WTC Connection issue (신항식)

1. WTC 개요

WTC(WebLogic Tuxedo Connector)는 Tuxedo의 Domain 간 통신 수단인 /T Domain 구현한 WebLogic 모듈이다. 이 WTC 모듈을 이용하여 기간 시스템인 Tuxedo 접근을 유용하게 함으로써 Web 기반 application에 대한 확장성을 극대화할 수 있게 되었다.

이 문서에서는 WTC에 대한 기본적인 개발방법과 설정 방법보다는 WTC를 통한 Tuxedo 연결방식, FAIL-OVER 등의 좀 더 깊은 주제를 다루려 한다.

1) JOLT 와 WTC의 비교

WTC를 이용하여 기존의 JOLT를 이용한 Tuxedo와의 단방향 통신을 양방향으로 통신을 할 수 있게 하였다. 아래는 WTC와 JOLT를 간단하게 비교한 것이다.

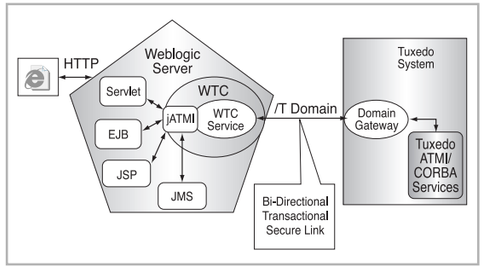
구 분	JOLT	WTC
호출방향	단방향 (JAVA → Tuxedo)	양방향 (WLS ↔ TUXEDO)
제 품	Tuxedo 계열 Product	WebLogic Component
특 징	JAVA program 가능	반드시 WebLogic 이여야 함.

특히 위의 비교내용을 보면 JOLT의 경우는 Tuxedo 계열 Product로써 추가로 구매해야 하는 점이 있는가 하면, WTC의 경우 WebLogic Component로써 추가 구매가 필요 없이 기본적으로 내장이 되어 있다는 점에서 이점이 있다.

그러나 WTC를 이용하려면 WAS로써 반드시 WebLogic을 사용해야만 Tuxedo에 접근이 가능하며, 다른 WAS나 일반 JAVA 어플리케이션에서 Tuxedo 연동을 하려 한다면, JOLT를 사용하여 통신을 해야 한다.

2) WTC 구조

아래 그림은 WTC구조를 개략적으로 나타낸 그림이다.



위의 그림에서 WTC는 Tuxedo 와 /T Domain을 통해서 통신을 하는 것을 알 수 있다.
이러한 통신을 통해서 WebLogic에 구현된 EJB, Servlet, JSP 등의 JAVA 모듈은 Connection을 얻어오거나, 데이터에 대한 버퍼설정, 그리고 Tuxedo service를 호출하는 일들을 마치 Tuxedo 에서 사용하는 ATMI 함수를 사용하여 처리하는 것처럼 JATMI 함수를 통해서 Tuxedo 와 통신을 하게 된다.
또한 이러한 WTC 통신은 양방향 통신이 가능하기 때문에 WebLogic에서의 Tuxedo service 호출 뿐만 아니라 Tuxedo 쪽에서 WebLogic에 존재하는 EJB를 통해서 WAS 쪽 Resource를 사용할 수 있다.

3) WTC Connection

WTC 통신에는 몇 가지 Connection에 대한 옵션이 존재를 한다. 이러한 Connection에 대한 Policy를 통해서 양쪽 Domain간의 접속시점 뿐만 아니라 Fail-Over, Fail-Back, Load Balancing 등이 존재하게 된다. 이번 기고에서는 이러한 문제에 대해서 좀더 깊게 알아보도록 한다.

2. WTC Connection Policy

Connection Policy 라는 말은 WTC를 통해서 Tuxedo와 통신을 하기 위한 최초 연결을 어떻게 할 것인가에 대한 설정이다. 이러한 설정에는 아래와 같이 4가지 종류가 존재한다.

