간, 간격당 수, 동시성 및 중단과 같은 메트릭을 제공하는 BlamePointTracer 의 인스턴스입니다. BlamePointTracer 는 보다 세분화된 Blame 스택을 위해 "중간" 구성 요소에 적용할 수 있습니다. 하지만 BlamePointTracer 는 Introscope Investigator 에 "간격당 오류 수" 메트릭을 채우지 않습니다.

표준 PBD 의 Blame 추적 프로그램

경계 Blame 추적 기능은 Introscope 및 .NET 에이전트에 제공되는 두 가지 표준 PBD 인 dotnet.pbd 와 sqlagent.pbd 를 통해 구현됩니다.

- *dotnet.pbd* 에 포함된 *PageInfoTracer* 는 FrontendTracer 의 인스턴스입니다.
- *sqlagent.pbd* 에 포함된 *SQLBackendTracer* 는 BackendTracer 의 인스턴스입니다.

사용자 지정 FrontendMarker 지시문

Introscope 9.0 에는 PBD 매개 변수 **is.frontend.unless** 가 도입되었습니다. 이 매개 변수를 사용하면 FrontendMarker 또는 그 하위 클래스(예: PageInfoTracer)를 통해 계측된 일부 클래스를 프런트엔드 구성 요소로 표시하지 않을 수 있습니다. 이 매개 변수는 쉼표로 구분된 절대 클래스 이름 목록으로 설정해야 합니다. 이 매개 변수는 초기 구성 요소가 받은 요청 유형을 처리할 수 있는 보다 구체적인 구성 요소로 전달하는 일반 '발송자'인 경우에 유용할 수 있습니다. 따라서 두 번째 구성 요소는 보다 적절한 프런트엔드 표식이 됩니다. 기본값은 빈 목록입니다. PBD 매개 변수는 동적이 아니므로 이 매개 변수의 값을 변경한 경우에는 계측된 응용 프로그램 서버를 다시 시작해야 합니다.

중요! 클래스 이름은 공백 **없이** 쉼표로만 구분해야 합니다. 공백을 사용할 경우 SetTracerParameter 지시문이 무효화됩니다.

이 매개 변수가 적용된 추적 프로그램에서 계측되며 매개 변수 목록에 지정된 클래스는 프런트엔드로 지정되지 않으며 Introscope Investigator 의 "Frontends"(프런트엔드) 노드 아래에 메트릭을 생성하지도 않습니다.

예를 들어 *ASP.index_aspx* 및 *Fib.CalculatorController* 클래스를 *com.ABCCorp* 패키지에서 *PageInfoTracer* 라는 FrontendMarker 를 사용하여 계측되는 프런트엔드로 처리하지 않으려면 다음의 PDB 지시문을 사용하십시오.

```
SetTracerParameter: PageInfoTracer is.frontend.unless
ASP.index_aspx,Fib.CalculatorController
```

제 9 장: 트랜잭션 추적 프로그램 옵션

이 섹션은 다음 항목을 포함하고 있습니다.

새로운 트랜잭션 추적 모드

CA APM Introscope 9.1 에서는 새로운 에이전트 트랜잭션 아키텍처와 추적 프로그램이 도입되었습니다. 새로운 트랜잭션 추적 모드는 이전 추적 프로그램이 기존 릴리스에 사용하던 트랜잭션 Blame 스택을 대체합니다. 새 모드 구현은 에이전트 성능을 개선하기 위해 계산 및 처리를 최적화합니다.

CA APM Introscope 9.1 을 사용하면 트랜잭션 Blame 스택으로 이전처럼 에이전트를 구성 및 실행할 수 있습니다. 에이전트의 "레거시" 모드가 완전하게 지원되지만 이후 릴리스의 개선된 에이전트 기능은 새 모드에서만 구현될 것입니다. "레거시" 모드로 에이전트를 실행할 경우 다음과 같은 기능을 사용할 수 없습니다.

- 버전 9.1 의 에이전트 CPU 사용 및 응답 시간 최적화
- .NET 에이전트에 대한 동적 계측
- SQL 에이전트 속성

  `introscope.agent.sqlagent.sql.artonly`

  `introscope.agent.sqlagent.sql.turnoffmetrics`

중요! CA APM 에이전트에서 새 모드로 레거시 추적 프로그램을 실행하는 것을 "혼합 모드" 구성이라고 합니다. 메모리가 많이 소모되고 서비스가 중단될 수 있으므로 혼합 모드로 실행하지 **마십시오**.

레거시 모드 추적 프로그램을 실행하는 경우 레거시 모드로 에이전트를 실행하십시오. 레거시 추적 프로그램이 있음을 고객에게 알리기 위해 다음과 같은 내용의 메시지가 에이전트 로그에 기록됩니다.

"An agent tracer using legacy API's has been detected. Running legacy tracers with the agent in new mode is not recommended. Please contact support who can refer you to documentation on how to upgrade your legacy tracers. In the interim, please configure the agent to use legacy mode or use the pre-configured version of the legacy agent package. For example, the legacy package for the Oracle WebLogic agent on UNIX is IntroscopeAgentFiles-Legacy-NoInstallerx.x.x.xweblogic.unix.tar"
(레거시 API 를 사용하는 에이전트 추적 프로그램이 감지되었습니다. 에이전트에서 새 모드로 레거시 추적 프로그램을 실행하지 않는 것이 좋습니다. 레거시 추적 프로그램 업데이트 방법에 대한 설명서를 알려 줄 수 있는 지원 담당자에게 문의하십시오. 그동안에는 레거시 모드를 사용하거나 사전 구성된 버전의 레거시 에이전트 패키지를 사용하도록 에이전트를 구성하십시오. 예를 들어 UNIX 에서 Oracle WebLogic 에이전트용 레거시 패키지는 IntroscopeAgentFiles-Legacy-NoInstallerx.x.x.xweblogic.unix.tar 입니다.)

새 모드 추적 프로그램을 실행할 때까지는 에이전트를 레거시 모드에서 실행되도록 되돌려 놓으십시오.

기본 제공되는 다음 확장을 사용하는 경우 에이전트가 레거시 모드로 실행되도록 구성하십시오.

- CA APM for IBM Websphere Portal
- CA APM for IBM CICS Transaction Gateway
- CA APM for IBM z/OS
- CA APM for CA SiteMinder Web Access Manager
- CA APM Integration for CA LISA

레거시 모드 트랜잭션 추적을 사용하도록 에이전트 구성

레거시 모드 트랜잭션 추적으로 변환하는 기능은 새 모드가 있는 버전의 버전 CA APM 에만 적용됩니다. 9.1 이전 버전의 CA APM 에는 새 모드가 없습니다.

레거시 모드를 사용하도록 에이전트를 구성할 때는 두 가지 옵션을 사용할 수 있습니다.

- 미리 구성된 레거시 에이전트 패키지를 배포합니다. 이러한 패키지는 파일 전용 에이전트 패키지로 제공되며, 설치 관리자가 제공되지 않습니다. 예를 들어 UNIX 에서 Oracle WebLogic 에이전트용 레거시 패키지는 다음과 같습니다.

  IntroscopeAgentFiles-Legacy-NoInstaller<*version_number*>weblogic.unix.tar

- 레거시 모드를 사용하도록 수동으로 구성합니다.

다음 단계를 따르십시오.

1. 모니터링되는 응용 프로그램을 중지합니다.
2. 새 로그를 준비하기 위해 <Agent_Home>/logs 디렉터리에 있는 기존 로그 파일을 아카이브하고 삭제합니다.
3. <Agent_Home>/core/config 디렉터리의 기존 .pbl 및 .pbd 파일 백업합니다.
4. 기존 <Agent_Home>/core/config/IntroscopeAgent.profile 을 백업합니다.
5. <Agent_Home>/examples/legacy 디렉터리에 있는 레거시 .pbl 및 .pbd 파일을 <Agent Home>/core/config 디렉터리에 복사합니다.
6. <Agent_Home>/core/config/IntroscopeAgent.profile 을 열고 다음 변경을 수행합니다.
   - 속성 introscope.agent.configuration.old=true 를 추가합니다.
   - 복사된 적절한 레거시 .pbl 및 .pbd 파일을 가리키도록 에이전트 속성 introscope.autoprobe.directivesFile 을 업데이트합니다. 예를 들어, spm.pbl 을 spm-legacy.pbl 로 대체합니다.
7. 모니터링되는 응용 프로그램을 다시 시작합니다.

표준 설치에서 레거시 모드를 사용하도록 특정 CA APM 확장을 구성하려면 다음 표에 나열되어 있는 .pbl 및 .pbd 파일을 참조하십시오.

확장 이름	레거시 표준 pbl 또는 pbd 파일
CA APM for Oracle Weblogic Server	ppweblogic-legacy.pbd spm-legacy.pbl
CA APM for Oracle Weblogic Portal	powerpackforweblogicportal-legacy.pbl spm-legacy.pbl

확장 이름	레거시 표준 pbl 또는 pbd 파일
CA APM for Oracle Service Bus	OSB-typical-legacy.pbl spm-legacy.pbl
CA APM for IBM Websphere Portal	powerpackforwebsphereportal.pbl spm-legacy.pbl
CA APM for IBM Websphere MQ	webspheremq-legacy.pbl
CA APM for IBM Websphere Process Server/Business Process Management	wps-legacy.pbd wesb-legacy.pbd spm-legacy.pbl
CA APM for TIBCO BusinessWorks	tibcobw-typical-legacy.pbl spm-legacy.pbl
CA APM for Software AG Webmethods Integration Server	webmethods-legacy.pbl spm-legacy.pbl
CA APM for CA SiteMinder Web Access Manager	smwebagentext.pbd
CA APM for IBM CICS Transaction Gateway	PPCTGTranTrace.pbd PPCTGServer-typical.pbd PPCTGClient-typical.pbd
CA APM for IBM z/OS	zos-full.pbl
CA APM for CA LISA	lisa-typical.pbl
CA APM for SYSVIEW	CTG_ECI_Tracer_For_SYSVIEW-legacy.pbd HTTP_Tracer_For_SYSVIEW-legacy.pbd WS_Tracer_For_SYSVIEW-legacy.pbd

트랜잭션 추적 샘플링 제어

기본적으로 Introscope Agent 는 응용 프로그램에서 한 시간에 한 번씩 각각의 정규화된 고유 URL 을 추적하여 트랜잭션 동작을 샘플링합니다. 이 샘플링을 통해 트랜잭션 추적을 명시적으로 실행하지 않고도 잠재적인 문제가 있는 트랜잭션 유형에 대한 기록 분석을 수행할 수 있습니다.

트랜잭션 추적 샘플링은 기본적으로 사용됩니다. 에이전트 프로필 IntroscopeAgent.profile 에서 다음 속성의 주석 처리를 제거하여 이 동작이 사용되지 않도록 설정할 수 있습니다.

```
introscope.agent.transactiontracer.sampling.enabled
```

주석 처리를 제거하고 false 로 설정하여 트랜잭션 추적 샘플링이 사용되지 않도록 설정합니다.

에이전트 프로필에서 다음 속성의 주석 처리를 제거하여 간격당 샘플링되는 트랜잭션의 수와 간격의 길이를 구성할 수 있습니다.

중요! 이러한 구성은 대개 Enterprise Manager 에서 수행합니다. 에이전트 프로필에서 다음과 같은 속성을 구성하여 Enterprise Manager 로 수행한 모든 구성을 재정의할 수 있습니다.

```
introscope.agent.transactiontracer.sampling.perinterval.count
```

주석 처리를 제거하고 간격당 샘플링되는 트랜잭션 수를 설정합니다. 기본값은 1 입니다.

```
introscope.agent.transactiontracer.sampling.interval.seconds
```

주석 처리를 제거하고 샘플 간격을 초 단위로 설정합니다. 기본값은 120 초입니다.

트랜잭션 추적 구성 요소 클램프

Introscope 에는 추적 크기를 제한하기 위해 클램프가 설정되며, 클램프는 기본적으로 5,000 개의 구성 요소로 설정됩니다. 이 제한에 도달하면 로그에 경고가 표시되고 추적이 중지됩니다.

이 기능을 통해 예상된 구성 요소 수 이상으로 트랜잭션이 크게 증가하는 구성 요소를 클램프할 수 있습니다. 예를 들어 구성 요소가 수백 개의 개체 상호 작용 및 백엔드 SQL 호출을 처리하는 경우에 해당합니다. 클램프 기능이 없으면 트랜잭션 추적 프로그램은 이를 트랜잭션 하나로 간주하여 무한 지속됩니다. 극단적인 몇몇 경우에는 클램프가 설정되지 않았으면 추적이 완료되기 전에 CLR 에 메모리가 부족할 수 있습니다.

트랜잭션을 클램프하기 위한 속성은 *IntroscopeAgent.profile* 파일에 있습니다.

```
introscope.agent.transactiontrace.componentCountClamp=5000
```

클램프된 구성 요소, 즉 CountClamp 를 초과하는 구성 요소를 생성하는 추적의 경우에는 추적에 별표가 표시되고 관련된 도구 설명이 표시됩니다. 이러한 추적을 확인하는 방법에 대한 자세한 내용은 *CA APM Workstation 사용자 안내서*를 참조하십시오.

클램프를 너무 낮게 설정하면 응용 프로그램이 시작될 때 성능 모니터링(PerfMon) 또는 LeakHunter 예외가 발생할 수 있습니다. 이 문제가 발생하면 관리되는 .NET 응용 프로그램을 다시 시작해야 합니다.

트랜잭션 추적 프로그램 옵션

지정한 조건을 충족하는 트랜잭션만 추적하도록 Introscope Transaction Tracer 를 구성할 수 있습니다. 사용자 ID 데이터 또는 HTTP 요청 및 세션 속성을 기준으로 필터링할 수 있습니다.

중요! 사용자 ID 나 HTTP 특성을 기준으로 필터링할 수 있지만 둘을 함께 사용하여 필터링할 수는 없습니다. 두 가지 필터 유형을 함께 구성하면 잘못된 메트릭이 생성될 수 있으므로 이렇게 구성하지 마십시오.

추적할 트랜잭션을 제어하려면

1. 필터 매개 변수 수집 설정 (페이지 142)에 설명되어 있는 대로 필터 매개 변수를 보고하도록 .NET 에이전트를 설정합니다.
2. 사용자 ID 또는 HTTP 요청 데이터를 기준으로 필터링하도록 구성합니다.
   - 사용자 ID 를 기준으로 트랜잭션 추적 필터링 (페이지 143)
   - HTTP 요청 데이터를 기준으로 트랜잭션 추적 필터링 (페이지 145).

필터 매개 변수 수집 설정

기본적으로 .NET 에이전트는 추적하는 트랜잭션의 URL 만 보고합니다. 개별 HTTP 속성을 보고하는 기능은 트랜잭션 추적이 시스템 오버헤드에 미치는 영향을 최소화하도록 제한됩니다. 필터링을 사용하려면 먼저 에이전트 프로필에 다음 행을 추가하여 HTTP 속성을 수집을 사용하도록 설정해야 합니다.

```
introscope.agent.asp.disableHttpProperties=false
```

이렇게 하면 다음과 같은 정보가 수집됩니다.

- 응용 프로그램 이름
- Session ID(세션 URL)
- 컨텍스트 경로
- 서버 이름
- URL
- 컨텍스트 경로
- Normalized URL(정규화된 URL)
- HTTP 헤더
- HTTP 매개 변수
- HTTP 특성

에이전트 프로필에 다음 행을 추가하여 다른 속성을 수집할 수도 있습니다.

```
introscope.agent.asp.optionalProperties=true
```

이렇게 하면 다음과 같은 정보가 수집됩니다.

- 스키마
- URL 참조 페이지
- Method

사용자 ID 를 기준으로 트랜잭션 추적 필터링

프런트엔드 구성 요소에서 사용자 ID 를 기준으로 트랜잭션 추적을 필터링하도록 .NET 에이전트를 구성하려면 먼저 응용 프로그램에서 사용자 ID 를 지정하는 방법을 결정해야 합니다. 이 정보는 응용 프로그램을 개발한 응용 프로그램 구축자가 제공할 수 있습니다.

Introscope Transaction Tracer 는 액세스하는 사용자 ID 를 다음과 같은 방법 중 하나를 사용하여 식별합니다.

- HTTP 컨텍스트 ID
- HTTP 요청 헤더
- URL 사용자 정보
- HTTP 세션의 특성

중요! 현재 사용 중인 응용 프로그램에서 사용자 ID 를 지정하는 방법에 해당하는 구성 프로세스만 수행하십시오.

단원 정보

컨텍스트 ID 별 필터 (페이지 144)
URL 사용자별 필터 (페이지 144)
요청 헤더별 필터 (페이지 144)
세션 특성을 기준으로 필터링 (페이지 144)

컨텍스트 ID 별 필터

HTTP 컨텍스트 ID 를 통해 사용자 ID 에 액세스하는 경우 에이전트 프로필에서 다음 속성의 주석 처리를 제거합니다.

```
introscope.agent.transactiontracer.userid.method=HttpContext.User.Identity.Name
```

URL 사용자별 필터

URL 사용자 정보를 통해 사용자 ID 에 액세스하는 경우 에이전트 프로필에서 다음 속성의 주석 처리를 제거합니다.

```
introscope.agent.transactiontracer.userid.method=HttpContext.Request.Uri.UserInfo
```

요청 헤더별 필터

사용자 ID 가 HTTP 요청 헤더에서 결정되는 경우 에이전트 프로필에서 다음 속성 쌍의 주석 처리를 제거하고 두 번째 속성을 지정하는 키 문자열을 정의하십시오.

```
introscope.agent.transactiontracer.userid.method=HttpRequest.Headers.Get
introscope.agent.transactiontracer.userid.key=<application defined key string>
```

세션 특성을 기준으로 필터링

사용자 ID 가 HTTP 세션의 특성인 경우, 에이전트 프로필에서 다음 속성 쌍의 주석 처리를 제거하고 두 번째 속성을 지정하는 키 문자열을 정의합니다.

```
introscope.agent.transactiontracer.userid.method=HttpContext.Session.Contents
introscope.agent.transactiontracer.userid.key=<application defined key string>
```

HTTP 요청 데이터를 기준으로 트랜잭션 추적 필터링

다음을 포함한 HTTP 요청 속성을 기준으로 트랜잭션 추적을 필터링할 수 있습니다.

