|  |  |
| --- | --- |
| **문서번호** | **SDC-Template-XXX** |



**네이버 응용프로그램 업그레이더 인터페이스 정의**

|  |
| --- |
| **기술본부.개인&공통서비스개발센터.어플컴포넌트개발팀** |
| **Issue Date : 2009-11-20** |
| **Version : 2.0** |

**0. 문서정보**

문서작성 일자

* 1. **문서일반 정보**

|  |  |
| --- | --- |
| **관련 서비스명** | 통합인스톨러 |
| **프로젝트명** | 네이버 응용 프로그램 업그레이더 인터페이스 |
| **담당부서/작성자** | 어플컴포넌트개발팀/ 전희영 | **작성일자** | 2009-11-20 |

* 1. **문서변경 이력**

|  |  |  |  |
| --- | --- | --- | --- |
| **버전** | **변경일자** | **변경자** | **변경내용** |
| 0.1 | 2008.07.15 | 전희영 | 최초 작성 |
| 0.2 | 2008.07.28 | 전희영 | 리뷰 후 DLL형식의 인터페이스로 구조 변경 |
|  | 2008.08.13 | 강대명 | 설치, 업데이트, 삭제시 Agent DLL 이용 시퀀스 추가 |
| 0.9 | 2008.08.14 | 전희영 | 인터페이스 문서 최종 정리 |
| 1.0 | 2008.0829 | 전희영 | Agent Dll 설치 주체 변경 |
| 1.1 | 2008.10.07 | 전희영 | 약관 관련 이슈로 통합 인터페이스 추가 |
| 1.2 | 2008.10.24 | 강대명 | 인터페이스 변경에 따른 시퀀스 다이어그램 변경공통 디렉토리 구조 설명 |
| 1.3 | 2008.11.06 | 전희영 | 오류 메시지 정의Request함수의 install.error 스펙 추가 |
| 1.4 | 2008.11.13 | 전희영 | 설치 프로그램, 삭제 프로그램 사일런트 모드 기호 변경 /M |
| 1.5 | 2008.11.17 | 전희영 | Request함수의 uninstall.error 스펙 추가 |
| 1.6 | 2008.11.26 | 전희영 | 5 Status정의 잘못된 부분 수정 |
| 1.7 | 2009.01.22 | 전희영 | 설치/삭제 인터페이스 정의 추가 (/NT) |
| 1.9 | 2009.02.19 | 전희영 | 삭제 인터페이스 정의 추가 |
| 2.0 | 2009.11.20 | 전희영 | 미니캘린더 GUID 추가 |
|  |

목차

[1. 개요 4](#_Toc213832402)

[1.1. 문서의 목적 4](#_Toc213832403)

[1.2. 문서의 요약 4](#_Toc213832404)

[1.3. 용어 4](#_Toc213832405)

[2. 요구 사항 6](#_Toc213832406)

[3. 업그레이더 인터페이스 정의 7](#_Toc213832407)

[3.1. 각 서비스와의 업그레이더 인터페이스 구성도 7](#_Toc213832408)

[3.2. AgentDll 필수 빌드 환경 7](#_Toc213832409)

[3.3. Agent Dll 필수 항목 9](#_Toc213832410)

[3.4. Agent DLL 배포 10](#_Toc213832411)

[3.4.1. 배포 주체 10](#_Toc213832412)

[3.4.2. 배포 방법 10](#_Toc213832413)

[3.4.3. 서비스 별 GUID 10](#_Toc213832414)

[3.5. 통합인스톨러 Agent DLL 관리 동작 시퀀스 11](#_Toc213832415)

[3.5.1. 데이터 모델 11](#_Toc213832416)

[3.5.2. 기본 구조 12](#_Toc213832417)

[3.5.3. 상세 설계 12](#_Toc213832418)

[3.6. AgentDll에서 작성되어야 할 함수 규약 – Appendix의 헤더파일 참고 13](#_Toc213832419)

[3.6.1. Request 13](#_Toc213832420)