Connection Policy	설명
On Startup	부팅시 연결하며, Fail 시 지속적으로 재시도
On Demand	Service 요청이 있을 때 연결
Incoming Only	상대 Domain의 요청에 의해서만 연결
LOCAL	Local Access Point 쪽 설정을 따름

위의 설정은 WTC에서 Local Tuxedo Access Point 와 Remote Tuxedo Access Point 의 두 부분에서 설정이 가능하며, Local Tuxedo Access Point 보다는 Remote Tuxedo Access Point가 우선순위를 갖기 때문에 Remote Tuxedo Access Point가 Override 한다고 할 수 있다. 이 두 곳에서 설정하는 값은 모두 WebLogic 쪽에서 Tuxedo 쪽으로 Connection 시도할 때 설정되는 값이라는 점에 유의해야 한다.

위의 Connection에 대한 특징과 Connection policy에 대해서 각각 자세히 설명하도록 한다.

1) Domain Gateway Connection 특징

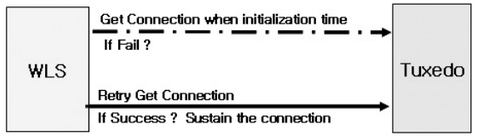
Connection이 이루어지는 방향성은 양쪽 Domain에서 모두 가능하다. 즉, WTC 설정을 하게 되면 WebLogic Domain과 Tuxedo Domain 양쪽에서 TCP/IP Listen을 하게 되며 먼저 연결되는 Connection에 대해서 양방향 통신을 모두 사용하게 된다. 그러므로 양쪽 Domain에 대해서 연결된 TCP/IP Connection은 단 하나 존재하게 된다.

그리고 한번 연결된 Connection에 대해서는 모든 요청에 대해서 Connection 시도없이 해당 Connection을 지속적으로 재사용을 하며, 만일 해당 Connection이 끊어진다면 Connection Policy의 설정값에 따라 Connection이 맺어지게 된다.

2) On Startup

아래 그림은 Connection Policy가 On Startup에 대해서 간단하게 설명하는 그림이다.

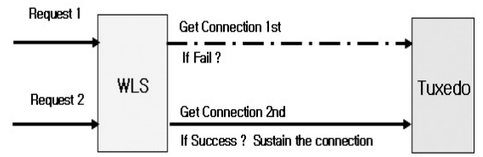
그림을 보면, WLS의 WTC가 기동 시 Tuxedo와 연결을 시도하게 된다. 그러나 최초 연결이 실패를 하더라도 WebLogic Console 상에서 Retry Interval과 Max Retries 값을 통해서 지속적으로 연결 시도를 할 수 있다.



이러한 Retry 속성에 대한 디폴트 값은 60초 간격으로 $2^{63}-1$ 회수만큼 retry을 시도하게 된다.
On startup의 또 다른 특징은 Connection이 이루어지는 순간 Service가 Advertise(Service가 Available 한 상태) 상태가 되기 때문에 Connection이 끊어진 상태에서 요청이 들어왔을 경우에는 끊어진 Connection으로 요청이 가지 않게 된다. 이러한 이유는 Connection이 끊어지게 되면 바로 Un-advertise(Service가 Unavailable 한 상태) 상태로 바뀌게 되기 때문이다. 이 부분에 대한 언급은 추후 Fail-Over에서 다시 언급한다.

3) On Demand

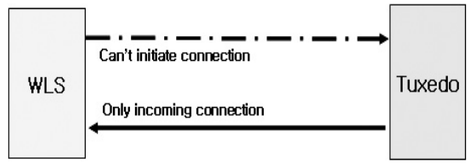
On demand는 최초 Service 요청이 일어나는 시점에 Connection이 이루어진다.
결국 아무런 요청이 없으면 Connection 조차 이루어지지 않게 된다. 그리고 만일 운영 중에 Connection이 network 등의 기타 다른 원인에 의해서 끊어지게 되어도, 계속적인 Service 요청이 발생한다면 계속 Connection을 연결하려는 시도가 이루어 진다. 그렇기 때문에 On Startup과 같은 Retry 에 대한 설정이 필요 없게 된다.



On demand는 Service advertise 되는 시점에 대해서 On Startup과 다른 점이 존재한다. Connection 이 이루어 지는 시점에 advertise 가 되는 On Startup 에 비해, On Demand의 경우는 Connection 이 실패 했더라도 항상 advertise 가 되어 있다. 결국 이 시점에 요청이 들어오게 되면 에러가 발생하게 된다.

4) Incoming Only

Incoming Only는 말 그대로 Connection 요청이 상대방 Domain 에서 들어올 때 연결을 맺는다는 것이다. 결국 WebLogic 입장에서 Incoming Only 설정을 했다면, Connection 요청은 반드시 Tuxedo 쪽에서의 요청에 의해서만 가능하다.