- 요청 헤더
- 요청 매개 변수
- 세션 특성

요청 매개변수, 세션 특성 등과 같은 여러 속성을 사용하여 필터링할 수 있습니다.

중요! 사용자 ID 별 필터링을 구성한 경우에는 HTTP 속성별 필터링을 구성하지 마십시오.

다음 단계를 따르십시오.

1. IntroscopeAgent.profile 파일을 엽니다.
2. Transaction Tracer Configuration 이라는 제목 아래에서 트랜잭션 추적 프로그램 속성을 찾습니다.
3. 특정 HTTP 요청 헤더를 수집하려면 다음 속성의 주석 처리를 제거하고 추적할 HTTP 요청 헤더를 쉼표로 구분된 목록으로 지정합니다.
   `introscope.agent.transactiontracer.parameter.httprequest.headers=User-Agent`
4. HTTP 요청 매개 변수 데이터를 수집하려면 이 속성의 주석 처리를 제거하고 추적할 HTTP 요청 매개 변수를 쉼표로 구분된 목록으로 지정합니다.
   `introscope.agent.transactiontracer.parameter.httprequest.parameters=parameter 1,parameter2`
5. HTTP 세션 특성 데이터를 수집하려면 다음 속성의 주석 처리를 제거하고 추적할 HTTP 세션 특성을 쉼표로 구분된 목록으로 지정합니다. 예를 들면 다음과 같습니다.
   `introscope.agent.transactiontracer.parameter.httpsession.attributes=attribute 1,attribute2`
6. 변경 사항을 IntroscopeAgent.profile 파일에 저장합니다.
7. 응용 프로그램을 다시 시작합니다.

구성 요소 중단 보고 구성

정의된 시간 동안 프로브 대상 구성 요소에서 응답이 없는 경우 Application Performance Management 가 중단됩니다. 기본적으로 APM Introscope Agent 가 이러한 상태를 감지하고 중단 메트릭을 보고합니다.

에이전트가 중단을 확인할 때마다 메서드 스택 맨 위에 있는 계측 대상 구성 요소가 모두 확인됩니다. 구성 요소가 중단된 경우 중단된 메트릭 및 중단 스냅숏이 생성됩니다. 또한 다운스트림 모니터링을 구독하는 각 구성 요소에 대해 중단 메트릭이 생성됩니다.

메서드 스택 맨 위에 있는 계측 대상 구성 요소에 대한 중단 메트릭이 먼저 생성됩니다. *introscope.agent.stalls.thresholdseconds* 값보다 오래 걸리는 구성 요소에 대해 이러한 메트릭이 생성됩니다.

다운스트림 구독자 구성 요소 중단

다운스트림 모니터링을 구독하는 구성 요소는 중단된 다운스트림 구성 요소를 호출할 때 중단됩니다.

기본적으로 다운스트림 중단 모니터링을 구독하는 구성 요소는 다음과 같습니다.

- FrontendTracer 프로브 대상 구성 요소
- BackendTracer 프로브 대상 구성 요소
- WebServiceBlamepointTracer@Servicelevel 프로브 대상 구성 요소
- WebServiceBlamepointTracer@Opertationlevel 프로브 대상 구성 요소

예

프런트엔드 구성 요소는 중단된 백엔드 구성 요소를 호출하는 Webservice 를 호출합니다.

프런트엔드 --> WebService --> 백엔드

백엔드, Webservice 및 프런트엔드 구성 요소에 대한 중단 메트릭이 생성되는데, 그러한 구성 요소 모두 기본적으로 다운스트림 모니터링을 구독하기 때문입니다.

업스트림 상속 구성 요소 중단

중단 검사기는 먼저 구성 요소 스택의 최상위 구성원을 검사하여 중단되었는지 확인합니다. 최상위 구성 요소가 중단되지 않은 경우 중단 검사기는 중단된 부모 구성 요소가 있는지 검사합니다. 부모가 중단된 경우 스택의 해당 구성 요소 및 각 업스트림 부모에 대한 중단 메트릭이 생성됩니다.

구성 요소 M1()이 여러 구성 요소 M2()를 호출하며 각각 걸리는 시간이 몇 초에 불과할 경우에는 M1()에 대한 중단 메트릭이 보고될 수 있습니다.

예제:

중단 임계값이 15 초로 설정되었습니다. 메서드 M1()이 메서드 M2()를 루프 상태로 100 회 호출합니다. M2()는 매번 응답하는 데 걸리는 시간이 2 초에 불과하므로 중단되지 않습니다. 그러나 결국 M1()은 중단됩니다.

트랜잭션 중단을 이벤트로 캡처하지 않도록 설정

기본적으로 Introscope 는 트랜잭션 중단을 트랜잭션 이벤트 데이터베이스에 이벤트로 캡처하고 해당 이벤트로부터 중단 메트릭을 생성합니다. 중단 메트릭은 트랜잭션의 첫 번째 메서드와 마지막 메서드에 대해 생성됩니다. 중단 이벤트 및 관련 메트릭은 Workstation 기록 이벤트 뷰어에서 볼 수 있습니다.

참고: 생성된 중단 메트릭은 항상 사용할 수 있지만 중단 이벤트는 Introscope Error Detector 가 설치되어 있어야만 볼 수 있습니다. 중단은 일반 오류로 저장되며 오류 탭 뷰 및 라이브 오류 뷰어에서 확인할 수 있습니다. 또한 "type:errorsnapshot"을 쿼리하여 기록 쿼리 뷰어에서 확인할 수도 있습니다. 중단된 트랜잭션은 라이브 오류 뷰어 및 오류 탭 아래에서 오류로 보고됩니다. "일치하는 오류 *" 조건을 사용하여 추적할 경우 이 동작은 설계된 동작이므로 중단된 트랜잭션이 트랜잭션 추적 창에서 오류로 보고되지 않습니다.

중단을 이벤트로 캡처하지 않도록 설정하거나, 중단 임계값을 변경하거나, .NET 에이전트가 중단을 검사하는 빈도를 변경하려면 다음 속성을 사용하십시오.

- introscope.agent.stalls.enable - 에이전트가 중단을 확인하고 발견된 중단에 대해 이벤트를 생성할지 여부를 제어합니다.
- introscope.agent.stalls.thresholdseconds - 트랜잭션을 중단된 것으로 판단하는 최소 임계값 응답 시간을 지정합니다.
- introscope.agent.stalls.resolutionseconds - 에이전트가 중단을 확인하는 빈도를 지정합니다.

중요! 중단을 이벤트로 캡처하는 기능을 사용하지 않으면 PBD 에서 중단 추적 프로그램이 더 이상 지원되지 않습니다.

비식별 트랜잭션 추적

CEM UI 에서 이 기능을 사용하도록 설정해도 비식별 트랜잭션에 대한 추적은 기본적으로 생성되지 않습니다.

비식별 트랜잭션에 대한 추적을 사용하려면 introscopeAgent.profile 에서 다음 속성의 값을 FALSE 로 설정하십시오.

```
introscope.agent.bizdef.turnOff.nonIdentifying.txn=FALSE
```

제 10 장: Introscope SQL 에이전트 구성

이 섹션은 다음 항목을 포함하고 있습니다.

SQL 에이전트 개요

Introscope SQL 에이전트는 자세한 데이터베이스 성능 데이터를 Enterprise Manager 에 보고합니다. SQL 에이전트를 사용하면 관리되는 응용 프로그램과 데이터베이스 간의 상호 작용을 추적하여 응용 프로그램에서 개별 SQL 문의 성능을 파악할 수 있습니다.

SQL 에이전트는 .NET 에이전트가 .NET 웹 응용 프로그램을 모니터링하는 것과 같은 방법으로 SQL 문을 모니터링합니다. SQL 에이전트는 매우 낮은 오버헤드로 응용 프로그램 또는 데이터베이스를 모니터링하므로 다른 프로세스에 방해가 되지 않습니다.

개별 SQL 문 수준까지 의미 있는 성능 측정 데이터를 제공하기 위해 SQL 에이전트는 트랜잭션 관련 데이터를 제외하고 원래 SQL 문을 Introscope 에서 사용되는 정규화된 문으로 변환하는 방법으로 성능 데이터를 요약합니다. 정규화된 문에는 신용 카드 번호와 같은 중요한 정보가 포함되지 않으므로 이 프로세스를 통해 데이터가 보호됩니다.

예를 들어 다음과 같은 SQL 쿼리가 있다고 가정해 보겠습니다.

```
SELECT * FROM BOOKS WHERE AUTHOR = 'Atwood'
```

SQL 에이전트는 이 SQL 쿼리를 다음과 같이 정규화된 문으로 변환합니다.

```
SELECT * FROM BOOKS WHERE AUTHOR = ?
```

마찬가지로 다음과 같은 SQL 업데이트 문이 있다고 가정해 보겠습니다.

```
INSERT INTO BOOKS (AUTHOR, TITLE) VALUES ('Atwood', 'The Robber Bride')
```

SQL 에이전트는 이 SQL 쿼리를 다음과 같이 정규화된 문으로 변환합니다.

```
INSERT INTO BOOKS (AUTHOR, TITLE) VALUES (?, ?)
```

따옴표로 묶은 텍스트('xyz')만 정규화됩니다.

정규화된 문에 대한 메트릭은 집계되며 Workstation Investigator 의 "JDBC" 노드에서 볼 수 있습니다.

SQL 에이전트 파일

SQL 에이전트는 모든 에이전트 설치에 포함되어 있습니다. SQL 에이전트 기능을 제공하는 파일은 다음과 같습니다.

- *wily/ext/wily.SQLAgent.ext.dll*
- *wily/sqlagent.pbd*

기본적으로 *wily.SQLAgent.ext.dll* 파일과 같은 에이전트 확장은 *wily/ext* 디렉터리에 설치됩니다. 에이전트 프로필의 *introscope.agent.extensions.directory* 속성을 사용하여 에이전트 확장 디렉터리의 위치를 변경할 수 있습니다. */ext* 디렉터리의 위치를 변경할 경우에는 */ext* 디렉터리의 내용도 이동해야 합니다.

SQL 문 정규화

일부 응용 프로그램의 경우 엄청나게 많은 수의 고유 SQL 문을 생성할 수 있습니다. Hibernate 와 같은 기술을 사용 중인 경우 긴 고유 SQL 문이 발생할 가능성이 증가합니다. 긴 SQL 문이 있으면 에이전트에서 메트릭이 폭발적으로 증가하여 성능이 저하되고 다른 시스템 문제가 발생할 수 있습니다.

Hibernate 에 대한 자세한 내용은 http://www.hibernate.org를 참조하십시오.

단원 정보

잘못 작성된 SQL 문으로 인한 메트릭 급증 (페이지 151)
SQL 문 정규화 옵션 (페이지 153)
기본 SQL 문 노멀라이저 (페이지 153)
사용자 지정 SQL 문 노멀라이저 (페이지 153)

잘못 작성된 SQL 문으로 인한 메트릭 급증

일반적으로 SQL 에이전트 메트릭의 수는 고유 SQL 문의 수에 근접해야 합니다. 응용 프로그램에서 사용하는 SQL 문의 수가 적은데도 SQL 에이전트에서 많은 수의 고유 SQL 메트릭을 보여 주고 있으며 그 수가 계속 증가하고 있다면 SQL 문의 작성 방식에 문제가 있을 수 있습니다.

SQL 문으로 인해 메트릭이 급증할 수 있는 일반적인 경우에는 몇 가지가 있습니다. 예를 들어 주석을 포함하거나 임시 테이블을 생성하거나 값 목록을 삽입하는 SQL 문은 실행될 때마다 뜻하지 않게 고유 메트릭 이름을 생성할 수 있습니다.

예: SQL 문의 주석

메트릭 급증의 일반적인 원인 중 하나는 SQL 문에서 주석이 사용된 방식에 있습니다. 예를 들어 다음과 같은 SQL 문이 있다고 가정합니다.

```
"/* John Doe, user ID=?, txn=? */ select * from table..."
```

이 경우 SQL 에이전트는 다음과 같이 주석을 메트릭 이름의 일부로 포함하는 메트릭을 생성합니다.

```
"/* John Doe, user ID=?, txn=? */ select * from table..."
```

SQL 문에 포함된 주석은 데이터베이스 관리자가 누가 어떤 쿼리를 실행하고 있는지 확인하는 데 유용하며, 쿼리를 실행하는 데이터베이스에서는 주석을 무시합니다. 그러나 SQL 에이전트에서는 SQL 문을 캡처할 때 주석 문자열을 구문 분석하지 않습니다. 따라서 SQL 에이전트에서는 각각의 고유한 사용자 ID 마다 고유한 메트릭을 생성하며, 이로 인해 메트릭이 급증하게 될 수 있습니다.

SQL 주석을 작은따옴표로 묶으면 이 문제를 방지할 수 있습니다. 예:

```
"/*' John Doe, user ID=?, txn=? '*/ select * from table..."
```

그러면 SQL 에이전트에서는 다음과 같은 메트릭을 생성하며 이 경우 주석으로 인한 고유 메트릭 이름은 더 이상 생성되지 않습니다.

```
"/* ? */ select * from table..."
```

예: 임시 테이블 또는 자동으로 생성된 테이블 이름

메트릭 급증의 또 다른 가능한 이유는 SQL 문에 자동으로 생성된 이름이 있는 임시 테이블을 참조하는 응용 프로그램에 기인할 수 있습니다. 예를 들어, Investigator 를 열어 Backends(백엔드)|{backendName}|SQL|{sqlType}|sql. 아래의 메트릭을 보는 경우 다음과 같은 메트릭을 확인할 수 있습니다.

```
SELECT * FROM TMP_12398139821038192091 2 WHERE ROW_ID = ?
```

이 SQL 문은 테이블 이름에 추가된 고유 식별자를 갖는 임시 테이블에 액세스합니다. *TMP_* 테이블 이름에 추가된 추가적인 숫자는 이 명령문이 실행될 때마다 고유한 메트릭 이름을 생성하므로 메트릭이 급증하게 됩니다.

예: 값 목록을 생성하거나 값을 삽입하는 문

메트릭 급증의 또 다른 일반적인 원인은 값 목록을 생성하거나 값을 대량으로 수정하는 SQL 문에 있습니다. 예를 들어 메트릭 급증이 발생할 수 있다는 경고를 받은 후 조사 과정에서 다음과 같은 SQL 문을 검토하게 되었다고 가정합니다.

```
#1 INSERT INTO COMMENTS (COMMENT_ID, CARD_ID, CMMT_TYPE_ID,
CMMT_STATUS_ID,CMMT_CATEGORY_ID, LOCATION_ID, CMMT_LIST_ID, COMMENTS_DSC,
USER_ID,LAST_UPDATE_TS) VALUES (?, ?, ?, ?, ?, ?, ?, "CHANGE CITY FROM CARROLTON,TO
CAROLTON, _ ", ?, CURRENT)
```

이 코드를 조사해 보면 *"CHANGE CITY FROM CARROLTON, TO CAROLTON, _ "* 부분에서 도시 배열이 생성됨을 알 수 있습니다.

마찬가지로, 잠재적 메트릭 급증을 조사할 경우 다음과 같은 SQL 문을 검토하게 될 수도 있습니다.

```
CHANGE COUNTRY FROM US TO CA _ CHANGE EMAIL ADDRESS FROM TO BRIGGIN @ COM _ "
```

이 코드를 조사해 보면 CHANGE COUNTRY 로 인해 긴 국가 목록이 생성됨을 알 수 있습니다. 또한 국가에 따옴표를 사용함으로써 사람들의 전자 메일 주소가 SQL 문 내에 삽입되게 되고, 이로 인해 메트릭 급증의 원인이 될 수 있는 고유 메트릭이 생성됩니다.

SQL 문 정규화 옵션

SQL 에이전트에는 긴 SQL 문의 처리를 위해 다음과 같은 노멀라이저가 포함됩니다.

- 기본 SQL 문 노멀라이저 (페이지 153)
- 사용자 지정 SQL 문 노멀라이저 (페이지 153)
- 정규식 SQL 문 노멀라이저
- 명령줄 SQL 문 노멀라이저

기본 SQL 문 노멀라이저

표준 SQL 문 노멀라이저는 기본적으로 SQL 에이전트에 있으며, 작은따옴표 내의 텍스트('xyz')를 정규화합니다. 예를 들어 다음과 같은 SQL 쿼리가 있다고 가정해 보겠습니다.

```
SELECT * FROM BOOKS WHERE AUTHOR = 'Atwood'
```

SQL 에이전트는 이 SQL 쿼리를 다음과 같이 정규화된 문으로 변환합니다.

```
SELECT * FROM BOOKS WHERE AUTHOR = ?
```

정규화된 문에 대한 메트릭은 집계되며 Workstation Investigator 에서 볼 수 있습니다.

사용자 지정 SQL 문 노멀라이저

SQL 에이전트를 사용하면 사용자 지정 정규화를 수행하기 위해 확장을 추가할 수 있습니다. 이렇게 하려면 SQL 에이전트에서 구현된 정규화 체계가 들어 있는 DLL 파일을 생성하십시오.

SQL 문 노멀라이저 확장을 적용하려면

1. 확장 DLL 파일을 만듭니다.

   SQL 노멀라이저 확장 파일의 진입점 클래스는 *com.wily.introscope.agent.trace.ISqlNormalizer* 인터페이스를 구현해야 합니다.

DLL 확장 파일을 만들려면 SQL 노멀라이저 확장에 필요한 특정 키가 들어 있는 매니페스트 파일을 만들어야 합니다. 매니페스트 파일을 만드는 방법은 아래의 2 단계에서 설명합니다. 그러나 확장이 작동하기 위해서는 다른 일반 키도 필요합니다. 이러한 키는 확장 파일을 구성하는 데 사용하는 유형입니다. 생성하는 확장 파일은 예를 들어 *Backends(백엔드)|{backendName}|SQL|{sqlType}|{actualSQLStatement}* 노드 아래의 메트릭과 같은 데이터베이스 SQL 문 텍스트 정규화와 관련됩니다. *{actualSQLStatement}*는 SQL 노멀라이저에 의해 정규화됩니다.