[4. 설치/삭제 프로그램 인터페이스 17](#_Toc213832421)

[4.1. 인터페이스 시나리오 17](#_Toc213832422)

[4.2. 인터페이스 규약 18](#_Toc213832423)

[4.3. 설치 프로그램 인터페이스 정의 18](#_Toc213832424)

[4.4. 삭제 프로그램 인터페이스 정의 19](#_Toc213832425)

[4.5. 설치/삭제 프로그램 Exit Code 정의 19](#_Toc213832426)

[5. Status 정의 19](#_Toc213832427)

[5.1.1. 서비스 업데이트 종료 시 결과 19](#_Toc213832428)

[5.1.2. 쿼리 오류 정의 20](#_Toc213832429)

[5.1.3. 서비스 설치 종료 시 결과 21](#_Toc213832430)

[5.1.4. 서비스 삭제 종료 시 결과 21](#_Toc213832431)

[6. 설치/업데이트/삭제 시 Agent DLL 사용 시나리오 22](#_Toc213832432)

[6.1. 설치 시 Agent DLL 사용 시나리오 22](#_Toc213832433)

[6.1.1. 통합인스톨러/통합업데이터 설치 시 Agent DLL 사용 시나리오 22](#_Toc213832434)

[6.2. 업그레이드 시 Agent DLL 사용 시나리오 24](#_Toc213832435)

[6.2.1. 통합인스톨러에 의한 업그레이드 시 Agent DLL 사용 시나리오 24](#_Toc213832436)

[6.2.2. 통합업데이터에 의한 업그레이드 시 Agent DLL 사용 시나리오 25](#_Toc213832437)

[6.3. 삭제 시 Agent DLL 사용 시나리오 시퀀스 26](#_Toc213832438)

[6.3.1. 기본 구조 26](#_Toc213832439)

[7. Agent DLL 배포 규약 26](#_Toc213832440)

[8. 각 서비스에서의 1차 리뷰 사항 27](#_Toc213832441)

[9. Appendix 29](#_Toc213832442)

[9.1. 공통 헤더 파일 29](#_Toc213832443)

[9.2. NSIS 프로그램에서 실행 파라미터 확인 방법 30](#_Toc213832444)

[9.3. Agetn Dll XML 형식 31](#_Toc213832445)

[9.4. Agent Dll샘플 구현 예제 31](#_Toc213832446)

[9.4.1. Type A – 포토매니저 31](#_Toc213832447)

[9.4.2. Type B - PC그린 36](#_Toc213832448)

[9.4.3. Type C – 툴바 39](#_Toc213832449)

[10. FAQ 41](#_Toc213832450)

1. 개요
	1. 문서의 목적
* 본 문서의 독자는 네이버 응용 프로그램 개발자들이다.
* 본 문서는 통합 인스톨러 / 업데이터의 개발을 위해 네이버 응용 프로그램과의 인터페이스 규약을 정하기 위함이며 통합 인스톨러 / 업데이터 출시 전에 네이버 응용 프로그램에서는 문서에 정의된 인터페이스를 숙지하고 구현하여야 한다.
* 본 문서는 통합 인스톨러, 통합 업데이터에서 필요한 네이버 응용 프로그램의 설치 유무, 버전 정보 조회, 업그레이더/언인스톨러 호출 방법 등 각종 정보를 조회하는 방법을 정의한다.
	1. 문서의 요약

[본 문서에서 기술하게 될 요구사항 명칭 및 정의 내용 기술]

|  |  |
| --- | --- |
| **항목** | **내용** |
| 요구 사항 | - 통합인스톨러/업데이터에서 네이버 응용 프로그램 설치 유무, 버전 정보 조사- 통합인스톨러/업데이터에서 네이버 응용 프로그램 업데이트 기능 호출 |
| 담당자 | 전희영, 강대명, 장호진, 네이버 각 응용 프로그램 개발자 |
| 구현 | 2008.08.18 ~ 2008.10.06 |
| 안정화 | 2008.10. 06 ~ 2008.11.06 |

* 1. 용어
* 업그레이더 : 서비스 내 모듈을 최신 버전으로 업데이트해 주는 역할을 담당하는 모듈이다. 업데이터와 같은 의미이다.
* 업데이트 : 서비스 내 모듈을 최신 버전으로 갱신해는 행위를 말한다. 업그레이드와 같은 의미이다.
* 일반 업데이트 : 기존 서비스에서 시행하고 있는 업데이트 방식을 의미한다. 서비스마다 다른 업데이트 후의 로직도 포함한다.