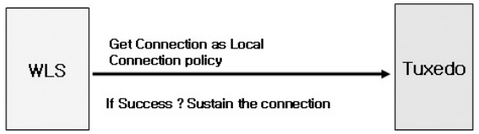


이 경우도 On Startup과 마찬가지로 Connection이 이루어지는 순간 Service가 Advertise(Service가 Available 한 상태)하게 되기 때문에 Connection이 끊어진 상태에서 요청이 들어왔을 경우에는 끊어진 Connection으로 요청이 가지 않게 된다.
이러한 이유는 Connection이 끊어지게 되면 바로 Un-advertise(Service가 Unavailable 한 상태) 상태로 바뀌게 되기 때문이다

5) LOCAL

LOCAL 옵션을 사용하는 경우는 Remote Tuxedo Access Point 에서만 가능하다.

LOCAL 옵션은 Local Tuxedo Access Point 값을 그대로 사용하겠다는 의미이기 때문에, 결국 Remote Tuxedo Access Point를 사용하지 않겠다는 의미와도 동일하다. 그래서 LOCAL 설정을 하면 전적으로 Connection Policy는 Local Tuxedo Access Point에 설정된 값에 의해서 적용이 된다.

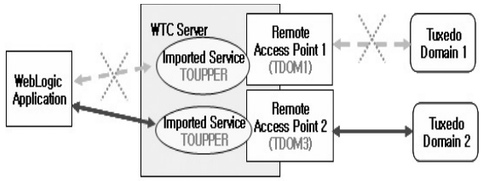


3. WTC Connection Fail-Over, Fail-Back

WTC Connection에 대해서 문제가 발생 하였을 때 Connection에 대한 Fail-Over, Fail-Back에 대해서 알아보고 Service call 에 대한 영향도를 알아보자.
각각의 그림에서 나타난 TOUPPER는 Tuxedo에 존재하는 Service를 의미하며, TDOM1, TDOM3은 각각 Remote Tuxedo Access Point를 의미한다.

1) On Startup

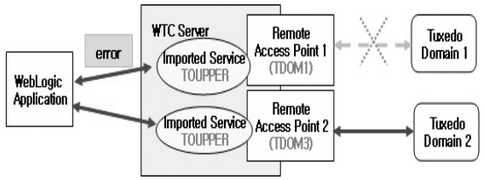
On Startup에 대한 Fail Over는 Service 가 Connection 의 상태에 따라서 advertise, un-advertise 되기 때문에 Available한 Connection으로만 service 요청이 가게 된다.
결국 자동으로 정상적인 Connection에 service 요청을 하므로 service 요청에 대한 에러 발생이 존재하지 않는다. 결국 아래 그림에서 보면 아래 라인으로만 service 요청이 가게 되는 것이다.



Connection에 대한 복구 시도는 Retry 옵션에 의해서 지속적으로 시도를 하게 되며, Connection 이 이루어 지는 시점에 Service가 advertise 되게 된다. 결국 그 시점부터 해당 Connection으로 요청이 이루어 지면서 자동적으로 Fail-Back 된다.

2) On Demand

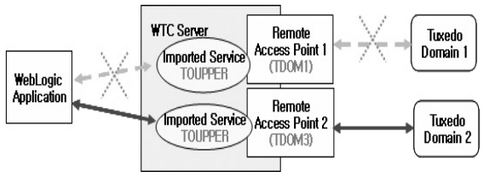
On Demand는 Connection 상태와는 무관하게 Service가 항상 Advertise 상태를 갖는다. 그렇기 때문에 WebLogic application은 해당 Service가 Available한 상태로 인식하고 Service 요청을 하게 되어 에러가 발생하게 된다. 결국 Fail-Over가 정상적으로 이루어지지 않는다는 것을 알 수 있다.



그러나 On Demand는 Connection 이 이루어지지 않은 상태에서는 요청이 들어올 때마다 Connection을 시도하므로 Fail-Back의 경우는 어떤 Connection policy 보다도 빠르게 된다는 것을 알 수 있다.