2. 생성된 확장의 매니페스트에 다음 키를 추가합니다.
   ```
   com-wily-Extension-Plugins-List:testNormalizer1
   ```
   참고: 이 키의 값은 원하는 대로 지정할 수 있습니다. 이 예에서는 *testNormalizer1* 이 사용됩니다. 이 키에 지정한 값을 다음 키에도 사용합니다.
   ```
   com-wily-Extension-Plugin-testNormalizer1-Type: sqlnormalizer
   com-wily-Extension-Plugin-testNormalizer1-Version: 1
   com-wily-Extension-Plugin-testNormalizer1-Name: normalizer1
   ```
   참고: 위의 키에는 normalizer1 과 같이 고유한 노멀라이저 이름이 포함되어야 합니다.
   ```
   com-wily-Extension-Plugin-testNormalizer1-Entry-Point-Class:
   <Thefully-qualified classname of your implementation of ISQLNormalizer>
   ```
3. 생성한 확장 파일을 *<Agent_Home>\wily\ext* 디렉터리에 둡니다.
4. *IntroscopeAgent.profile* 에서 다음 속성을 찾아 설정합니다.
   ```
   introscope.agent.sqlagent.normalizer.extension
   ```
   생성한 확장의 매니페스트 파일에서 이 속성을 *com-wily-Extension-Plugin-{plugin}-Name* 으로 설정합니다. 이 속성 값은 대/소문자를 구분하지 않습니다. 예:
   ```
   introscope.agent.sqlagent.normalizer.extension=normalizer1
   ```
   참고: 이 속성은 핫 속성입니다. 확장 이름을 변경하면 확장이 다시 등록됩니다.
5. 필요한 경우 *IntroscopeAgent.profile* 에서 다음 속성을 추가하여 오류 스로틀 수를 설정할 수 있습니다.
   ```
   introscope.agent.sqlagent.normalizer.extension.errorCount
   ```
   오류 및 예외에 대한 자세한 내용은 예외 (페이지 155)를 참조하십시오.

   참고: 사용자 지정 노멀라이저 확장에서 발생한 오류가 오류 제한 수를 초과하면 확장이 사용할 수 없게 설정됩니다.

6. *IntroscopeAgent.profile* 을 저장합니다.
7. 응용 프로그램을 다시 시작합니다.

예외

사용자가 만든 확장이 한 쿼리에 대해 예외를 throw 하는 경우 기본 SQL 문 노멀라이저는 해당 쿼리에 대해 기본 정규화 체계를 사용합니다. 이 경우 로그에 확장에서 예외가 throw 되었다는 오류 메시지가 기록되고 스택 추적 정보를 포함하는 디버그 메시지가 기록됩니다. 하지만 이와 같은 예외가 5 번 throw 되면 기본 SQL 문 노멀라이저가 사용자가 만든 확장이 사용되지 않도록 설정하고 노멀라이저가 변경될 때까지 이후의 쿼리에서 해당 확장을 사용하려는 시도를 중단시킵니다.

Null 또는 비어 있는 문자열

사용자가 만든 확장이 쿼리에 대해 null 문자열이나 비어 있는 문자열을 반환한 경우 StatementNormalizer 는 해당 쿼리에 대해 기본 정규화 체계를 사용하고 확장이 null 값을 반환했음을 나타내는 정보 메시지를 로그에 기록합니다. 하지만 그와 같은 null 또는 비어 있는 문자열이 5 번 반환된 후에는 StatementNormalizer 가 메시지 기록을 중지하고 확장을 계속 사용하려고 시도합니다.

제 11 장: 문제 해결 및 팁

이 단원에서는 .NET 에이전트 작업과 관련된 공통 구성 문제와 권장 사항을 설명합니다.

이 섹션은 다음 항목을 포함하고 있습니다.

.NET 에이전트가 제대로 설치되었는지 여부 확인

.NET 에이전트가 설치되었는지 또는 특정 컴퓨터에서 사용하지 않도록 설정되었는지 알지 못하거나 설치 또는 구성 오류를 인식하지 못하는 경우가 생길 수 있습니다. .NET 에이전트가 설치되어 있는지 확실하지 않거나 .NET 에이전트가 올바르게 구성되었는지 확인하려는 경우에는 다음과 같은 방법으로 환경을 확인해 볼 수 있습니다.

- .NET 에이전트 환경 변수가 설정되어 있고 값이 올바른지 확인합니다.

  "명령" 창을 열고 *set* 을 입력한 후 출력을 텍스트 파일로 지정한 다음 파일에 다음과 같은 환경 변수가 설정되어 있는지 확인합니다.

  ```
  com.wily.introscope.agentProfile=C:\Program Files\CA
  Wily\Introscope9.1.0.0\wily\IntroscopeAgent.profile
  Cor_Enable_Profiling=0x1
  COR_PROFILER={5F048FC6-251C-4684-8CCA-76047B02AC98}
  ```

- Windows 시스템 폴더에서 *assembly* 디렉터리(예: *C:\WINDOWS\assembly*)로 이동한 후 *wily.Agent* 가 등록된 "어셈블리 이름"으로 나열되고 버전 정보가 올바른지 확인합니다.

  목록에 *wily.Agent* 가 표시되지 않으면 *wilyregtool.exe* 또는 *gacutil.exe* 를 사용하여 어셈블리를 GAC(전역 어셈블리 캐시)에 수동으로 등록합니다.

- IIS 작업자 프로세스를 실행하는 사용자 계정을 확인하고, 작업자 프로세스를 실행하는 모든 계정이 *<Agent_Home>* 디렉터리에 대해 모든 권한을 가지고 있는지 확인합니다.

에이전트 확장의 버전 정보 수정

.NET 에이전트 및 선택적 확장 설치 시기에 따라 확장의 버전 번호와 .NET Agent 의 버전 번호가 다를 수 있습니다. 대부분의 경우 .NET 에이전트와 확장 간의 버전 정보가 다르면 에이전트가 로그에 오류 메시지를 기록하고 확장이 올바르게 작동하지 못합니다. 이 경우 버전 정보를 수동으로 업데이트하여 이 유형의 문제를 해결할 수 있습니다. 사용 중인 환경에 따라 다음 중 하나를 수행하십시오.

- 올바른 에이전트 버전 번호를 사용하도록 개별 응용 프로그램을 구성합니다.
- 전역적으로 올바른 에이전트 버전 번호를 사용하도록 모든 응용 프로그램을 구성합니다.

중요: *하나의 옵션*만 선택하고 두 옵션을 모두 수행하지는 마십시오.

개별 응용 프로그램의 버전 정보를 구성하려면

1. 설치하려는 확장이 일반 *.exe* 인 경우 *<Extension_Name>.exe.config* 라는 파일을 만들어 원래 *.exe* 와 동일한 디렉터리에 저장합니다.
2. 확장이 IIS 응용 프로그램인 경우 *w3wp.exe.config* 라는 파일을 생성하고 *w3wp.exe* 와 동일한 디렉터리에 배치합니다. 이것은 기본 도메인인 경우의 해결 방법입니다.
3. 확장이 IIS 응용 프로그램인 경우 각각의 개별 응용 프로그램에 대해 *web.config* 에 다음 항목을 추가합니다.

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="..." .../>
      <bindingRedirect oldVersion="0.0.0.0 - 65535.65535.65535.65535"
newVersion="<AGENT.VERSION.NUMBER>"/>
   </dependentAssembly>
</assemblyBinding>
```

   *assemblyIdentity 이름*을 입력하고 에이전트 버전 번호를 .NET 에이전트 버전 번호로 바꿉니다. 버전 정보는 네 자리 숫자로 구성되어야 합니다. 예를 들어 **9.0.7** 버전의 .NET 에이전트를 설치한 경우 다음을 추가할 수 있습니다.

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="wily.Agent"  publicKeyToken="2B41FDFB6CD662A5"/>
      <bindingRedirect oldVersion="9.0.5.0 - 65535.65535.65535.65535"
newVersion="9.0.7.0" />
   </dependentAssembly>
</assemblyBinding>
```

참고: 위 파일이 이미 있는 경우 *<assemblyBinding>* 노드를 *<runtime>* 아래에 추가하십시오.

모든 응용 프로그램을 전역적으로 구성하려면

- *machine.config* 파일에 다음을 추가합니다. 이 파일은 일반적으로 응용 프로그램의 *%runtime install path%\Config* 디렉터리에 있습니다.

```
assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="..." .../>
      <bindingRedirect oldVersion="0.0.0.0 - 65535.65535.65535.65535"
newVersion="<AGENT.VERSION.NUMBER>"/>
   </dependentAssembly>
</assemblyBinding>
```

  *assemblyIdentity 이름*을 입력하고 에이전트 버전 번호를 .NET 에이전트 버전 번호로 바꿉니다. 버전 정보는 네 자리 숫자로 구성되어야 합니다. 예를 들어 **9.0.7** 버전의 .NET 에이전트를 설치한 경우 다음을 추가할 수 있습니다.

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
   <dependentAssembly>
      <assemblyIdentity name="wily.Agent"  publicKeyToken="2B41FDFB6CD662A5"/>
      <bindingRedirect oldVersion="9.0.5.0 - 65535.65535.65535.65535"
newVersion="9.0.7.0" />
   </dependentAssembly>
</assemblyBinding>
```

참고: *machine.config* 에 코드 조각을 추가하면 전역적으로 모든 응용 프로그램에 영향을 미칩니다.

hotdeploy 디렉터리의 사용 여부 선택

hotdeploy 디렉터리는 *IntroscopeAgent.profile* 을 편집하지 않고 대개는 관리되는 응용 프로그램을 다시 시작할 필요 없이 새로운 지시문을 배포하는 데 사용됩니다. 그러나 이 기능을 사용할 때는 상당한 주의가 요구됩니다. 사용자 지정 PBD 파일이 잘못된 구문을 포함하거나 메트릭을 너무 많이 수집하도록 구성되어 있으면 그 영향이 더 빠르게 나타납니다. 잘못된 PBD 를 사용하면 NativeProfiler 가 종료될 수 있으며 PBD 가 너무 많은 메트릭을 수집하는 경우에는 응용 프로그램 성능이 저하될 수 있습니다.

이러한 문제를 해결하기 위해 다음과 같이 하는 것이 좋습니다.

- 모든 지시문을 QA 및 성능 환경에서 테스트하고 검증한 이후에 프로덕션 환경에서 사용합니다.
- 새로운 PBD 배포 옵션을 반영하도록 서버 환경의 변경 제어 프로세스를 업데이트합니다.

새 PBD 를 *hotdeploy* 디렉터리에 배치하면 .NET 에이전트가 해당 PBD 를 자동으로 배포합니다. 그러나 이미 실행 중인 클래스와 응용 프로그램의 경우에는 해당 응용 프로그램을 다시 시작해야만 새 PBD 나 변경된 PBD 의 내용이 적용됩니다.

참고: PBD 파일만 *hotdeploy* 디렉터리에 배포할 수 있습니다. 에이전트는 이 디렉터리에 있는 모든 ProbeBuilder 목록(PBL)은 무시합니다.

잘못된 PBD 파일이 자동으로 배포되지 않도록 하려면 *hotdeploy* 디렉터리를 사용하지 않도록 설정할 수 있습니다.

hotdeploy 디렉터리를 사용하지 않게 설정하려면

1. *hotdeploy* 디렉터리에 저장되어 있는 모든 사용자 지정 PBD 파일을 *<Agent_Home>* 디렉터리로 이동합니다.
2. *IntroscopeAgent.profile* 을 텍스트 편집기에서 엽니다.
3. *introscope.autoprobe.directivesFile* 속성에서 *hotdeploy* 를 제거합니다.
4. 사용할 사용자 지정 PBD 를 *introscope.autoprobe.directivesFile* 에 추가합니다. 예:
   ```
   introscope.autoprobe.directivesFile=default-typical.pbl,custom1.pbd,custom2.pbd,custom3.pbd
   ```
5. *IntroscopeAgent.profile* 을 저장하고 에이전트를 다시 시작합니다.

부록 A: .NET 에이전트 속성

이 섹션은 아래의 주제를 포함하고 있습니다:

.NET 에이전트와 Enterprise Manager 사이의 연결

에이전트가 Enterprise Manager 에 연결하는 방법을 제어할 수 있습니다.

introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT

에이전트가 기본적으로 연결되는 Enterprise Manager 를 실행하는 컴퓨터의 호스트 이름을 지정합니다.

기본값

localhost

예

```
introscope.agent.enterprisemanager.transport.tcp.host.DEFAULT=localhost
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT

Enterprise Manager 를 호스팅하는 컴퓨터에서 에이전트로부터의 연결을 수신 대기하는 포트 번호를 지정합니다.

기본값

기본 포트는 구성하려는 통신 채널의 유형에 따라 달라집니다. 에이전트와 Enterprise Manager 간의 직접 통신에 사용되는 기본 포트는 5001 입니다.

예

```
introscope.agent.enterprisemanager.transport.tcp.port.DEFAULT=5001
```

참고

- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.
- HTTPS(HTTP over SSL)를 사용하여 Enterprise Manager 에 연결하려는 경우 기본 포트는 8444 입니다. SSL 을 사용하여 Enterprise Manager 에 연결하려는 경우 기본 포트는 5443 입니다. 그러나 이러한 기본 설정은 기본적으로 주석으로 처리되어 있습니다.

introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT

에이전트에서 Enterprise Manager 에 연결할 때 사용할 기본 클라이언트 소켓 팩터리를 지정합니다.

기본값

기본 소켓 팩터리는 구성하려는 통신 채널의 유형에 따라 달라집니다. 에이전트와 Enterprise Manager 간의 직접 통신에 사용되는 기본 소켓 팩터리는 다음과 같습니다.

com.wily.isengard.postofficehub.link.net.DefaultSocketFactory

예

```
introscope.agent.enterprisemanager.transport.tcp.socketfactory.DEFAULT=com.wily.isengard.postofficehub.link.net.DefaultSocketFactory
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

에이전트 장애 조치

에이전트와 기본 Enterprise Manager 사이의 연결이 끊어질 경우 다음 속성은 에이전트가 연결을 시도하는 백업 Enterprise Manager 와 에이전트가 기본 Enterprise Manager 에 연결을 다시 시도하는 빈도를 지정합니다.

introscope.agent.enterprisemanager.connectionorder

에이전트와 기본 Enterprise Manager 간의 연결이 끊어질 경우 에이전트에서 사용하는 백업 Enterprise Manager 의 연결 순서를 지정합니다.

속성 설정

에이전트가 연결할 수 있는 다른 Enterprise Manager 의 이름

기본값

DEFAULT 호스트 이름, 포트 번호 및 소켓 팩터리 속성으로 정의된 Enterprise Manager

예

```
introscope.agent.enterprisemanager.connectionorder=DEFAULT
```

참고

- 쉼표로 구분된 목록을 사용하십시오.
- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds

거부된 에이전트가 다음 Enterprise Manager 에 다시 연결하려고 하는 각 시도 사이의 시간 간격(초)을 지정합니다.

- 에이전트 프로필 introscope.agent.enterprisemanager.connectionorder 속성 값에 구성된 순서를 기반으로 하는 Enterprise Manager
- loadbalancing.xml 구성을 기반으로 하는 허용되는 모든 Enterprise Manager

에이전트는 Enterprise Manager 에 연결할 수 없는 경우 다음과 같은 방식으로 연결을 처리합니다.

- 허용되는 다음 Enterprise Manager 에 연결을 시도합니다.
- 허용되지 않는 Enterprise Manager 에는 연결하지 않습니다.

참고: loadbalancing.xml 구성 및 에이전트와 Enterprise Manager 간의 연결 구성에 대한 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

기본값

기본 간격은 120 초입니다.

예