1. 요구 사항
2. 통합 인스톨러 / 업데이터가 설치된 서비스의 업그레이더를 실행 시켜야 함
	1. 통합 인스톨러 / 업데이터가 버전 변경을 확인해야 함
	2. Admin 페이지에서 설정한 값에 따라 각 서비스의 업그레이더를 Silent 또는 Normal 모드로 실행시킬 수 있어야 함
	3. 경우에 따라 업그레이드 후에 서비스의 실행 또는 실행 금지로 구분하여 실행시킬 수 있어야 함
	4. 각 서비스의 업데이트 결과를 파악해야 함
	5. 각 서비스의 업데이트 로직은 통합 인스톨러 / 업데이터가 가지고 있지 않아야 함
3. 통합 인스톨러 / 업데이터가 각 서비스의 설치 프로그램을 다운로드 받고 실행 시켜야 함
	1. Admin 페이지에서 설정한 값에 따라 각 서비스의 설치 프로그램을 Silent 또는 Normal 모드로 실행시킬 수 있어야 함
	2. 각 서비스의 설치 결과 파악해야 함
	3. Admin 페이지에서 설정한 값에 따라 설치 후에 해당 서비스를 실행 또는 실행 금지 모드로 설치 프로그램을 실행 시킬 수 있어야 함
4. 통합 업데이터가 각 서비스의 삭제 프로그램의 런칭 역할을 해야 함
	1. 각 서비스의 업데이트 진행을 Silent 모드로 진행되도록 해야 함
	2. 각 서비스의 삭제 결과를 파악 해야 함
5. 하위 호환성을 보장해야 함
	1. 다음 몇 가지 인터페이스는 하위 버전(인터페이스 작업되지 않은 버전)에서도 동작해야 함
		1. 설치 되었는지 확인하는 인터페이스 (IsInstalled)
		2. 설치되어 있는 버전 확인하는 인터페이스 (GetLocalVersion)
		3. 버전이 변경되었는지 확인하는 인터페이스 (IsVersionChanged)
		4. 삭제 경로를 반환하는 인터페이스 (GetUnistallerInfo)
6. 업그레이더 인터페이스 정의
	1. 각 서비스와의 업그레이더 인터페이스 구성도

* + - 1. 각 서비스를 배포할 때 Agent Dll, Agent Dll XML파일을 같이 배포한다. 배포는 CDN에 업로드를 말함. (Agent Dll XML 형식은 Appendix 참고)
				* 서비스 설치 프로그램에서 Agnet Dll을 설치하지 않는다.
				* 통합 인스톨러와 업데이터에서 Agent Dll XML을 참고하여 Agent Dll을 다운로드 및 등록한다.
				* Agent Dll 배포의 주체 : 각 서비스
				* Agent Dll 다운로드 및 설치의 주체 : 통합 인스톨러 및 업데이터
			2. 통합 인스톨러 / 업데이터는 Agent XML을 이용하여 Agent Dll의 버전 정보를 확인, 다운로드하고 공용 Agent 정보 보관소에 등록한다. (파일로 기록 예정)
			3. 통합 인스톨러 / 업데이터는 각 서비스의 Agent Dll의 인터페이스 함수를 호출하여 각 서비스의 버전 확인, 업데이트 기능, 업데이트 결과 확인 등을 수행한다.
	1. AgentDll 필수 빌드 환경
1. Issue:

VS2005에서 빌드된 DLL의 경우 Side By Side 의 문제가 발생하므로 런타임 라이브러리를 다중 스레드(/MT) 형태로 링크하지 않으면 LoadLibrary 호출 시 문제가 발생한다.

|  |  |
| --- | --- |
| 문제 상황( Agent DLL 의 빌드 상황) | 결과(LoadLibrary 호출 결과) |
| 컴파일러 VS2005 - (다중 스레드 DLL ) /MD 링크 | LoadLibrary 호출시 해당 DLL이 존재하는 디렉토리에 manifest 와 사용 DLL 들이 없다면 LoadLibrary 실패 |
| 컴파일러 VS2005 - (다중 스레드) /MT 링크 | LoadLibrary 호출시 정상 호출 |
| 컴파일러 VS2003 - /MD 링크 | LoadLibrary 호출시 정상 호출 |
| 컴파일러 VS2003 - /MT 링크 | LoadLibrary 호출시 정상 호출 |

이로 인해 Agent DLL 에 대해 LoadLibrary 에 문제가 발생할 수 있으므로 AgentDLL 빌드시에 다음과 같이 런타임 라이브러리를 정적으로 링킹되도록 수정해야 한다.( VS2003에서는 빌드한 DLL의 경우는 정상적으로 동작하는 것으로 보이나, 사용하는 DLL이 없을 경우 문제가 발생할 수 있으므로 /MT 옵션을 이용해서 빌드해 주시길 바랍니다.)

1. 해결책
	1. IDE: VS2003 또는 VS2005
	2. 속성 설정: C/C++ -> 코드 생성 ->런타임 라이브러리 설정 -> 다중 스레드(/MT), 다중 스레드 디버그(/MTd)



* 1. Agent Dll 필수 항목

각 서비스의 Agent Dll은 통합인스톨러 / 업데이터에서 요청 시 다음과 같은 기능을 수행해야 한다.

1. 서비스 ID 반환 (Admin Page에 입력되는 서비스 GUID와 같아야 함)
2. 로컬 메인 버전 반환
3. 서버 메인 버전 반환
4. 메인 버전 변경 여부
5. 업그레이더 실행
	1. 사일런트 모드로 업그레이더 구동 방법
	2. 일반 업데이트 방식의 구동 방법
	3. 업데이트 이외의 동작은 하지 않고 업데이트만 수행하는 방법
6. 업그레이더 실행 결과 확인
7. 삭제 프로그램 경로 반환

\*삭제 프로그램은 설치 프로그램처럼 NSIS형태의 프로그램이므로 설치/삭제 프로그램 인터페이스 를 따른다.

* 1. Agent DLL 배포
		1. 배포 주체

 \* Agent DLL의 배포는 각 서스테이닝 팀에서 담당한다. 단 배포 때, 배포 파일의 위치를 통합인스톨러 Admin 운영자에게 알려줘야 한다.(운영자는 차후 지정)

* + 1. 배포 방법

1. Agent DLL 파일과 AgentDLL xml 정보 파일을 CDN에 배포한다.

2. 통합인스톨러 서비스 Admin 에게 해당 위치를 알려준다.

3. Admin 은 관리페이지에서 각 서스테이닝 팀의 Agent DLL 과 XML 파일의 경로를 등록한다.

4. 통합인스톨러(통합업데이터) 에서는 해당 정보를 가져와서 Agent DLL 과 XML 파일을 다운로드 받아서 자체적으로 관리한다.