3) Incoming Only

Incoming Only는 On Startup 과 Fail-Over 형태가 동일하다. Service가 Connection의 상태에 따라서 advertise, un-advertise 되기 때문에 available 한 Connection으로만 service 요청이 가게 된다



마찬가지로 Fail-Back의 경우는 외부 Domain으로부터 요청이 와서 Connection이 이루어지는 시점에 Service가 Advertise 되므로 Connection이 이루어지는 시점에 Fail-Back이 된다고 볼 수 있다.

4. WTC 사용 시 참고사항

아래는 WTC 사용 시 유용한 정보이다. 참고하기 바란다.

1) applicationQueueSize

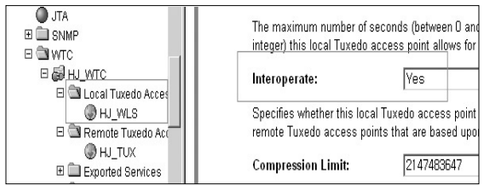
WTC를 이용하여 Tuxedo 영역의 Domain에서 WebLogic 영역의 Domain의 application을 호출할 때는 반드시 EJB를 이용하여 호출해야 한다. 이 경우 Tuxedo application쪽 Service Logic 처리시간에 비해서 WebLogic쪽 application 수행 시간이 상대적으로 길어 Thread 잠식 시간이 오래 된다면, WebLogic Thread Starvation을 발생할 위험이 존재한다. 이 경우 아래와 같은 옵션을 줌으로써 WTC관련 Thread Queue를 따로 설정하여 Default Thread Queue에 대한 부담을 덜어 줄 수 있다.

```
JAVA_OPTIONS=
-Dweblogic.wtc.applicationQueueSize=threads 개수.
```

위의 옵션은 WLS8.1 SP3부터 유효하다.

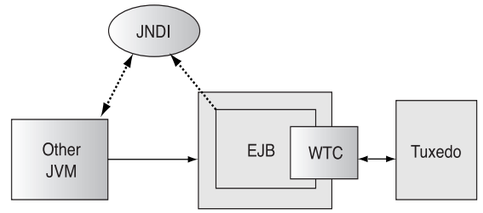
2) Tuxedo 6.5 버전과의 호환성.

Tuxedo는 7.1 버전 이후로 많은 변화가 존재했다. 그래서 Tuxedo 7.1 이전, 이후 버전간의 호환성을 위해서 옵션을 설정한다. 마찬가지로 WTC를 이용한 WebLogic에서도 이러한 옵션을 설정해 줘야 Tuxedo 7.1 이전 버전과의 통신을 보장할 수 있다. 아래는 Console 상의 설정 예이다. 참고하기 바란다.



3) tuxedo.services.TuxedoConnection

WebLogic 어플리케이션에서 Tuxedo와 연동을 위한 WTC connection을 얻기 위한 JNDI 이름으로써 tuxedo.services.TuxedoConnection를 이용한다. 그러나 일반적인 JNDI service의 경우는 다른 JVM에서 접근하여 Remote call 이 가능한 형태로 제공하는 service 이지만, WTC의 경우는 JNDI를 사용하여 얻은 Remote object를 통한 Remote call 사용을 금하고 있다.
결국 아래 그림과 같이 다른 JVM 에서의 Remote Call을 이용하여 WTC Resource를 사용하려 할 때는 EJB 모듈로 한번 Wrapping 해서 접근하는 구조로 가야할 것이다.



4) WTC log trace level 설정

WebLogic에서 option 설정을 통해서 WTC trace logging을 할 수 있다. 설정방법 및 Trace Level은 아래와 같다.

```
JAVA_OPTIONS=-Dweblogic.wtc.TraceLevel=100000
```

참고로 아래 표와 같이 Trace Level은 큰 값을 가질수록 많은 정보를 Logging 하게 된다.

Value	Components	Description
10000	TBRIDGE_IO	Tuxedo Queuing Bridge input and output
15000	TBRIDGE_EX	more Tuxedo Queuing Bridge information
20000	GWT_IO	Gateway input and output, including the ATMI verbs
25000	GWT_EX	more Gateway information
50000	JAMTI_IO	JAMTI input and output, including low-level JAMTI calls
55000	JAMTI_EX	more JAMTI information
60000	CORBA_IO	CORBA input and output
65000	CORBA_EX	more CORBA information
100000	All Components	information on all WebLogic Tuxedo Connector components