```
introscope.agent.enterprisemanager.failbackRetryIntervalInSeconds
=120
```

참고

이 속성의 변경 사항이 적용되도록 관리되는 응용 프로그램을 다시 시작하십시오.

이 속성은 기본적으로 주석으로 처리되어 있습니다.

이 속성은 에이전트가 다음 CA APM 구성 요소를 통과하여 연결할 수 있는 환경에 유용합니다.

- 클러스터
- 수집기 및 독립 실행형 Enterprise Manager
- 클러스터, 수집기 및 독립 실행형 Enterprise Manager 의 조합

다음 예에서는 에이전트가 다른 클러스터의 Enterprise Manager 에 연결할 수 있고 이 속성이 설정되어 있지 않은 경우에 발생하는 결과를 보여 줍니다.

1. 클러스터의 Enterprise Manager 에 연결된 에이전트가 연결이 끊어집니다.
2. 에이전트는 클러스터 2 의 Enterprise Manager 에 허용되지 않는 모드로 연결합니다.
3. 에이전트는 클러스터 1 의 허용되는 Enterprise Manager 를 사용할 수 있게 되는 시기를 알 수 없습니다.

이 속성은 Enterprise Manager 에 연결할 수 있게 될 때까지 에이전트가 허용되는 Enterprise Manager 에 계속 연결을 시도하도록 합니다.

에이전트 메트릭 만료 처리

에이전트 메트릭 만료 처리는 사용되지 않는 메트릭을 주기적으로 에이전트 메모리 캐시에서 제거합니다. 사용되지 않는 메트릭은 구성된 시간 동안 새 데이터를 보고하지 않은 메트릭입니다. 오래된 메트릭을 제거하면 에이전트 성능을 향상하고 발생할 수 있는 메트릭 급증을 방지할 수 있습니다.

참고: 시스템이 처리할 수 있는 것보다 많은 메트릭을 보고하도록 부주의하게 에이전트를 설정하는 경우 메트릭이 급증합니다. 너무 많은 메트릭이 보고되면 응용 프로그램 서버의 성능에 영향을 미칠 수 있고 심한 경우 서버가 전혀 작동하지 않게 될 수도 있습니다.

그룹에 속한 메트릭은 그룹의 모든 메트릭이 제거 후보에 해당하는 경우에만 제거됩니다. 현재 *BlamePointTracer* 및 *MetricRecordingAdministrator* 메트릭만 그룹으로 제거됩니다. 다른 메트릭은 개별적으로 제거됩니다.

MetricRecordingAdministrator 에는 메트릭 그룹을 생성, 검색 또는 제거하기 위한 다음과 같은 인터페이스가 있습니다.

- *getAgent().IAgent_getMetricRecordingAdministrator.addMetricGroup*

  문자열 구성 요소, 수집 메트릭. 구성 요소 이름은 메트릭 그룹의 메트릭 리소스 이름입니다. 동일한 메트릭 노드 아래에 매트릭이 있어야만 그룹으로 간주됩니다. 메트릭은 *com.wily.introscope.spec.metric.AgentMetric* 데이터 구조의 모음입니다. 이 모음에는 *AgentMetric* 데이터 구조만 추가할 수 있습니다.

- *getAgent().IAgent_getMetricRecordingAdministrator.getMetricGroup*

  문자열 구성 요소. 메트릭 리소스 이름인 구성 요소 이름을 기준으로 메트릭의 모음을 가져올 수 있습니다.

- *getAgent().IAgent_getMetricRecordingAdministrator.removeMetricGroup*

  문자열 구성 요소. 메트릭 그룹은 메트릭 리소스 이름인 구성 요소를 기반으로 제거됩니다.

- *getAgent().IAgent_getDataAccumulatorFactory.isRemoved*

  메트릭이 제거되었는지 확인합니다. 확장에 있는 누산기의 인스턴스를 유지하려는 경우 이 인터페이스를 사용합니다. 메트릭 만료 처리로 인해 누산기가 제거되면 이 인터페이스를 통해 사용되지 않는 참조를 유지하지 않도록 합니다.

중요! 다른 CA Technologies 제품에서 사용하기 위한 용도 등으로 *MetricRecordingAdministrator* 인터페이스를 사용하는 확장을 생성하는 경우 사용자 고유의 누산기 인스턴스를 제거해야 합니다. 호출되지 않았다는 이유로 메트릭이 만료 처리되면 나중에 해당 메트릭의 데이터를 사용할 수 있게 되어도 이전 누산기 인스턴스가 새 메트릭 데이터 포인트를 생성하지 않습니다. 이러한 상황을 방지하려면 사용자 고유의 누산기 인스턴스를 삭제하지 말고 대신 *getDataAccumulatorFactory* 인스턴스를 사용하십시오.

에이전트 메트릭 만료 처리 구성

에이전트 메트릭 만료 처리는 기본적으로 사용됩니다. 이 기능을 해제하려면 introscope.agent.metricAging.turnOn (see page 174) 속성을 사용합니다. 이 속성을 *IntroscopeAgent.profile* 에서 제거하면 에이전트 메트릭 만료 처리가 기본적으로 해제됩니다.

에이전트 메트릭 만료 처리는 에이전트의 하트비트에 실행됩니다. 하트비트는 introscope.agent.metricAging.heartbeatInterval (see page 174) 속성을 사용하여 구성됩니다. 하트비트 빈도를 낮게 유지해야 합니다. 하트비트가 높으면 에이전트와 CA Introscope®의 성능에 영향을 미칩니다.

각 하트비트 동안 특정 메트릭 집합이 확인됩니다. 이 집합은 introscope.agent.metricAging.dataChunk (see page 175) 속성을 사용하여 구성할 수 있습니다. 이 값도 낮게 유지해야 합니다. 이 값이 높으면 성능에 영향을 줍니다. 기본값은 하트비트당 메트릭을 500 개 확인합니다. 메트릭 500 개 각각을 확인하여 제거 후보에 해당하는지 확인합니다. 예를 들어 이 속성을 하트비트당 메트릭 500 개 단위의 청크를 확인하도록 설정하는 경우 에이전트 메모리의 메트릭이 총 10,000 개 있으면 메트릭 10,000 개를 모두 확인하느라 성능이 저하되고 시간이 오래 걸립니다. 그러나 이 속성을 더 높은 값으로 설정하면 모든 10,000 개 메트릭을 더 빨리 확인하지만 오버헤드가 높아집니다.

특정 기간 후 메트릭에 새 데이터가 수신되지 않은 경우 메트릭은 제거 후보입니다. 이 기간은 introscope.agent.metricAging.numberTimeslices (see page 175) 속성을 사용하여 구성할 수 있습니다. 이 속성은 기본적으로 3000 으로 설정되어 있습니다. 메트릭이 제거 조건을 충족하면 해당 그룹의 모든 메트릭이 메트릭 제거 후보인지 확인합니다. 이 요구 사항도 충족하면 메트릭이 제거됩니다.

introscope.agent.metricAging.turnOn

에이전트 메트릭 만료 처리를 설정하거나 해제합니다.

속성 설정

True 또는 False

기본값

True

예

```
introscope.agent.metricAging.turnOn=true
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.metricAging.heartbeatInterval

제거할 메트릭을 확인하는 시간 간격을 초 단위로 지정합니다.

기본값

1800

예

```
introscope.agent.metricAging.heartbeatInterval=1800
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.metricAging.dataChunk

각 간격에서 확인할 메트릭의 수를 지정합니다.

기본값

500

예

```
introscope.agent.metricAging.dataChunk=500
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.metricAging.numberTimeslices

이 속성은 데이터를 제거 대상으로 만들기 전에 새 데이터가 없는지 확인하는 간격의 수를 지정합니다.

기본값

3000

예

```
introscope.agent.metricAging.numberTimeslices=3000
```

참고

이 속성에 대한 변경 내용은 즉시 적용되므로 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.metricAging.metricExclude.ignore.0

지정한 메트릭을 제거하지 않도록 제외합니다. 만료 처리에서 하나 이상의 메트릭을 제외하려면 목록에 메트릭 이름이나 메트릭 필터를 추가합니다.

속성 설정

메트릭을 쉼표로 구분한 목록입니다. 메트릭 이름에서 별표(*)를 와일드카드로 사용할 수 있습니다.

기본값

기본값은 Threads 로 시작하는 메트릭 이름(*Threads**)입니다.

예

```
introscope.agent.metricAging.metricExclude.ignore.0=Threads*
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

에이전트 메트릭 클램프

에이전트가 Enterprise Manager 로 보내지는 메트릭의 수를 대략적으로 클램프하도록 구성할 수 있습니다. 생성된 메트릭 수가 속성 값을 초과하면 에이전트는 새 메트릭의 수집 및 전송을 중지합니다.

introscope.agent.metricClamp

에이전트가 Enterprise Manager 로 보내지는 메트릭의 수를 대략적으로 클램프하도록 구성합니다.

기본값

5000

예

```
introscope.agent.metricClamp=5000
```

참고

- 이 속성이 설정되어 있지 않으면 메트릭 클램프가 적용되지 않습니다. 즉, 오래된 메트릭도 여전히 값을 보고합니다.
- 이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.
- 이 클램프 속성은 apm-events-thresholds-config.xml 파일에 있는 introscope.enterprisemanager.agent.metrics.limit 속성과 함께 작동합니다.

  참고: introscope.enterprisemanager.agent.metrics.limit 속성에 대한 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

introscope.enterprisemanager.agent.metrics.limit 클램프 값이 introscope.agent.metricClamp 값보다 먼저 트리거되면 Enterprise Manager 가 에이전트 메트릭을 읽기는 하지만 Investigator 메트릭 브라우저 트리에서 해당 메트릭을 보고하지는 않습니다.

introscope.agent.metricClamp 클램프 값이 introscope.enterprisemanager.agent.metrics.limit 클램프 값보다 먼저 트리거되면 에이전트에서 Enterprise Manager 로의 메트릭 전송이 중지됩니다.

introscope.agent.simpleInstanceCounter.referenceTrackingLimit

SimpleInstanceCounter 메트릭은 각 클래스에 생성되는 인스턴스 수를 추적합니다. 인스턴스 수가 너무 많으면 .NET 에이전트의 오버헤드가 크게 증가할 수 있습니다. 이 속성을 사용하면 각 클래스의 추적된 인스턴스 수를 제한할 수 있습니다.

옵션

정수

기본값

25000

예

```
introscope.agent.simpleInstanceCounter.referenceTrackingLimit
=25000
```

참고

이 속성에 대한 변경 사항은 즉시 적용되므로 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

에이전트 메모리 오버헤드

상당한 에이전트 메모리 오버헤드는 극단적인 몇몇 경우에만 발생합니다. 일반적으로 메모리 소비를 줄이면 대신 응답 시간이 늘어날 수 있습니다. 그러나 각 응용 프로그램은 고유하며 메모리 사용량과 응답 시간 사이의 균형도 응용 프로그램 자체에 따라 달라질 수 있습니다.

introscope.agent.reduceAgentMemoryOverhead

이 속성은 사용할 에이전트 구성을 지정합니다. 에이전트 메모리 오버헤드를 줄이려면 주석 처리를 제거하십시오. 이 속성은 기본적으로 true 로 설정되고 주석 처리되어 있습니다.

속성 설정

True 또는 False

기본값

True

예

```
introscope.agent.reduceAgentMemoryOverhead=true
```

참고

이 속성의 변경 내용이 적용되도록 관리되는 응용 프로그램을 다시 시작하십시오.

에이전트 이름 지정

에이전트 및 로그 파일의 자동 이름 지정 및 이름 지정과 관련된 작업을 제어할 수 있습니다.

introscope.agent.agentAutoNamingEnabled

에이전트 자동 이름 지정을 사용하여 지원되는 응용 프로그램 서버의 .NET 에이전트 이름을 가져올지 여부를 지정합니다.

옵션

True 또는 False

기본값

True

예

```
introscope.agent.agentAutoNamingEnabled=true
```

참고

- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.
- WebLogic 의 경우 시작 클래스를 지정하고 WebSphere 의 경우 사용자 지정 서비스를 지정해야 합니다.
- 지원되는 응용 프로그램 서버와 함께 제공된 에이전트 프로필에는 true 로 설정하고 주석 처리를 제거하지 않아야 합니다.

introscope.agent.agentAutoNamingMaximumConnectionDelayInSeconds

에이전트가 Enterprise Manager 에 연결하기 전에 이름 지정 정보를 가져오기 위해 기다리는 최대 시간(초)을 지정합니다.

속성 설정

초 단위의 지연 시간을 나타내는 양의 정수입니다.

기본값

기본값은 120 초입니다.

예

```
introscope.agent.agentAutoNamingMaximumConnectionDelayInSeconds=120
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.agentAutoRenamingIntervalInMinutes

에이전트 이름이 변경되었는지 여부를 확인하기 위해 에이전트를 검사하는 빈도를 지정합니다. 지정된 간격마다 에이전트는 이름이 변경되었는지 확인합니다.

속성 설정

해당 간격의 시간(분)을 나타내는 양의 정수입니다.

기본값

기본 간격은 10 분입니다.

예

```
introscope.agent.agentAutoRenamingIntervalInMinutes=10
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.disableLogFileAutoNaming

에이전트 로그 파일의 자동 이름 지정을 사용할지 여부를 지정합니다. 기본적으로 에이전트가 자동 이름 지정을 사용하도록 구성된 경우, 해당 로그 파일 이름도 에이전트 이름 또는 타임스탬프를 사용하여 자동으로 지정됩니다. 이 속성을 true 로 설정하면 기본 동작이 사용되지 않습니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.disableLogFileAutoNaming=false
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.agentName

다른 모든 에이전트 이름 지정 방법이 실패하는 경우 에이전트에 사용할 이름을 지정합니다. 이 속성 값이 유효하지 않거나 프로필에서 속성이 삭제된 경우 기본 에이전트 이름은 *Unknown Agent* 입니다. 이 속성은 기본적으로 주석으로 처리되어 있습니다. 이 속성의 주석 처리를 제거하여 다른 에이전트 이름 지정 방법이 실패할 때 사용할 기본 에이전트 이름을 지정할 수 있습니다.

속성 설정

다른 모든 방법으로 에이전트 이름을 결정할 수 없을 때 에이전트 이름으로 사용할 텍스트 문자열입니다.

기본값

AgentName

예

```
introscope.agent.agentName=AgentName
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.clonedAgent

동일한 컴퓨터에서 응용 프로그램의 동일한 복사본을 실행할 수 있도록 합니다. 동일한 컴퓨터에서 응용 프로그램의 동일한 복사본이 실행 중인 경우에는 이 속성을 true 로 설정하십시오.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.clonedAgent=false
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.display.hostName.as.fqdn

이 속성은 에이전트 이름이 FQDN(정규화된 도메인 이름)으로 표시되는지 여부를 지정합니다. FQDN 을 사용하도록 설정하려면 이 속성 값을 'true'로 설정하십시오. 기본적으로 에이전트는 호스트 이름을 표시합니다.

참고: Catalyst 가 통합된 경우 이 속성을 'true'로 설정하십시오.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.display.hostName.as.fqdn=false
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

에이전트 기록(비즈니스 기록)

에이전트에서 비즈니스 트랜잭션 기록을 처리하는 방식을 제어할 수 있습니다.

참고: 에이전트 비즈니스 기록에 대한 자세한 내용은 *CA APM 트랜잭션 정의 안내서*를 참조하십시오.

introscope.agent.bizRecording.enabled

에이전트의 비즈니스 트랜잭션 기록을 사용하거나 사용하지 않도록 설정합니다.

속성 설정

True 또는 False

기본값

True

예

```
introscope.agent.bizRecording.enabled=true
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

에이전트의 비즈니스 트랜잭션 기록을 보다 세부적으로 구성하려면 응용 프로그램 심사 맵에 대한 추가 속성을 참조하십시오.

추가 정보:

응용 프로그램 심사 맵 (페이지 187)
응용 프로그램 심사 맵 비즈니스 트랜잭션 POST 매개 변수 (페이지 195)

에이전트 스레드 우선 순위

에이전트 스레드의 우선 순위를 제어할 수 있습니다.

introscope.agent.thread.all.priority

에이전트 스레드의 우선 순위를 지정합니다.

속성 설정

값은 0(낮음)부터 4(높음)까지 설정할 수 있습니다

기본값

기본 우선 순위는 2 입니다.

예

```
introscope.agent.thread.all.priority=2
```

응용 프로그램 심사 맵

응용 프로그램 심사 맵 데이터를 구성할 수 있습니다.

참고: 응용 프로그램 심사 맵을 사용하는 방법에 대한 자세한 내용은 *CA APM Workstation 사용자 안내서*를 참조하십시오.

introscope.agent.appmap.enabled

응용 프로그램 심사 맵에 대한 모니터링된 코드의 추적을 활성화 또는 비활성화합니다.

속성 설정

True 또는 False

기본값

True

예

```
introscope.agent.appmap.enabled=true
```

참고

기본적으로 사용하도록 설정됩니다.

introscope.agent.appmap.metrics.enabled

응용 프로그램 심사 맵 노드에 대한 메트릭의 추적을 활성화 또는 비활성화합니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.appmap.metrics.enabled=false
```

참고

이 속성은 기본적으로 주석으로 처리되어 있습니다.

introscope.agent.appmap.queue.size

응용 프로그램 심사 맵에 대한 버퍼 크기를 설정합니다.

속성 설정

양의 정수

기본값

1000

예

```
introscope.agent.appmap.queue.size=1000
```

참고

- 값은 양의 정수여야 합니다.
- 값을 0 으로 설정하면 버퍼에 제한이 없습니다.
- 이 속성은 기본적으로 주석으로 처리되어 있습니다.

introscope.agent.appmap.queue.period

응용 프로그램 심사 맵 데이터를 Enterprise Manager 로 보내는 빈도(밀리초)를 설정합니다.

속성 설정

양의 정수

기본값

1000

예

```
introscope.agent.appmap.queue.period=1000
```

참고

- 반드시 양의 정수여야 합니다.
- 값이 0 으로 설정되면 기본값이 사용됩니다.
- 이 속성은 기본적으로 주석으로 처리되어 있습니다.

introscope.agent.appmap.intermediateNodes.enabled

응용 프로그램의 프런트엔드 노드와 백엔드 노드 사이에 중간 노드를 포함하는 기능을 사용하거나 사용하지 않도록 설정합니다.

속성 설정

True 또는 False

기본값

False

예

```
#introscope.agent.appmap.intermediateNodes.enabled=true
```

참고

- 이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.
- 이 속성을 true 로 설정하면 에이전트 성능이 저하될 수 있습니다.
- 이 속성은 기본적으로 주석으로 처리되어 있습니다.

응용 프로그램 심사 맵 및 Catalyst 통합

Catalyst 통합에 맞게 응용 프로그램 심사 맵을 구성할 수 있습니다.

참고: 응용 프로그램 심사 맵을 사용하는 방법에 대한 자세한 내용은 *CA APM Workstation 사용자 안내서*를 참조하십시오.

정보 전송 기능 구성

이 속성은 Catalyst 와의 통합을 위해 에이전트에서 추가 정보를 보내는 기능을 사용하거나 사용하지 않도록 설정합니다.

다음 단계를 따르십시오.

1. 기본 IntroscopeAgent.profile 파일을 텍스트 편집기에서 엽니다.

   introscope.agent.appmap.catalystIntegration.enabled=<false|true> 행을 찾아 값을 다음과 같이 설정합니다.

   true

   Catalyst 와의 통합을 위해 에이전트에서 추가 정보를 보내는 기능을 사용하도록 설정합니다.

   false

   이 구성을 사용하지 않도록 설정합니다.

   다음 예에서는 형식을 보여 줍니다.

   ```
   introscope.agent.appmap.catalystIntegration.enabled=false
   ```

   참고: 이 속성은 기본적으로 주석으로 처리되어 있습니다.

2. 파일을 저장하고 닫습니다.

   에이전트가 이 구성을 사용하도록 설정됩니다.

사용 가능한 네트워크 목록 구성

introscope.agent.primary.net.interface.name 속성은 Catalyst 통합을 위해 에이전트에서 사용하는 호스트 컴퓨터의 기본 네트워크 인터페이스 이름을 지정합니다. 이 속성의 구성을 변경할 수 있으며 변경 사항은 자동으로 적용됩니다.