* + 1. 서비스 별 GUID

|  |  |
| --- | --- |
| 서비스명 | GUID |
| 내PC검색 | 45DD4898-23ED-4940-A7E1-FA54757B47C0 |
| 네이버폰 | F4EBF667-826A-4ddd-A97E-8FA41791BE8E |
| 데스크바 | 0C951B20-ECD0-4187-837D-5EF493FC20C1 |
| 데스크톱 | DD108EC6-CF48-4fa1-9EB9-F6C066D8E35A |
| 비디오매니저 | 07093FD5-C8DA-4acd-B093-535F1AFC84B7 |
| 툴바 | 39B419B3-D960-46b2-97CB-82F5BD61799B |
| 포토매니저 | C239AE19-431C-4f6b-A73E-853EF95E62CC |
| 포토뷰어 | 07FD7005-E505-41cd-899E-D168249DDD6F |
| PC그린 | 5EC74483-8CE1-42d7-A361-C46C42F9D6C4 |
| 미니캘린더 | 9958363F-8095-4da9-B6B2-B69757C640B4 |

* 1. 통합인스톨러 Agent DLL 관리 동작 시퀀스
		1. 데이터 모델
			1. 디렉토리 구조

Agent DLL 을 지정된 디렉토리에 트리 형태로 저장된다.

* 공용 Agent Directory 구조( 공용디렉토리/NHN Corporation/Common\UpgradeInterface )

|  |  |
| --- | --- |
| OS | 공용 디렉토리 경로  |
| Windows 2000 | C:\Documents And Settings\All Users\Application Data |
| Windows XP | C:\Documents And Settings\All Users\Application Data |
| Windows Vista | C:\ProgramData |

* + 1. 기본 구조



* + 1. 상세 설계

1. GUID를 이용하여 Agent DLL 찾기를 시작한다.

2. 공용 디렉토리를 가져온다.(공용 디렉토리는 CLSID\_COMMON\_APPDATA 로 찾을 수 있는 디렉토리)

3. 공용 디렉토리에서 GUID 디렉토리를 찾는다.

 GUID 디렉토리명 = 공용 디렉토리명 + GUID명

 A. If( 디렉토리가 없으면 )

 Agent DLL 다운로드

4. Version.XML 파일을 분석한다.

 A. If( Version.XML 파일이 없으면 )

 Agent DLL 다운로드

5. Version.XML 에서 받은 DLL Name으로 파일을 찾는다.

 A.If( Version.XML 파일이 없으면 )

 Agent DLL 다운로드

6. 해당 Agent DLL 정보를 넘겨준다.

* 1. AgentDll에서 작성되어야 할 함수 규약 – Appendix의 헤더파일 참고
		1. Request
* 업데이트에 필요한 각종 정보 및 실행를 요청하는 함수, 추후 추가되는 요구사항에 유연하게 대처하는 통합 함수임
* 함수 규약
	+ 함수 원형 : int Request (const char\* query, char\* response, int response\_length)
	+ Parameter
		1. query : 각종 정보 요청 XML 데이터
			- query의 값은 다음과 같은 형식의 xml 데이터가 전달된다.

Attribute로 ID가 전달되고 설정 value가 필요한 ID에는 value 엘리먼트가 전달된다.

(value가 있는 ID : agreement.shown, upgrader.run)

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> **<query id = “query id”> <value>설정값</value> </query>**</root> |

* + 1. response : [In/Out] 버퍼에 XML형태로 요청된 값 반환 또는 실행에 필요한 값을 전달한다.
			- response의 값은 아래 queryID목록 표를 참고로 다음과 같이 리턴 해야 한다.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> <queryID>요청값</queryID> **<error>** **<id>오류ID</id>** **<message>기타 오류 메시지</message>** **</error>**</root> |

* + - * <queryID>는 queryid 이름으로 한다.
			* 각 query에서 실패 또는 오류 발생시 <error> 정보를 만들어서 보내야 한다.
				+ 각 query에서 발생할 수 있는 오류가 있다면 <error> 엘리먼트를 이용하여 오류를 반환하면 된다.
				+ <message>는 <id>가 NQUERY\_ERROR\_ETC 일 경우에만 각 AgentDll에서 사용자에게 보여질 메시지를 <message> 엘리먼트에 넣어서 반환한다.
				+ <error> 엘리먼트는 모든 Query의 **필수 사항은 아니며** 각 쿼리에서 AgentDll에서 처리중 오류를 반환해야 할 필요가 있는 경우에만 리턴하도록 한다.
				+ [5.1.2 쿼리 오류 정의 참고](#쿼리오류정의)
		1. response\_length : 버퍼 길이
	+ return value : response 길이, -1이면 실패