참고: 에이전트 로깅 수준이 DEBUG 로 설정된 경우 구성에 사용할 수 있는 네트워크 인터페이스 이름에 대한 정보가 로그 파일에 나타납니다. 또는 이 속성에 대해 기본 네트워크 인터페이스 이름을 결정하기 위해 네트워크 인터페이스 유틸리티를 사용할 수 있습니다.

다음 단계를 따르십시오.

1. 기본 IntroscopeAgent.profile 파일을 텍스트 편집기에서 엽니다.
2. introscope.agent.primary.net.interface.name=<false|true> 줄을 찾고 이름 값을 지정합니다.

   다음 예제는 이름 형식을 나타냅니다.

   ```
   introscope.agent.primary.net.interface.name=eth4
   ```

   참고: 기본 값은 정의되어 있지 않습니다. 이 속성이 설정되지 않은 경우 에이전트는 첫 번째 사용 가능한 네트워크 인터페이스를 기본 인터페이스로 지정합니다. 이 속성의 이름 값을 결정하기 위해 네트워크 인터페이스 유틸리티를 사용할 수 있습니다.
3. (선택 사항) 하위 인터페이스 번호(0 부터 시작)를 지정하여 여러 네트워크 주소를 사용할 수 있습니다.

   다음 예제는 하위 인터페이스 번호 형식을 나타냅니다.

   ```
   introscope.agent.primary.net.interface.name=eth4.1
   ```
4. 파일을 저장하고 닫습니다.

   프로필이 구성을 사용하도록 설정됩니다.

추가 정보:

네트워크 인터페이스 유틸리티 사용 (페이지 277)

응용 프로그램 심사 맵 비즈니스 트랜잭션 POST 매개 변수

POST 매개 변수와 일치시켜 보다 복잡한 모니터링을 수행할 수 있도록 Local Product Shorts 를 구성할 수 있습니다.

introscope.agent.bizdef.matchPost

이 속성은 POST 매개 변수와 일치되는 시기를 결정합니다.

속성 설정

이 속성의 유효한 설정은 *never*, *before* 또는 *after* 입니다.

- 에이전트 기능을 모두 사용하고 성능을 향상시키려는 경우에는 이 속성을 **never** 로 설정하십시오. 이 설정을 사용하면 응용 프로그램에서 URL, 쿠키 또는 헤더 매개 변수를 사용하여 모든 비즈니스 트랜잭션을 식별할 수 있지만, POST 매개 변수를 통해서만 식별되는 비즈니스 트랜잭션은 일치시킬 수 없습니다.

- 최대의 에이전트 성능을 얻으려면 이 속성을 **before** 로 설정하십시오. 이 설정을 사용하면 응용 프로그램에서 POST 매개 변수를 사용하여 일부 또는 모든 비즈니스 트랜잭션을 식별할 수 있지만, HTTP 양식 요청을 위해 서블릿 스트림에 직접 액세스할 수는 없습니다. 이 속성이 *before* 로 설정되어 있으면 새로 배포되는 응용 프로그램에서도 표준 API 를 따라야 합니다.

  중요! 이 속성을 *before* 로 설정하면 응용 프로그램에 악영향을 줄 수도 있습니다. 구현 전에 CA Technologies 담당자와 함께 이 속성 설정을 검토하십시오.

- 비즈니스 트랜잭션을 POST 매개 변수와 안전하게 일치시키되 에이전트 기능을 제한하려면 이 속성을 **after** 로 설정하십시오. 이 속성이 *after* 로 설정되어 있으면 에이전트는 프로세스에서 POST 매개 변수로 식별된 비즈니스 트랜잭션을 매핑하거나 이러한 비즈니스 트랜잭션에 대한 전체 메트릭 집합을 생성할 수 없습니다. 또한 이 설정은 다른 옵션에 비해 CPU 시간을 좀 더 소비하지만, POST 매개 변수 기능이 필요한 경우에는 가장 안전한 설정으로 간주됩니다. 이 설정을 사용하면 응용 프로그램에서 POST 매개 변수를 사용하여 일부 또는 모든 비즈니스 트랜잭션을 식별할 수 있지만, 서블릿 스트림에 대한 직접 액세스를 항상 금지할 수는 없습니다.

예

```
introscope.agent.bizdef.matchPost=after
```

참고

- *never* - POST 매개 변수를 일치시키지 않습니다. 이는 가장 빠른 옵션이지만 비즈니스 트랜잭션 구성 요소 일치가 부정확하게 될 수 있습니다.
- *before* - 서블릿이 실행되기 전에 POST 매개 변수를 일치시킵니다.
- *after* - 서블릿이 실행된 후에 POST 매개 변수 패턴을 일치시킵니다. 크로스 프로세스 매핑과 일부 메트릭은 사용할 수 없습니다. 이 매개 변수의 기본 설정입니다.

알려진 제한 사항

에이전트 기록을 사용하여 정의된 메트릭은 Investigator 의 응용 프로그램 심사 맵에 표시됩니다. 에이전트 기록을 구성할 경우 정규식 사용과 관련하여 몇 가지 알려진 제한 사항이 있습니다. 대부분의 제한 사항은 POST 매개 변수와 관련이 있습니다.

알려진 제한 사항은 다음과 같습니다.

- POST 매개 변수 값에는 행 마침 표시(.)를 사용할 수 없습니다.
- POST 매개 변수 정의가 비즈니스 트랜잭션 정의에 종속된 경우 비즈니스 트랜잭션 구성 요소에 세 가지 메트릭만 제공됩니다. 해당 메트릭은 다음과 같습니다.
  - 평균 응답 시간
  - 간격당 응답 수
  - 간격당 오류 수
- POST 매개 변수 정의가 비즈니스 트랜잭션 정의에 종속된 경우 트랜잭션 추적 구성 요소의 비즈니스 구성 요소 이름은 비즈니스 서비스, 비즈니스 트랜잭션 및 비즈니스 트랜잭션 구성 요소의 특정 이름이 아니라 일반 이름입니다. 이 내용은 일치하지 않는 POST 매개 변수 정의에 종속된 비즈니스 트랜잭션 정의에도 적용됩니다.
- JBoss 및 Tomcat 의 일부 버전에서는 헤더 키를 소문자 값으로 저장하여 *caseSensitiveName* 특성이 *HEADER_TYPE* 에 대해 올바로 작동하지 않게 될 수 있습니다.

참고: 에이전트 기록에 대한 자세한 내용은 *CA APM 트랜잭션 정의 안내서*를 참조하십시오.

AutoProbe

에이전트가 AutoProbe 와 상호 작용하는 방식을 제어할 수 있습니다.

중요! 다음 속성은 필수 매개 변수입니다. AutoProbe 속성을 설정하지 않거나 값이 올바르지 않으면 Introscope 가 제대로 작동하지 않습니다.

introscope.autoprobe.directivesFile

배포할 ProbeBuilder 지시문 파일(*.pbd*) 및 ProbeBuilder 목록 파일(*.pbl*)을 지정합니다. 이 속성에 대해 나열된 파일에 따라 계측 대상 구성 요소가 결정됩니다. 이 속성은 필수 항목입니다.

속성 설정

단일 항목 또는 쉼표로 구분된 항목 목록입니다. 목록에는 다음 정보의 조합이 포함될 수 있습니다.

- ProbeBuilder 지시문 파일(*.pbd*) 이름
- ProbeBuilder 목록 파일(*.pbl*) 이름
- *hotdeploy* 디렉터리 이름(*hotdeploy* 디렉터리에 있는 *.pbd* 파일은 에이전트 프로필을 편집하지 않아도 자동으로 로드됨)

목록에 추가하는 파일 이름의 절대 경로를 지정하거나, *IntroscopeAgent.profile* 파일의 위치에 상대적인 경로를 사용하여 파일 이름을 지정할 수 있습니다.

기본값

기본 항목은 에이전트를 설치할 때 선택한 옵션에 따라 달라집니다.

예

```
introscope.autoprobe.directivesFile=default-full.pbl,hotdeploy
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.autoprobe.enable

이 속성이 false 로 설정되는 경우 다음과 같은 조건이 존재합니다.

- AutoProbe 가 비활성화되었습니다.
- .NET 에이전트가 Enterprise Manager 에 연결하지 않습니다.
- .NET 에이전트가 Investigator 에 표시되지 않습니다.
- .NET 에이전트가 메트릭을 보고하지 않습니다.

속성 설정

옵션

true 또는 false

기본값

true

예

```
introscope.autoprobe.enable=true
```

참고: 변경 내용을 적용하려면 JVM 을 다시 시작해야 합니다.

introscope.autoprobe.logfile

계측 로그 파일의 이름과 위치입니다. 로그 파일의 위치를 기본값 이외의 다른 위치로 이동하려면 이 속성을 설정하십시오.

속성 설정

계측 로그 파일의 절대 또는 상대 경로입니다.

기본값

기본 위치는 *<Agent_Home>* 디렉터리에 상대적인 *logs\AutoProbe.log* 입니다.

예

```
introscope.autoprobe.logfile=logs/AutoProbe.log
```

로깅을 사용하지 않도록 설정하려면 로그 파일에서 다음과 같이 주석 처리하십시오.

```
introscope.autoprobe.logfile=logs/AutoProbe.log
```

참고

- 이 속성의 변경 사항을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

CA CEM 에이전트 프로필 속성

CA CEM 관련 IntroscopeAgent.profile 속성을 구성할 수 있습니다. CA Introscope 에이전트 프로필 파일은 *<Agent_Home>*\wily 디렉터리에 있습니다.

모든 CA CEM 관련 속성은 CA CEM 및 CA Introscope®와의 통합이 작동하는 데 필요한 옵션으로 미리 구성되어 있습니다.

introscope.autoprobe.directivesFile

directivesFile 속성 구성을 통해 ServletHeaderDecorator / HTTPHeaderDecorator 및 CEMTracer 를 사용하도록 설정해야 합니다.

지시문 파일 속성은 AutoProbe 용 지시문 파일(PBD) 또는 지시문 목록(PBL)을 찾을 위치를 지정합니다.

AutoProbe 는 지시문을 사용하여 응용 프로그램을 사용하도록 설정하고 에이전트가 Enterprise Manager 에 보고할 메트릭을 결정합니다.

설정

설치된 에이전트 응용 프로그램 서버에 따라 다르며, 형식은 <*appserver*>-full.pbl 또는 <*appserver*>-typical.pbl 입니다.

기본값

```
default-typical.pbl
```

예

```
introscope.autoprobe.directivesFile=weblogic-typical.pbl
```

참고

단순히 이 속성 목록의 끝에 "ServletHeaderDecorator.pbd" 또는 "httpheaderdecorator.pbd"를 추가해도 되지만 다음 단계를 따르는 것이 더 좋습니다.

1. 속성에 지정된 PBL 파일(위 예의 경우 weblogic-typical.pbl)을 찾습니다.
2. 텍스트 편집기에서 PBL 파일을 엽니다.
3. Java Agent 의 경우 ServletHeaderDecorator.pbd 행의 주석 처리를 제거하여 이 행을 활성화합니다.

4. .NET 에이전트의 경우 httpheaderdecorator.pbd 행의 주석 처리를 제거하여 이 행을 활성화합니다.
5. PBL 파일의 변경 내용을 저장합니다.

introscope.agent.remoteagentconfiguration.allowedFiles

이 속성은 임의의 컴퓨터에서 에이전트 디렉터리로 원격 복사가 허용되는 파일을 식별합니다.

Enterprise Manager 는 이 속성의 파일 이름을 사용하여 에이전트로 보낼 유효한 CA CEM 도메인 구성 파일을 식별합니다. 도메인 구성 파일은 CA CEM 비즈니스 서비스와 트랜잭션 정의를 포함하고 있습니다.

설정

올바른 파일 이름을 사용하십시오.

기본값

domainconfig.xml

예

```
introscope.agent.remoteagentconfiguration.allowedFiles=domainconfig.xml
```

참고

이 속성은 또한 Introscope CLW(Command-Line Workstation) 구성 파일 보내기 명령에도 적용됩니다.

자세한 내용은 Command-Line Workstation 사용을 참조하십시오.

이 속성은 CA CEM 릴리스에 적용됩니다.

introscope.agent.remoteagentconfiguration.enabled

이 부울 값이 true 로 설정되면 다른 컴퓨터에서 에이전트로 원격 파일 복사를 허용할 수 있습니다.

Enterprise Manager 가 CA CEM 도메인 구성 파일을 에이전트에 보내려면 이 속성을 true 로 설정해야 합니다. 도메인 구성 파일은 CA CEM 비즈니스 서비스와 트랜잭션 정의를 포함하고 있습니다.

설정

true 또는 false

기본값

- Java Agent 의 경우 true
- .NET 에이전트의 경우 false

예

```
introscope.agent.remoteagentconfiguration.enabled=true
```

참고

원격 사용자는 또한 CA Introscope 명령줄 Workstation(CLW) 구성 파일 보내기 명령을 사용하여 introscope.agent.remoteagentconfiguration.allowedFiles 속성에 지정된 파일을 에이전트 디렉터리에 복사할 수 있습니다.

이 속성은 CA CEM 4.0 및 4.1 릴리스에 대해 유효합니다. 이 속성은 또한 "Introscope 설정" 페이지에서 "CEMTracer 4.0 / 4.1 지원" 옵션이 선택된 경우 CA CEM 4.2 / 4.5 에서만 유효합니다.

"CEMTracer 4.0 / 4.1 지원" 옵션을 사용하면 시간이 지남에 따라 4.0 또는 4.1 에서 4.2 / 4.5 로 에이전트 마이그레이션을 스태거할 수 있습니다. 이 옵션은 필요한 경우에만 사용하십시오.

호환되지 않는 에이전트, 즉 지원되지 않는 .NET 에이전트, EPA 에이전트 또는 기타 Java 가 아닌 에이전트의 경우에는 introscope.agent.remoteagentconfiguration.enabled 속성을 false 로 설정하십시오.

introscope.agent.decorator.enabled

이 부울 값이 true 로 설정되어 있으면 에이전트가 HTTP 응답 헤더에 추가적인 성능 모니터링 정보를 추가하도록 구성됩니다. ServletHeaderDecorator / HTTPHeaderDecorator 는 각 트랜잭션에 GUID 를 연결하고 이 GUID 를 HTTP 헤더 x-apm-info 에 삽입합니다.

그러면 CA CEM 과 CA Introscope 간의 트랜잭션 상관 관계가 설정됩니다.

설정

true 또는 false

기본값

- Java Agent 의 경우 false
- .NET 에이전트의 경우 true

예

```
introscope.agent.decorator.enabled=false
```

introscope.agent.decorator.security

이 속성에 따라 CA CEM 으로 전송되는 데코레이트된 HTTP 응답 헤더의 형식이 결정됩니다.

설정

- clear - 일반 텍스트 인코딩
- encrypted - 헤더 데이터가 암호화됨

기본값

clear

예

```
introscope.agent.decorator.security=clear
```

참고

기본 설정 clear 는 초기 테스트에 적절합니다. 그러나 이 설정은 방화벽 바깥으로 노출하고 싶지 않은 트랜잭션 헤더의 정보를 노출할 수 있습니다. 보다 안전한 프로덕션 환경을 위해 이 속성을 encrypted 로 설정하십시오.

이 속성을 encrypted 로 설정하려면 지원되는 JVM 을 사용하십시오.

참고: JVM 지원 정보는 *Compatibility Guide*(호환성 안내서)를 참조하십시오.

introscope.agent.cemtracer.domainconfigfile

CA CEM 비즈니스 서비스 및 트랜잭션 계층을 지정하는 CA CEM 도메인 구성 파일의 이름입니다. CEMTracer 는 설치 디렉터리에서 이 이름의 파일을 찾습니다.

CA CEM 관리자가 CA CEM 에서 "모든 모니터 동기화"를 클릭할 때마다 도메인 구성 파일이 Enterprise Manager 로 푸시된 다음 연결된 각 에이전트로 푸시됩니다.

CA CEM 릴리스 관련 정보는 아래의 **참고**를 보십시오.

설정

임의의 유효한 파일 이름을 사용할 수 있습니다.

기본값

```
domainconfig.xml
```

예

```
introscope.agent.cemtracer.domainconfigfile=domainconfig.xml
```

참고

이 속성은 CA CEM 4.0 및 4.1 릴리스에 대해 유효합니다. 이 속성은 또한 "Introscope 설정" 페이지에서 "CEMTracer 4.0 / 4.1 지원" 옵션이 선택된 경우에만 CA CEM 4.2 / 4.5 에서 유효합니다.

"CEMTracer 4.0 / 4.1 지원" 옵션을 사용하면 시간이 지남에 따라 4.0 또는 4.1 에서 4.2 / 4.5 로 에이전트 마이그레이션을 스태거할 수 있습니다. 이 옵션은 필요한 경우에만 사용하십시오.

- 에이전트가 Enterprise Manager 에 연결되지 않은 경우 도메인 구성 파일이 전송되지 않습니다.

- 에이전트 디렉터리가 읽기 전용인 경우 도메인 구성 파일을 기록할 수 없습니다.
- CEMTracer 4.0 / 4.1 이 에이전트에서 활성화되지 않은 경우 전달된 이후에는 도메인 구성 파일에서 어떤 동작도 수행되지 않습니다.

introscope.agent.cemtracer.domainconfigfile.reloadfrequencyinminutes

에이전트가 도메인 구성 파일을 다시 로드하는 빈도(분)입니다. (4.0 / 4.1 에이전트는 Enterprise Manager 가 전달할 때마다 도메인 구성 파일을 자동으로 다시 로드하지 않습니다. 변경되지 않은 경우 에이전트는 이 파일을 다시 로드하지 않습니다.)

CA CEM 릴리스 관련 정보는 아래의 **참고**를 보십시오.

설정

숫자 값

기본값

1

예

```
introscope.agent.cemtracer.domainconfigfile.reloadfrequencyinminutes=1
```

참고

이 속성은 CA CEM 4.0 및 4.1 릴리스에 대해 유효합니다. 이 속성은 또한 "Introscope 설정" 페이지에서 "CEMTracer 4.0 / 4.1 지원" 옵션이 선택된 경우에만 CA CEM 4.2 / 4.5 에서 유효합니다.

"CEMTracer 4.0 / 4.1 지원" 옵션을 사용하면 시간이 지남에 따라 4.0 또는 4.1 에서 4.2 로 에이전트 마이그레이션을 스태거할 수 있습니다. 이 옵션은 필요한 경우에만 사용하십시오.

introscope.agent.distribution.statistics.components.pattern

BlamePointTracer 로부터 응답 시간 분포 정보를 수집하려면 이 속성의 주석 처리를 제거하고 속성을 편집하십시오. 이 응답 시간 정보는 평균 응답 시간 메트릭을 생성하는 데 사용할 수 있습니다.

추가 정보:

분포 통계 메트릭을 수집하도록 에이전트를 구성하는 방법 (페이지 73)

ChangeDetector 구성

로컬 에이전트가 ChangeDetector 와 함께 작동하는 방식을 제어할 수 있습니다.

참고: ChangeDetector 사용에 대한 자세한 내용은 *CA APM ChangeDetector 사용자 안내서*를 참조하십시오.

introscope.changeDetector.enable

ChangeDetector 사용 여부를 지정합니다. ChangeDetector 가 사용되도록 설정하려면 이 속성을 true 로 설정합니다. 주석 처리되어 기본적으로 false 로 설정되어 있습니다. ChangeDetector 가 사용되도록 설정한 경우 ChangeDetector 관련 추가 속성도 설정해야 합니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.changeDetector.enable=false
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.changeDetector.agentID

로컬 에이전트를 식별하기 위해 ChangeDetector 가 사용하는 텍스트 문자열을 지정합니다. 이 속성은 기본적으로 주석으로 처리되어 있습니다. ChangeDetector 를 활성화하는 경우 이 속성의 주석 처리를 제거하고 적절한 값으로 설정해야 합니다.

기본값

기본값은 *SampleApplicationName* 입니다.

예

```
introscope.changeDetector.agentID=SampleApplicationName
```

introscope.changeDetector.rootDir

ChangeDetector 파일의 루트 디렉터리를 지정합니다. 루트 디렉터리는 ChangeDetector 가 해당 로컬 캐시 파일을 생성하는 폴더입니다.

속성 설정

ChangeDetector 파일의 루트 디렉터리에 대한 전체 경로를 나타내는 텍스트 문자열입니다.

기본값

기본 경로는 *c://sw//AppServer//wily//change_detector* 입니다.

예

```
introscope.changeDetector.rootDir=c://sw//AppServer//wily//change_detector
```

참고

예제에서처럼 백슬래시를 사용하여 백슬래시 문자를 이스케이프하십시오.

introscope.changeDetector.isengardStartupWaitTimeInSec

에이전트가 시작된 후 ChangeDetector 가 Enterprise Manager 연결을 시도하기 전에 대기하는 기간을 초 단위로 지정합니다. 이 속성은 기본적으로 주석으로 처리되어 있습니다.

기본값

기본값은 15 초입니다.

예

```
introscope.changeDetector.isengardStartupWaitTimeInSec=15
```

introscope.changeDetector.waitTimeBetweenReconnectInSec

Enterprise Manager 연결을 다시 시도하기 전에 ChangeDetector 가 대기하는 기간을 초 단위로 지정합니다. 이 속성은 기본적으로 주석으로 처리되어 있습니다.

기본값

기본값은 10 초입니다.

예

```
introscope.changeDetector.waitTimeBetweenReconnectInSec=10
```

introscope.changeDetector.profile

ChangeDetector 데이터 원본 구성 파일의 절대 또는 상대 경로를 지정합니다. 이 속성은 기본적으로 주석으로 처리되어 있습니다.

기본값

기본값은 ChangeDetector-config.xml 입니다.

예

```
introscope.changeDetector.profile=CDConfig\\ChangeDetector-config
.xml
```

참고

예제에서처럼 백슬래시를 사용하여 백슬래시 문자를 이스케이프하십시오.

introscope.changeDetector.profileDir

데이터 원본 구성 파일이 들어 있는 디렉터리에 대한 절대 또는 상대 경로를 지정합니다. 이 속성이 설정된 경우 *introscope.changeDetector.profile* 속성으로 지정된 모든 파일과 함께 이 디렉터리의 모든 데이터 원본 구성 파일이 사용됩니다. 이 속성은 기본적으로 주석으로 처리되어 있습니다.

기본값

기본값은 changeDetector_profiles 입니다.

예

```
introscope.changeDetector.profileDir=c:\\CDconfig\\changeDetector_profiles
```

참고

백슬래시를 사용하여 백슬래시 문자를 이스케이프 처리하십시오.

프로세스 간 트랜잭션 추적

테일 필터로 인해 발생한 다운스트림 추적을 자동으로 수집하도록 설정할 수 있습니다. 이 속성을 사용하도록 설정한 상태에서 테일 필터가 포함된 트랜잭션 추적 세션을 오랫동안 실행하면 원치 않는 추적이 Enterprise Manager 에 필요 이상으로 전송됩니다.

introscope.agent.transactiontracer.tailfilterPropagate.enable

테일 필터가 있을 경우 다운스트림 에이전트에서 자동 추적 수집이 트리거되는지 여부를 제어합니다. 이 속성은 헤드 필터 전달로 인한 자동 다운스트림 추적의 수집에는 영향을 주지 않습니다.

속성 설정

True 또는 False

기본값

False(주석 처리됨)

예

```
introscope.agent.transactiontracer.tailfilterPropagate.enable=false
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

기본 도메인 구성

기본 도메인이 Enterprise Manager 에 연결하는 방법을 제어할 수 있습니다.

introscope.agent.dotnet.enableDefaultDomain

기본 도메인에 연결된 에이전트를 Enterprise Manager 에 연결할지 여부를 결정합니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.dotnet.enableDefaultDomain=false
```

참고

이 속성은 *IntroscopeAgent.profile* 에 추가해야만 사용할 수 있습니다.

이 속성을 true 로 설정하면 기본 도메인을 모니터링하는 에이전트도 Investigator 에 함께 보고됩니다.

동적 계측

사용자 지정 PBD 를 작성하거나 응용 프로그램 서버를 다시 시작하거나 에이전트를 다시 시작하지 않고도 클래스 및 메서드를 동적으로 계측할 수 있도록 설정할 수 있습니다.

참고: Introscope Workstation 에서 트랜잭션 추적 및 동적 계측을 사용하는 방법에 대한 자세한 내용은 *CA APM Workstation 사용자 안내서*를 참조하십시오.

introscope.agent.remoteagentdynamicinstrumentation.enabled

이 속성은 동적 계측의 원격 관리를 사용하거나 사용하지 않도록 설정합니다.

속성 설정

True 또는 False

기본값

True

예

```
introscope.agent.remoteagentdynamicinstrumentation.enabled=true
```

참고

- 변경 사항을 적용하려면 관리되는 응용 프로그램을 다시 시작합니다.
- 동적 계측은 CPU 를 많이 사용하는 작업입니다. 계측되는 클래스를 최소화하는 구성을 사용하십시오.
- 동적 계측을 사용하도록 설정한 경우 In-Process Side-by-Side 실행과 같이 동일한 프로세스 내에서 여러 CLR 를 모니터링하는 기능은 작동하지 않습니다. 자세한 내용은 com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs (페이지 237) 속성을 참조하십시오.

ErrorDetector

에이전트가 ErrorDetector 와 상호 작용하는 방식을 제어할 수 있습니다.

introscope.agent.errorsnapshots.enable

에이전트가 심각한 오류에 대한 트랜잭션 세부 정보를 캡처할 수 있도록 합니다. Introscope ErrorDetector 는 에이전트와 함께 기본적으로 설치됩니다. 오류 스냅숏을 볼 수 있도록 하려면 이 속성을 true 로 설정해야 합니다.

속성 설정

True 또는 False

기본값

True

참고

이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.errorsnapshots.throttle

에이전트가 15 초 동안 보낼 수 있는 최대 오류 스냅숏 수를 지정합니다.

기본값

10

예

```
introscope.agent.errorsnapshots.throttle=10
```

참고

이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.errorsnapshots.ignore.<index>

오류 메시지 필터를 하나 이상 지정합니다. 속성 이름에 인덱스 식별자(예: .0, .1, .2 ...)를 추가하여 필터를 필요한 수만큼 지정할 수 있습니다. 와일드카드(*)를 사용하여 지정한 조건과 일치하는 오류 메시지가 무시되도록 할 수도 있습니다. 정의한 필터와 일치하는 오류에 대해서는 오류 스냅숏이 생성되지 않으며, Enterprise Manager 로 해당 오류 이벤트가 전송되지도 않습니다.

중요! 이 속성은 SOAP 오류 메시지를 필터링하는 데 사용할 수 없습니다.

기본값

다음과 같은 예제 정의가 주석 처리되어 제공됩니다.

예

```
introscope.agent.errorsnapshots.ignore.0=*com.company.HarmlessExc
eption*
introscope.agent.errorsnapshots.ignore.1=*HTTP Error Code: 404*
```

참고

이 속성은 동적 속성입니다. 런타임 중에 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

확장

에이전트 확장의 위치를 구성할 수 있습니다.

introscope.agent.extensions.directory

에이전트가 로드할 모든 확장의 위치를 지정합니다. 디렉터리의 절대 또는 상대 경로를 지정할 수 있습니다. 절대 경로를 지정하지 않을 경우, 지정한 값은 *IntroscopeAgent.profiles* 파일의 위치를 기준으로 확인됩니다.

기본값

기본 위치는 *<Agent_Home>*/ext 디렉터리에 있는 *ext* 디렉터리입니다.

예

```
introscope.agent.extensions.directory=../ext
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

필터링된 매개 변수

필터링된 매개 변수를 수집하도록 설정할 수 있습니다.

introscope.agent.asp.disableHttpProperties

기본적으로 .NET 에이전트는 추적하는 트랜잭션의 URL 만 보고합니다. 개별 HTTP 속성을 보고하는 기능은 트랜잭션 추적이 시스템 오버헤드에 미치는 영향을 최소화하도록 제한됩니다.

필터링을 사용하려면 먼저 이 속성을 false 로 설정하여 HTTP 속성의 수집을 설정해야 합니다.

응용 프로그램 이름, 세션 ID, 컨텍스트 경로, 서버 이름, URL, 컨텍스트 경로, 정규화된 URL, HTTP 헤더, HTTP 매개 변수, HTTP 특성을 수집합니다.

속성 설정

옵션

True 또는 False

기본값

False

예

```
#introscope.agent.asp.disableHttpProperties=false
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

HTTP Header Decorator

CA CEM 의 통합 솔루션의 일부인 Servlet Header Decorator 를 사용할 수 있습니다.

introscope.agent.decorator.enabled

이 부울 값이 true 로 설정되어 있으면 에이전트가 HTTP 응답 헤더에 추가적인 성능 모니터링 정보를 추가하도록 구성됩니다.

HTTPHeaderDecorator 는 각 트랜잭션에 GUID 를 연결하고 해당 GUID 를 HTTP 헤더인 x-wily-info 에 삽입합니다. 이렇게 하면 CA CEM 과 Introscope 사이의 트랜잭션 상관 관계가 형성됩니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.decorator.enabled=false
```

LeakHunter 구성

에이전트가 LeakHunter 와 상호 작용하는 방식을 제어할 수 있습니다.

introscope.agent.leakhunter.enable

LeakHunter 기능이 사용되는지 여부를 제어합니다.
LeakHunter 를 사용하려면 값을 true 로 설정하십시오.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.leakhunter.enable=false
```

참고

- 이 옵션을 설정하면 CPU 및 메모리 사용량이 높아질 수 있습니다. 다른 메트릭을 통해 메모리 누수가 있음을 파악한 경우에만 이 기능을 사용하십시오.
- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.leakhunter.logfile.location

LeakHunter.log 파일의 위치를 제어합니다. 파일 이름의 절대 또는 상대 경로를 지정할 수 있습니다. 상대 경로는 *<Agent_Home>* 디렉터리를 기준으로 확인됩니다. LeakHunter 가 로그 파일에 데이터를 기록하지 않도록 하려면 이 값을 비워 두거나 주석 처리하십시오.

기본값

기본 경로는 *../../logs/LeakHunter.log* 입니다.이 경우 로그 파일이 *<Agent_Home>logs* 디렉터리에 배치됩니다.

예

```
introscope.agent.leakhunter.logfile.location=../../logs/LeakHunter.log
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.leakhunter.logfile.append

응용 프로그램을 다시 시작할 때 로그 파일을 대체하거나 기존 로그 파일에 정보를 추가할지 여부를 지정합니다.

속성 설정

True 또는 False

- False 는 로그 파일을 대체합니다.
- True 는 기존 로그 파일에 정보를 추가합니다.

기본값

False

예

```
introscope.agent.leakhunter.logfile.append=false
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.leakhunter.leakSensitivity

LeakHunter 누수 감지 알고리즘의 민감도 수준을 제어합니다. 민감도 설정이 높으면 보고되는 잠재 누수가 많아지고, 민감도가 낮으면 보고되는 잠재 누수가 적어집니다.

속성 설정

누수 민감도 값은 1(낮음)에서 10(높음) 사이의 양의 정수여야 합니다.

기본값

5

예

```
introscope.agent.leakhunter.leakSensitivity=5
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.leakhunter.timeoutInMinutes

LeakHunter 에서 새 잠재 누수를 찾는 데 걸리는 시간(분)을 제어합니다. 지정된 시간이 지나면 LeakHunter 는 새 잠재 누수 찾기를 중지하지만 이전에 식별한 잠재 누수는 계속 추적합니다.

속성 설정

음수가 아닌 양의 정수여야 합니다.

기본값

기본값은 120 분입니다.

예

```
introscope.agent.leakhunter.timeoutInMinutes=120
```

참고

- LeakHunter 가 항상 새 잠재 누수를 찾도록 하려면 이 값을 0 으로 설정하십시오.
- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.leakhunter.collectAllocationStackTraces

LeakHunter 가 잠재 누수에 대한 할당 스택 추적을 생성하는지 여부를 제어합니다. 이 속성을 *true* 로 설정하면 잠재 누수의 할당에 대한 보다 정확한 데이터를 얻을 수 있지만 필요한 메모리 및 CPU 오버헤드가 늘어납니다. 따라서 기본값은 *false* 입니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.leakhunter.collectAllocationStackTraces=
false
```

참고

- 이 속성을 *true* 로 설정하면 CPU 사용량 및 메모리와 관련하여 시스템 오버헤드가 높아질 수 있습니다.
- 이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.leakhunter.ignore.0

ignore 속성을 하나 이상 사용하여 LeakHunter 가 잠재 누수를 검사할 때 특정 컬렉션을 무시하도록 지정할 수 있습니다. 일반 컬렉션의 경우 *system.Collections.Generic.List`1* 과 같이 일반 유형의 자격이 포함된 구문을 사용합니다.사용자가 지정한 조건과 일치하는 컬렉션이 무시됩니다.

속성 설정

패턴과 일치하는 클래스의 쉼표로 구분된 목록입니다.

기본값

없음

예

```
#introscope.agent.leakhunter.ignore.0=
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

Logging(로깅)

속성을 사용하여 로깅을 구성할 수 있습니다.

introscope.agent.log.config.path

이 속성은 Log4Net 구성 파일을 가리킵니다.

속성 설정

기본값

```
logging.config.xml
```

예

```
introscope.agent.log.config.path=logging.config.xml
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

NativeProfiler

다음 속성을 *IntroscopeAgent.profile* 에 설정하여 NativeProfiler 작업을 제어할 수 있습니다. 이러한 속성은 Native Image Generator 를 사용하여 만든 응용 프로그램의 모니터링, 클래스 이름을 저장할 메모리 내 캐시의 크기 및 NativeProfiler 로그 파일의 위치와 내용을 제어합니다.

introscope.nativeprofiler.clrv4.transparency.checks.disabled

.NET 4 CLR 은 프로파일러에 의해 계측된 코드를 무료화할 수 있는 transparent 어셈블리를 검사합니다. 이 검사를 비활성화하려면 이 속성의 값을 " true" 로 설정하십시오.

속성 설정

true 또는 false

기본값

true

예

```
#introscope.nativeprofiler.clrv4.transparency.checks.disable=true
```

참고: 이 값을 적용하려면 IIS 를 다시 설정해야 합니다.

introscope.nativeprofiler.logfile

NativeProfiler 가 읽는 지시문 및 NativeProfiler 가 계측하는 메서드에 대한 정보를 기록하는 로그 파일의 경로를 설정합니다.

속성 설정

파일 위치의 절대 또는 상대 경로입니다.

기본값

logs/nativeprofiler.log

예

```
introscope.nativeprofiler.logfile=logs/nativeprofiler.log
```

참고

NativeProfiler 는 읽을 대상 PBD 및 PBL 에서 활성 상태의 모든 지시문에 대한 정보를 기록하고, 계측된 구체적인 메서드도 참조용으로 기록합니다.

introscope.nativeprofiler.logBytecode

NativeProfiler 가 계측된 바이트 코드를 나열할지 여부를 결정합니다. 이 속성이 true 로 설정되어 있으면 NativeProfiler 로그 파일에 계측된 바이트 코드가 나열됩니다. 이 속성은 기본적으로 false 로 설정되고 주석 처리되어 있습니다.

속성 설정

True 또는 False

기본값

False

예

```
#introscope.nativeprofiler.logBytecode=false
```

참고

이 속성은 기본적으로 주석으로 처리되어 있습니다.

introscope.nativeprofiler.logAllMethodsNoticed

이 속성이 사용되도록 설정된 경우 계측되지 않은 메서드를 비롯하여 NativeProfiler 가 인식하는 모든 메서드를 로깅합니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.nativeprofiler.logAllMethodsNoticed=false
```

참고

기본적으로 사용되지 않도록 설정됩니다.

introscope.nativeprofiler.directivematching.cache.max.size

에이전트의 메모리 내 캐시에 저장되는 클래스 이름의 최대 수를 지정합니다. 기본적으로 에이전트는 모니터링할 클래스가 포함된 지시문 그룹을 찾으면 해당 지시문 그룹의 캐시를 메모리 내에 생성합니다. 에이전트는 사용자가 IIS 를 시작할 때마다 이전에 발견된 클래스의 캐시를 생성합니다. 모니터링할 새로운 클래스를 응용 프로그램 코드에서 사용함에 따라 캐시가 증가합니다. 메모리 내 캐시에는 기본적으로 최대 5000 개의 클래스 이름이 저장됩니다.