표 queryID 목록 (\*\* 표시는 하단에 추가 설명 함)

|  |  |  |
| --- | --- | --- |
| query id | Response value | 설명 |
| \*agreement.shown | <agreement.shown>true</agreement.shown><agreement.shown>false</agreement.shown> | 업데이트 시 통합인스톨러에서 약관 창 띄웠음을 설정, query 성공 실패를 response 값으로 알려 주어야 함 |
| <value>true</value><value>false</value> |
| agreement.isshow | <agreement.isshow>true</agreement.isshow><agreement.isshow>false</agreement.isshow> | 업데이트시 통합인스톨러에서 약관창을 보여주어야 하는가에 대한 여부  |
| agreement.url | <agreement.url> <num>2</num> <url.0>문자열</url.0> <url.1>문자열</url.1></agreement.url> | <num>에 전체 약관 URL 개수 반환<url.0><url.1>에 전체약관 경로 삽입 |
| guid | <guid>문자열</guid> | 서비스 GUID 반환 |
| isinstalled | <isinstalled>true</isinstalled><isinstalled>false</isinstalled> | 설치되어 있는지 여부 반환 |
| uninstaller.url  | <uninstaller.url><type>0</type><url>문자열</url>**<parameter>문자열</parameter>**</uninstaller.url><type> 값은 [6.1 Appendix 공통 헤더 파일 참조](#공통헤더파일) (NAVERAPP\_UNISTALLER\_TYPE) | <type> 언인스톨러 타입 반환<url> 언인스톨러 URL 반환**<parameter> 언인스톨러에 사용되어야 하는 command parameter (필요한 경우에만 사용)** |
| version.local | <version.local>문자열</version.local> | 로컬 버전 반환 |
| version.recent | <version.recent>문자열</version.recent> | 최신 버전 반환 |
| version.ischanged | <version.ischanged>true</version.ischanged><version.ischanged>false</version.ischanged> | 버전 변경 여부 반환 |
| \*upgrader.run | <upgrader.run>true</upgrader.run><upgrader.run>false</upgrader.run> | 업그레이더 실행 요청, 업그레이더 실행 결과를 response 값으로 알려주어야 함 |
| <value>0</value><value>1</value><value>2</value>[6.1 Appendix 공통 헤더 파일 참조](#공통헤더파일) (NAVERAPP\_UPDATE\_STATUS) |
| upgrader.status | <upgrader.status>0</upgrader.status><upgrader.status>1</upgrader.status><upgrader.status>2</upgrader.status><upgrader.status>3</upgrader.status>[6.1 Appendix 공통 헤더 파일 참조](#공통헤더파일) (NAVERAPP\_UNISTALLER\_TYPE) | 업그레이더 실행 상태 반환 |
| \*\*install.error | <install.error>true</ install.error><install.error>false</ install.error>\* true시에는 반드시 <error>를 구성해야 한다. | 설치프로그램 실행 중 발생한 오류 여부와 오류 코드, 오류 문자열 반환오류가 없으면 false를 리턴한다. |
| uninstall.error | <uninstall.error>true</uninstall.error><uninstall.error>false</uninstall.error>\* true시에는 반드시 <error>를 구성해야 한다. | 삭제 프로그램 실행 중 발생한 오류 여부와 오류 코드, 오류 문자열 반환오류가 없으면 false를 리턴한다.install.error 스펙과 동일, 3.6.1.3 install.error 참고 |

* + - 1. agreement.shown

통합 인스톨러/업데이터에서 약관 보았다는 것을 Agent Dll을 통해서 서비스 프로그램에 알려주는 것으로서 다음과 같이 query id와 value가 같이 전달된다.

표 query 예제

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> **<query id = “agreement.shown”> <value>true</value> </query>**</root> |

요청한 결과를 다음과 같이 response 로 넘겨주면 된다.