이 속성의 값을 늘리면 캐시에 클래스 이름을 5000 개 이상 저장해야 할 경우 시작 시간을 개선할 수 있습니다. 하지만 값을 늘리면 에이전트에 필요한 메모리 오버헤드가 증가할 수 있습니다. 속성 값을 줄이면 에이전트의 메모리 오버헤드도 감소합니다. 5000 개보다 적은 클래스를 모니터링할 경우 값을 줄이는 것이 좋습니다.

참고: 이 캐시에는 클래스 개체가 저장되지 않습니다.

기본값

기본적으로 클래스 이름을 최대 5000 개까지 저장할 수 있는 캐시가 설정됩니다.

예

```
introscope.nativeprofiler.directivematching.cache.max.size=5000
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.nativeprofiler.generic.agent.trigger.enabled

이 속성을 사용하면 일반 트리거를 통해 .NET 에이전트를 시작할 수 있습니다. 이 속성을 사용하도록 설정하면 에이전트는 첫 번째 사용자 코드가 실행된 후 일반 트리거를 사용하여 시작됩니다. 이 작업은 제한 가능한 기본 도메인 프로브를 시작합니다. 이 속성을 사용하지 않도록 설정하면 에이전트는 IIS 가 실행 중일 때 기본 메서드 또는 시작 메서드를 사용하여 시작됩니다.

속성 설정

True 또는 False

기본값

False

예

```
introscope.nativeprofiler.generic.agent.trigger.enabled=true
```

참고

- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.
- ProcSxS 에서는 에이전트 이름이 다음과 같이 나타납니다.
  `<ProcessName>_<ApplicationDomainName>`

com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple.clrs

CLR(공용 언어 런타임) 버전이 여러 개인 경우에 에이전트가 모니터링 작업을 처리하는 방법을 제어합니다. 기본적으로 에이전트는 가장 먼저 로드된 CLR 버전에 따라 .NET Framework 버전을 하나만 모니터링합니다. 이 속성을 사용하면 기본 동작을 수정하고 .NET Framework 2.0 구성 요소를 모니터링할지, .NET Framework 4 구성 요소를 모니터링할지 지정할 수 있습니다.

속성 설정

유효한 값은 다음과 같습니다.

- **없음** - 기본 동작을 사용하고 모니터링할 .NET Framework 버전을 먼저 로드된 CLR 에 따라 선택하도록 지정합니다.
- **V2** - .NET Framework 2.0 구성 요소만 모니터링하도록 지정합니다. v2.0.50727 과 같이 CLR 버전 번호를 지정할 수도 있습니다.
- **V4** - .NET Framework 4 구성 요소만 모니터링하도록 지정합니다. v4.0.30319 와 같이 CLR 버전 번호를 지정할 수도 있습니다.
- **기본값** - 기본값은 None 입니다. 이 값은 .NET Framework 버전 하나만 모니터링함을 나타냅니다. 모니터링되는 .NET Framework 버전은 먼저 로드된 응용 프로그램에 따라 결정됩니다.

참고: .NET Framework 2.0 구성 요소와 .NET Framework 4 구성 요소를 동시에 모니터링할 수는 없습니다. 동적 계측을 사용하도록 설정한 경우에는 이 기능이 작동하지 않습니다.

예

```
com.wily.introscope.nativeprofiler.monitor.inprocsxs.multiple
.clrs=v2
```

참고

이 속성의 변경 내용을 적용하려면 응용 프로그램을 다시 시작해야 합니다.

성능 모니터 데이터 수집

성능 모니터 카운터에서 데이터를 수집하는 작업을 제어하도록 속성을 구성할 수 있습니다.

introscope.agent.perfmon.metric.filterPattern

모니터링할 성능 모니터 카운터를 식별하는 정규식을 지정합니다.

속성 설정

일치하는 성능 모니터 카운터 이름을 찾기 위한 텍스트 문자열 정규식입니다. 프로세스별 스레드 메트릭 같은 새로운 카운터를 추가하려면 목록 맨 끝에 새 표현식을 쉼표로 구분하여 추가합니다.
예: *...,|Thread|{osprocessname}*|**

기본값

기본 필터는 프로세서, .NET Data Provider, .NET CLR 및 ASP.NET 성능 모니터 카운터, 개체 및 인스턴스를 모니터링합니다.

예

```
introscope.agent.perfmon.metric.filterPattern=|Processor|*|*,|.NE
T Data Provider*|*|*,|.NET CLR*|{osprocessname}|*,|.NET CLR
Data|*|*,|Process|{osprocessname}|*,|ASP.NET|*
```

참고

|*|* 필터를 사용하면 모든 카운터를 인스턴스가 없는 것으로 열거하도록 성능 모니터에 지시하는 것과 같습니다. 일부 카운터에 대해서는 이 설정을 사용할 수 없습니다.

introscope.agent.perfmon.metric.limit

각 간격에 보고할 수 있는 성능 모니터 메트릭의 최대 수를 지정합니다.

기본값

기본값은 메트릭 1000 개입니다.

예

```
introscope.agent.perfmon.metric.limit=1000
```

introscope.agent.perfmon.metric.pollIntervalInSeconds

Performance Monitor Collection Agent 가 성능 모니터 개체, 카운터 및 인스턴스에서 새 메트릭 값을 확인하는 빈도를 지정합니다. 기본 폴링 간격은 15 초이며 모든 메트릭 값을 확인합니다.

기본값

기본 간격은 15 초입니다.

예

```
introscope.agent.perfmon.metric.pollIntervalInSeconds=15
```

introscope.agent.perfmon.category.browseEnabled

새 성능 모니터 카운터에 대한 검색을 사용하거나 사용하지 않도록 설정합니다.

속성 설정

True 또는 False

기본값

True(사용)

예

```
introscope.agent.perfmon.category.browseEnabled=true
```

introscope.agent.perfmon.category.browseIntervalInSeconds

Performance Monitor Collection Agent 가 새 성능 모니터 개체를 검색하는 빈도를 결정합니다.

속성 설정

간격(초)을 나타내는 양의 정수입니다.

기본값

기본 간격은 600 초(10 분)입니다.

예

```
introscope.agent.perfmon.category.browseIntervalInSeconds=600
```

프로세스 이름

프로세스 이름을 정의하도록 속성을 구성할 수 있습니다.

introscope.agent.customProcessName

Enterprise Manager 및 Workstation 에 나타나는 프로세스 이름을 지정합니다. 이 속성의 주석 처리를 제거하고 Enterprise Manager 및 Workstation 에 나타나는 사용자 지정 이름으로 프로세스 이름을 정의합니다.

기본값

CustomProcessName

예

```
#introscope.agent.customProcessName=CustomProcessName
```

참고

- 이 속성은 기본적으로 주석으로 처리되어 있습니다.

- 이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.defaultProcessName

사용자 지정 프로세스 이름을 정의하지 않고 에이전트가 기본 응용 프로그램 클래스의 이름을 확인할 수 없으면 기본 프로세스 이름이 사용됩니다.

기본값

.NET 프로세스

예

```
introscope.agent.defaultProcessName=.NET Process
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

계측 구성 제한

프로세스 또는 실행 파일 대상 집합에 대한 계측을 구성할 수 있습니다.

참고: 계측 제한에 대한 자세한 내용은 계측 구성을 참조하십시오.

introscope.agent.dotnet.monitorApplications

계측할 프로세스 및 응용 프로그램을 지정합니다.

옵션

정규화된 경로는 지원되지 않습니다. 응용 프로그램 실행 파일의 프로세스 이미지 이름만 사용하십시오. 이름은 대소문자를 구분하며 Windows 프로세스 관리에 보고되는 이름과 정확히 일치해야 합니다.

기본값

```
w3wp.exe,aspnet_wp.exe
```

예

```
introscope.agent.dotnet.monitorApplications=w3wp.exe,aspnet_w
p.exe,dllhost.exe
```

introscope.agent.dotnet.monitorAppPools

계측되는 응용 프로그램 풀을 지정합니다.

옵션

이 속성의 주석 처리를 제거하고 계측할 IIS 응용 프로그램 풀을 지정합니다.

기본값

"NULL","DefaultAppPool","AppPool1","AppPool2"

예

```
#introscope.agent.dotnet.monitorAppPools=
"NULL","DefaultAppPool","AppPool1","AppPool2"
```

참고

- 응용 프로그램 풀 이름은 따옴표로 묶어야 합니다. 어떠한 응용 프로그램 풀에서도 실행되지 않도록 응용 프로그램을 지정하려면 "NULL"을 사용하십시오.
- 주석 처리된 상태로 두어 모든 응용 프로그램 풀을 계측하거나, 주석 처리를 제거하고 계측하려는 응용 프로그램 풀만 나열하십시오.

소켓 메트릭

다음 속성은 소켓 메트릭의 생성을 제어합니다.

introscope.agent.sockets.reportRateMetrics

개별 소켓의 입력/출력 대역폭 속도 메트릭을 보고하도록 설정합니다.

속성 설정

True 또는 False

기본값

True

예

```
introscope.agent.sockets.reportRateMetrics=true
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

SQL 에이전트

다음은 SQL 에이전트에 사용되는 속성입니다.

introscope.agent.sqlagent.sql.maxlength

Investigator 트리에 SQL 에이전트 메트릭에 대해 표시되는 SQL 문의 바이트 단위 크기를 제한합니다.

기본값

기본 제한은 990 바이트입니다.

예

```
introscope.agent.sqlagent.sql.maxlength=990
```

참고

이 속성은 기본적으로 *IntroscopeAgent.profile* 에 표시되지 않습니다. 속성을 에이전트 프로필에 추가하여 값을 변경할 수 있습니다.

introscope.agent.sqlagent.normalizer.extension

이 속성은 미리 구성된 정규화 체계를 재정의하는 데 사용되는 SQL 노멀라이저 확장의 이름을 지정합니다.

사용자 지정 정규화 확장 작업을 수행하려면 해당 매니페스트 특성 *com-wily-Extension-Plugin-{pluginName}-Name* 의 값이 이 속성에 지정된 값과 일치해야 합니다.

이름을 쉼표로 구분된 목록으로 지정할 경우 에이전트는 기본 노멀라이저 확장을 사용합니다.

예를 들어 다음과 같은 설정을 사용하면 정규화에 RegexSqlNormalizer 가 사용됩니다.

```
introscope.agent.sqlagent.normalizer.extension=ext1, ext2
```

이 속성은 SQL 에이전트 메트릭에 대해 Investigator 트리에 나타나는 SQL 문의 크기(바이트)를 제한합니다.

속성 설정

미리 구성된 정규화 체계를 재정의하는 데 사용되는 SQL 노멀라이저 확장의 이름입니다.

기본값

RegexSqlNormalizer

예

```
introscope.agent.sqlagent.normalizer.extension=RegexSqlNormalizer
```

참고

기본 설정을 사용하는 경우에는 다음과 같이 정규식 SQL 문의 노멀라이저 속성도 구성해야 합니다.

- introscope.agent.sqlagent.normalizer.regex.matchFallThrough (see page 252)
- introscope.agent.sqlagent.normalizer.regex.keys (see page 248)
- introscope.agent.sqlagent.normalizer.regex.key1.pattern (see page 249)
- introscope.agent.sqlagent.normalizer.regex.key1.replaceAll (see page 250)
- introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat (see page 251)
- introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive (see page 252)

이 속성의 변경 내용은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.normalizer.regex.keys

정규식 SQL 문 노멀라이저를 설정하려면 이 속성을 introscope.agent.sqlagent.normalizer.extension (see page 246)과 함께 사용하십시오. 이 속성은 regex 그룹 키를 지정합니다. regex 그룹 키는 순서대로 평가됩니다.

기본값

key1

예

```
introscope.agent.sqlagent.normalizer.regex.keys=key1
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.normalizer.regex.key1.pattern

정규식 SQL 문 노멀라이저를 설정하려면 이 속성을 introscope.agent.sqlagent.normalizer.extension (see page 246)과 함께 사용하십시오. 이 속성은 SQL 과 일치시키는 데 사용되는 regex 패턴을 지정합니다.

속성 설정

java.util.Regex 패키지에서 허용되는 유효한 모든 regex 항목을 여기에 사용할 수 있습니다.

기본값

.*call(.*\)\.FOO(.*\)

예

```
introscope.agent.sqlagent.normalizer.regex.key1.pattern=.*call(.*\)\.FOO(.*\)
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.normalizer.regex.key1.replaceAll

정규식 SQL 문 노멀라이저를 설정하려면 이 속성을 introscope.agent.sqlagent.normalizer.extension (see page 246)과 함께 사용하십시오. 이 속성이 false 로 설정되어 있으면 SQL 쿼리에서 첫 번째로 나타나는 일치하는 패턴이 대체 문자열로 바뀝니다. 이 속성이 true 로 설정되어 있으면 SQL 쿼리에서 나타나는 일치하는 패턴이 모두 대체 문자열로 바뀝니다.

속성 설정

True 또는 False

기본값

false

예

```
introscope.agent.sqlagent.normalizer.regex.key1.replaceAll=false
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat

정규식 SQL 문 노멀라이저를 설정하려면 이 속성을 introscope.agent.sqlagent.normalizer.extension (see page 246)과 함께 사용하십시오. 이 속성은 대체 문자열 형식을 지정합니다.

속성 설정

java.util.Regex 패키지와 *java.util.regex.Matcher* 클래스에서 허용되는 유효한 모든 regex 항목을 여기에 사용할 수 있습니다.

기본값

$1

예

```
introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat=$1
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive

정규식 SQL 문 노멀라이저를 설정하려면 이 속성을 introscope.agent.sqlagent.normalizer.extension (see page 246)과 함께 사용하십시오. 이 속성은 패턴 일치 시 대/소문자 구분 여부를 지정합니다.

속성 설정

true 또는 false

기본값

false

예

```
introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive=false
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.normalizer.regex.matchFallThrough

정규식 SQL 문 노멀라이저를 설정하려면 이 속성을 introscope.agent.sqlagent.normalizer.extension (see page 246)과 함께 사용하십시오. 이 속성이 true 로 설정되어 있으면 SQL 문자열이 모든 regex 키 그룹과 비교하여 평가됩니다.

구현은 체인으로 연결됩니다. 예를 들어 SQL 이 여러 키 그룹과 일치하는 경우 group1 의 정규화된 SQL 출력이 group2 에 대한 입력으로 제공되는 방식으로 처리됩니다.

이 속성이 false 로 설정되어 있으면 키 그룹이 일치하는 즉시 해당 그룹의 정규화된 SQL 출력이 반환됩니다.

속성 설정

True 또는 False

기본값

false

예

```
introscope.agent.sqlagent.normalizer.regex.matchFallThrough=false
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.sqlagent.sql.artonly

introscope.agent.sqlagent.sql.artonly 속성은 "평균 응답 시간" 메트릭만 생성하고 보내도록 에이전트를 구성하는 데 사용할 수 있습니다. 백엔드 아래의 모든 SQL 에이전트 메트릭에 영향을 미칩니다. 이 속성 값이 true 이면 SQL 메트릭 및 트랜잭션 추적에 대한 에이전트 성능이 향상될 수 있습니다.

참고: introscope.agent.sqlagent.sql.turnoffmetrics (see page 255)를 true 로 설정하면 이 속성이 재정의됩니다.

중요! 이 속성 설정이 작동하려면 다음 추적 프로그램 매개 변수를 설정해야 합니다.

```
SetTracerParameter: StatementToConnectionMappingTracer
agentcomponent "SQL Agent"
```

이 속성은 기본적으로 다음과 같이 해제되어 있습니다.

```
introscope.agent.sqlagent.sql.artonly=false
```

이 속성의 변경 사항은 즉시 적용되며 관리 인터페이스를 사용하여 변경할 수 있습니다.

참고: 이 속성은 연결 수와 같은 사용자 지정 메트릭은 제어하지 않습니다.

introscope.agent.sqlagent.sql.rawsql

introscope.agent.sqlagent.sql.rawsql 속성은 "트랜잭션 추적"에서 *정규화되지 않은* SQL 을 SQL 구성 요소에 대한 매개 변수로 추가하도록 에이전트를 구성합니다. 이 속성 값이 true 이면 SQL 메트릭 및 트랜잭션 추적에 대한 에이전트 성능이 향상될 수 있습니다.

이 속성은 기본적으로 다음과 같이 해제되어 있습니다.

introscope.agent.sqlagent.sql.rawsql=false

이 속성의 변경 사항은 관리되는 응용 프로그램을 다시 시작한 후에 적용됩니다.

중요! 이 속성을 사용하도록 설정하면 암호 및 중요한 정보가 "트랜잭션 추적"에 표시될 수 있습니다.

introscope.agent.sqlagent.sql.turnoffmetrics

introscope.agent.sqlagent.sql.turnoffmetrics 속성을 사용하여 SQL 문 메트릭을 해제하면 에이전트에서 Enterprise Manager 로 전송되는 메트릭의 수를 줄일 수 있습니다. 이 속성 값이 true 이면 SQL 메트릭 및 트랜잭션 추적에 대한 에이전트 성능이 향상될 수 있습니다.

중요! 이 속성 설정이 작동하려면 다음 추적 프로그램 매개 변수를 설정해야 합니다.

```
SetTracerParameter: StatementToConnectionMappingTracer
agentcomponent "SQL Agent"
```

이 속성은 기본적으로 다음과 같이 해제되어 있습니다.

```
introscope.agent.sqlagent.sql.turnoffmetrics=false
```

이 속성은 introscope.agent.sqlagent.sql.artonly 속성을 재정의합니다.

이 속성의 변경 사항은 즉시 적용되며 관리 사용자 인터페이스를 사용하여 변경할 수 있습니다.

introscope.agent.sqlagent.sql.turnofftrace

introscope.agent.sqlagent.sql.turnofftrace 속성은 에이전트가 백엔드 아래의 SQL 문에 대해 트랜잭션 추적 구성 요소를 생성하여 Enterprise Manager 로 전송할지 여부를 제어합니다. 이 속성 값이 true 이면 SQL 메트릭 및 트랜잭션 추적에 대한 에이전트 성능이 향상될 수 있습니다.

중요! 이 속성 설정이 작동하려면 다음 추적 프로그램 매개 변수를 설정해야 합니다.

```
SetTracerParameter: StatementToConnectionMappingTracer
agentcomponent "SQL Agent"
```

이 속성은 기본적으로 다음과 같이 해제되어 있습니다.

```
introscope.agent.sqlagent.sql.turnofftrace=false
```

이 속성의 변경 사항은 즉시 적용되며 관리 사용자 인터페이스를 사용하여 변경할 수 있습니다.

중단 메트릭

다음 단원에서는 중단 메트릭과 관련된 속성을 정의합니다.

introscope.agent.stalls.thresholdseconds

이 속성은 실행 중인 프로세스가 중단된 프로세스로 간주되기 전까지 허용되는 시간(초)을 지정합니다. "중단 수" 메트릭의 정확성을 보장하려면 중단 임계값을 15 초 이상으로 설정해야 합니다. 이 설정을 통해 Enterprise Manager 가 하베스트 주기를 완료하기까지 허용되는 시간을 지정할 수 있습니다.

기본값

기본값은 30 초입니다.

예

```
introscope.agent.stalls.thresholdseconds=30
```

참고

이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.stalls.resolutionseconds

이 속성은 에이전트가 중단 발생 여부를 검사하는 빈도를 지정합니다. "중단 수" 메트릭의 정확성을 보장하려면 중단 레졸루션을 10 초 이상으로 설정해야 합니다 이 설정을 통해 Enterprise Manager 가 하베스트 주기를 완료하기까지 허용되는 시간을 지정할 수 있습니다.

기본값

기본값은 매 10 초입니다.

예

```
introscope.agent.stalls.resolutionseconds=10
```

참고

이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

트랜잭션 추적

다음은 트랜잭션 추적과 관련된 속성입니다.

introscope.agent.crossprocess.compression

크로스 프로세스 트랜잭션 추적 데이터의 크기를 줄이려면 이 속성을 사용하십시오.

속성 설정

lzma, gzip, none

기본값

lzma

예

```
introscope.agent.crossprocess.compression=lzma
```

참고

- 이 옵션을 사용하면 에이전트 CPU 오버헤드가 늘어나지만 프로세스 간 헤더의 크기는 줄어듭니다.
- *lzma* 압축이 *gzip* 보다 효율적이지만 CPU 를 더 많이 사용할 수 있습니다.
- .NET 에이전트는 *gzip* 옵션을 지원하지 않으므로 상호 운용성이 필요한 경우에는 *gzip* 을 사용하면 안 됩니다.
- 이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.crossprocess.correlationid.maxlimit

크로스 프로세스 매개 변수 데이터의 최대 허용 크기입니다.

크로스 프로세스 매개 변수 데이터의 총 크기가 이 제한보다 큰 경우에는 압축을 적용한 후에도 일부 데이터가 드롭되고 일부 크로스 프로세스 상관 관계 기능이 제대로 작동하지 않습니다.

그러나 이 설정을 사용하면 너무 큰 헤더 크기로 인해 사용자 트랜잭션이 네트워크 전송 시 실패하지 않도록 보호됩니다.

기본값

4096

예

```
introscope.agent.crossprocess.correlationid.maxlimit=4096
```

참고

- 위의 *introscope.agent.crossprocess.compression* 및 *introscope.agent.crossprocess.compression.minlimit* 속성과 함께 사용됩니다.
- 이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.crossprocess.compression.minlimit

압축을 적용할 크로스 프로세스 매개 변수 데이터 길이에 대한 최소 길이를 설정하려면 이 속성을 사용하십시오.

속성 설정

0 에서 총 최대 제한의 두 배까지 introscope.agent.crossprocess.correlationid.maxlimit (see page 259)에 설정할 수 있습니다.

기본값 1500 보다 작게 설정되면 압축이 보다 빈번히 실행되므로 더 많은 CPU 오버헤드를 발생시킵니다. 기본 설정 값인 1500 은 일반적으로 정상적인 조건에서 CPU 에 오버헤드를 발생시키지 않습니다.

기본값

1500

예

```
introscope.agent.crossprocess.compression.minlimit=1500
```

참고

- 위의 introscope.agent.crossprocess.compression 속성에서만 사용됩니다.
- 이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.

introscope.agent.transactiontrace.componentCountClamp

트랜잭션 추적에서 허용되는 구성 요소 수를 제한합니다.

기본값

5000

중요! 클램프 크기가 늘어나면 메모리 요구 사항도 높아집니다. 극단적인 경우 JVM 의 최대 힙 크기를 조정해야 할 수 있습니다. 그러지 않으면 관리되는 응용 프로그램에 메모리가 부족할 수 있습니다.

예

```
introscope.agent.transactiontrace.componentCountClamp=5000
```

참고

- 클램프를 초과하는 트랜잭션 추적은 에이전트에서 삭제되고 에이전트 로그 파일에 경고 메시지가 기록됩니다.
- 이 속성은 동적 속성입니다. 런타임 동안 이 속성의 구성을 변경할 수 있으며, 변경 사항은 자동으로 선택됩니다.
- 설정된 제한에 도달하면 로그에 경고가 나타나고 추적이 중지됩니다.
- 0 은 올바른 값이 **아닙니다**. introscope.agent.transactiontrace.componentCountClamp=0 을 설정하지 **마십시오**.

introscope.agent.transactiontrace.headFilterClamp

헤드 필터링에 허용되는 최대 구성 요소 수준을 지정합니다. 헤드 필터링은 전체 트랜잭션을 수집할 목적으로 트랜잭션의 시작을 검사하는 프로세스입니다. 헤드 필터링 기능은 첫 번째 Blame 관련 구성 요소가 나타날 때까지 각 구성 요소를 검사합니다. 호출 스택 수준이 매우 깊은 트랜잭션의 경우 클램프가 적용되지 않으면 문제가 될 수 있습니다. 클램프 값은 에이전트가 고정된 수준까지만 조회하도록 하여 이 동작이 메모리 및 CPU 사용률에 미치는 영향을 제한합니다.

기본값

30

경고! 클램프 크기가 늘어나면 메모리 요구 사항도 높아집니다. 이 경우 가비지 수집 동작이 영향을 받으므로 응용 프로그램 전체 성능에 영향을 줍니다.

예

```
introscope.agent.transactiontrace.headFilterClamp=30
```

참고

- 이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.
- 샘플링이나 사용자가 시작한 트랜잭션 추적과 같은 일부 다른 메커니즘이 수집할 트랜잭션을 선택하도록 활성화되어 있지 *않은 경우*에는 가능한 수집에 대해 클램프를 초과하는 수준의 트랜잭션 추적이 더 이상 검사되지 않습니다.

introscope.agent.transactiontracer.parameter.httprequest.headers

캡처할 HTTP 요청 헤더 데이터를 쉼표로 구분된 목록으로 지정합니다. 쉼표로 구분된 목록을 사용하십시오.

기본값

주석 처리됨. *User-Agent*

예

```
introscope.agent.transactiontracer.parameter.httprequest.headers=
User-Agent
```

참고

IntroscopeAgent.profile 에는 이 속성의 값을 Null 값으로 설정하는 주석 처리된 문이 포함되어 있습니다. 필요한 경우 문의 주석 처리를 제거하고 원하는 헤더 이름을 지정할 수 있습니다.

introscope.agent.transactiontracer.parameter.httprequest.parameters

캡처할 HTTP 요청 매개 변수 데이터를 쉼표로 구분된 목록으로 지정합니다.

기본값

주석 처리됨. 일반 매개 변수

예

```
introscope.agent.transactiontracer.parameter.httprequest.parameters=parameter1,parameter2
```

참고

IntroscopeAgent.profile 에는 이 속성의 값을 Null 값으로 설정하는 주석 처리된 문이 포함되어 있습니다. 필요한 경우 문의 주석 처리를 제거하고 원하는 매개 변수 이름을 지정할 수 있습니다.

introscope.agent.transactiontracer.parameter.httpsession.attributes

캡처할 HTTP 세션 특성 데이터를 쉼표로 구분된 목록으로 지정합니다.

기본값

주석 처리됨. 일반 매개 변수

예

```
introscope.agent.transactiontracer.parameter.httpsession.attributes=attribute1,attribute2
```

참고

IntroscopeAgent.profile 에는 이 속성의 값을 Null 값으로 설정하는 주석 처리된 문이 포함되어 있습니다. 필요한 경우 문의 주석 처리를 제거하고 원하는 매개 변수 이름을 지정할 수 있습니다.

introscope.agent.transactiontracer.sampling.enabled

트랜잭션 추적 프로그램 샘플링을 사용하지 않으려면 다음 속성의 주석 처리를 제거하십시오.

속성 설정

True 또는 False

기본값

False

예

```
introscope.agent.transactiontracer.sampling.enabled=false
```

참고

이 속성의 변경 사항은 즉시 적용되며 관리되는 응용 프로그램을 다시 시작할 필요가 없습니다.

introscope.agent.transactiontracer.sampling.perinterval.count

이 속성은 일반적으로 Enterprise Manager 에서 구성됩니다. 이 속성을 에이전트에서 구성할 경우 Enterprise Manager 의 구성은 사용되지 않습니다. 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

기본값

1

예

```
introscope.agent.transactiontracer.sampling.perinterval.count=1
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

introscope.agent.transactiontracer.sampling.interval.seconds

이 속성은 일반적으로 Enterprise Manager 에서 구성됩니다. 이 속성을 에이전트에서 구성할 경우 Enterprise Manager 의 구성은 사용되지 않습니다.

참고: 자세한 내용은 *CA APM 구성 및 관리 안내서*를 참조하십시오.

기본값

120

예

```
introscope.agent.transactiontracer.sampling.interval.seconds=120
```

참고

이 속성의 변경 내용을 적용하려면 관리되는 응용 프로그램을 다시 시작해야 합니다.

세션 ID 수집 구성

introscope.agent.transactiontracer.parameter.capture.sessionid 속성은 트랜잭션 추적 프로그램 데이터에서 세션 ID 수집을 사용하거나 사용하지 않도록 설정합니다. 기본적으로 이 속성은 트랜잭션 추적 프로그램 데이터에서 사용하도록 설정되고 기록됩니다. 이 속성을 사용하지 않을 경우 필터에 데이터를 사용할 수 없습니다.

다음 단계를 따르십시오.

1. IntroscopeAgent.profile 파일을 텍스트 편집기에서 엽니다.

다음 행을 찾습니다.

```
# Uncomment the following property to disable sessionid capture
in TransactionTracer data.
# By default, it is enabled and recorded in the TT Data.

#
introscope.agent.transactiontracer.parameter.capture.sessioni
d=true
```

2. 지시에 따라 행을 주석으로 처리하거나 주석 처리를 제거하여 속성을 사용하거나 사용하지 않도록 설정합니다.

```
#
introscope.agent.transactiontracer.parameter.capture.sessioni
d=true
```

3. 파일을 저장하고 닫은 다음 에이전트를 다시 시작합니다.

에이전트 구성이 세션 ID 수집에 대해 지정된 값을 사용하도록 설정됩니다.

introscope.agent.transactiontracer.userid.key

사용자 정의된 키 문자열입니다.

기본값

주석 처리됨. 일반 매개 변수

예

```
#introscope.agent.transactiontracer.parameter.httpsession.attributes=attribute1,attribute2
```

참고

IntroscopeAgent.profile 에는 이 속성의 값을 Null 값으로 설정하는 주석 처리된 문이 포함되어 있습니다. *HttpServletRequest.getHeader* 또는 *HttpServletRequest.getValue* 를 사용하여 사용자 ID 를 액세스하는 경우 사용자는 필요하면 이 명령문의 주석 처리를 제거하고 올바른 값을 제공할 수 있습니다.

자세한 내용은 introscope.agent.transactiontracer.userid.method (see page 271)를 참조하십시오.

introscope.agent.transactiontracer.userid.method

사용자 ID 를 반환하는 메서드를 지정합니다. 에이전트 프로필에는 세 개의 허용 가능한 각 값에 대한 주석 처리된 속성 정의가 포함되어 있습니다.

getRemoteUser, getHeader 또는 getValue 중 어떤 메서드로 사용자 ID 에 액세스할지에 따라 적절한 문의 주석 처리를 제거하십시오.

속성 설정

허용 가능한 값은 다음과 같습니다.

- *HttpServletRequest.getRemoteUser*
- *HttpServletRequest.getHeader*
- *HttpServletRequest.getValue*

기본값

주석 처리됨. 위의 옵션 참조

예

IntroscopeAgent.profile 에는 세 개의 허용 가능한 각 값에 대한 주석 처리된 속성 정의가 포함되어 있습니다. 사용할 속성의 주석 처리를 제거할 수 있습니다.

```
introscope.agent.transactiontracer.userid.method=HttpServletReque
st.getRemoteUser
#introscope.agent.transactiontracer.userid.method=HttpServletRequ
est.getHeader
#introscope.agent.transactiontracer.userid.method=HttpSession.get
Value
```

URL 그룹화

다음은 프런트엔드 메트릭을 위한 URL 그룹을 구성하는 데 사용되는 속성입니다.

introscope.agent.urlgroup.keys

프런트엔드 이름 지정에 대한 구성 설정입니다.

기본값

기본값

예

```
introscope.agent.urlgroup.keys=default
```

참고

URL 주소가 두 개의 URL 그룹에 속하는 경우 이 속성에서 URL 그룹에 대한 키를 나열하는 순서는 중요하지 않습니다. 더 좁은 패턴에 의해 정의된 URL 그룹은 더 넓은 패턴에 의해 정의된 URL 그룹에 선행해야 합니다.

예를 들어, 키가 alpha 인 URL 그룹에 하나의 주소가 포함되어 있고 키가 beta 인 URL 그룹에 첫 번째 URL 그룹의 주소를 포함하는 네트워크 세그먼트에 있는 모든 주소를 포함하는 경우, 키 매개 변수에서 alpha 가 beta 에 선행해야 합니다.

introscope.agent.urlgroup.group.default.pathprefix

프런트엔드 이름 지정에 대한 구성 설정입니다.

기본값

*

예

```
introscope.agent.urlgroup.group.default.pathprefix=*
```

introscope.agent.urlgroup.group.default.format

프런트엔드 이름 지정에 대한 구성 설정입니다.

기본값

기본값

예

```
introscope.agent.urlgroup.group.default.format=default
```

부록 B: 응용 프로그램별 구성

.NET Framework 에서는 *.config* 확장명을 사용하는 선택적 XML 형식 파일을 사용하여 응용 프로그램별 매개 변수를 구성할 수 있습니다.

구성에 필요한 속성 추가

ASP.NET 응용 프로그램의 경우 응용 프로그램별 설정 및 구성을 위한 기본 파일은 *web.config* 입니다. 이 파일은 응용 프로그램 루트 디렉터리에 저장되어 있습니다.

다른 .NET 실행 파일의 경우 구성 파일은 응용 프로그램에 *.config* 확장명을 추가하여 이름이 지정됩니다. 해당 파일은 응용 프로그램 실행 파일과 동일한 디렉터리에 저장됩니다. 예를 들어 *testapp.exe* 의 선택적 구성 파일 이름은 *testapp.exe.config* 가 됩니다.

.config 파일에는 Introscope 와 관련된 구성을 추가할 수 있습니다. 예를 들어 개별 응용 프로그램이 *IntroscopeAgent.profile* 파일의 자체 인스턴스를 참조할 수 있도록 매개 변수를 설정하여 응용 프로그램마다 서로 다른 에이전트 구성을 사용할 수 있게 할 수 있을 뿐만 아니라 웹 서비스에 프로세스 간 트랜잭션 상관 관계를 사용하도록 매개 변수를 설정할 수 있습니다.

다음 단계를 따르십시오.

1. 응용 프로그램 구성 파일을 엽니다.
2. 응용 프로그램 구성 파일에 *sectionGroup* 과 *section* 을 추가합니다. 각각의 이름을 다음과 같이 지정합니다.

```
<configuration>
   <configSections>
      <sectionGroup>
         <sectionGroup name="com.wily.introscope.agent">
         <section
name="env.parameters"type="System.Configuration.NameValueSectionHandler" />
      </sectionGroup>
```

3. *env.parameters* 섹션에 새 속성을 추가합니다. 예:

```
<com.wily.introscope.agent>
   <env.parameters>
      <add key="com.wily.introscope.agentProfile"
value="e:\\junkyard\\dotnettest\\Agent.profile" />
   </env.parameters>
</com.wily.introscope.agent>
```

예를 보려면 *sample.exe.config* 를 참조하십시오.

참고: .NET 에이전트 설치 시에는 구성 파일이 필요하지 않습니다.

부록 C: 네트워크 인터페이스 유틸리티 사용

네트워크 인터페이스 유틸리티를 사용하여 Catalyst 통합을 위해 에이전트에서 사용되는 호스트 컴퓨터의 네트워크 인터페이스 이름 값을 확인할 수 있습니다.

이 섹션은 다음 항목을 포함하고 있습니다.

네트워크 인터페이스 이름 확인

네트워크 인터페이스 유틸리티는 introscope.agent.primary.net.interface.name 속성에 대한 이름 및 하위 인터페이스 값을 제공합니다.

다음 단계를 따르십시오.

1. 다음 디렉터리로 이동합니다.

   <Agent_Home>\wily\tools

2. NetInterface.exe 를 실행합니다.

   .NET 에 지원되는 네트워크 인터페이스 이름이 브라우저의 "네트워크 인터페이스" 탭에 표시됩니다.

추가 정보:

사용 가능한 네트워크 목록 구성 (페이지 194)