표 response 예제

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> **< agreement.shown >true</ agreement.shown >**</root> |

* + - 1. upgrader.run

통합 인스톨러/업데이터에서 Agent Dll을 통해서 서비스 프로그램의 업그레이드를 요청하는 것으로서 다음과 같이 query id와 value가 같이 전달된다.

표 query 예제

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> **<query id = “upgrader.run”> <value>2</value> </query>**</root> |

요청한 결과를 다음과 같이 response 로 넘겨주면 된다.

표 response 예제

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> **<upgrader.run>true</upgrader.run>**</root> |

* + - 1. install.error

사일런트 설치 시 [4.5 설치/삭제 프로그램의 Exit Code](#설치삭제프로그램ExitCode정의)로 설치 프로그램의 오류 상태를 정확히 알려줄 수 없기 때문에 설치 프로그램에서 오류 발생시에 메시지를 파일 또는 레지스트리 또는 기타 방법으로 저장하였다가 AgentDll에서 install.error 쿼리 발생시 오류 여부와 오류 메시지를 리턴 하여 주어야 한다.

오류가 없는 경우에는 <install.error>false<install.error>를 리턴하면 된다.

오류 ID, 오류 메시지 저장 방법 및 로드 방법은 각 서비스에서 결정하여 사용하도록 한다.

표 response 예제

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?><root> <install.error>true</install.error> <error> <id>1000</id> <message>비디오매니저 사용시 필요한 다이렉트X를 설치해야만 합니다. DirectX를 설치하 신 후 비디오매니저 설치를 진행해 주시기 바랍니다.</message> </error></root> |

< [6.1.1.2 기본구조(설치 오류) : 설치 오류 흐름도 참고](#설치오류흐름도) >

1. 설치/삭제 프로그램 인터페이스
	1. 인터페이스 시나리오
2. 통합 인스톨러 / 업데이터에서 해당 서비스의 설치 또는 삭제 프로그램을 실행 한다.
3. 통합 인스톨러 / 업데이터에서 설치 또는 삭제 프로세스를 감시하고 프로세스가 종료되면 GetExitCode() 함수로 종료 코드를 확인하여 성공, 취소, 실패 등을 파악한다.
	* 사일런트 설치 시 설치 도중 오류가 발생할 경우 사용자가 어떤 오류인지 파악할 수 없으므로 오류 메시지를 파일 또는 레지스트리등에 기록하여둔다.
	* AgentDll은 <install.errormessage>쿼리가 오면 기입된 오류 메시지를 반환한다. 기입되어 있지 않을 경우에는 -1을 리턴하면 된다.
4. 통합 인스톨러 / 업데이터에서 Agent Dll을 이용하여 설치 또는 삭제된 것을 파악한다.
	1. 인터페이스 규약
5. 설치 프로그램과 삭제 프로그램은 NSIS 형태의 프로그램이고 실행 파라미터를 통해서 인터페이스를 정의한다.
6. 파라미터 규약은 스페이스로 구분되고 기호는 슬래시(/)로 시작된다. 기호뒤에 특정 정보가 올때는 따옴표(“)로 묶는다. 모든 파라미터 기호는 같이 사용될 수 있다.
7. NSIS형태의 설치/삭제 프로그램이 아닌 경우에도 아래 인터페이스 규약에 맞추어 개발되어야 한다.
8. 기호는 반드시 대문자만 허용한다.
9. **네이버 툴즈인스톨러에서 설치/삭제 프로그램을 호출 할 때는 반드시 /NT 파라미터를 넘긴다.**
10. **네이버 툴즈업데이터에서 설치/삭제 프로그램을 호출 할 때는 반드시 /NT 파라미터를 넘긴다.**

실행파일.exe 기호1 기호2 …

ex:XXXXSetup.exe /NT /M /O
XXXXUninstall.exe /NT /M /O
XXXXSetup.exe /NT

 SI: Service Installer, SD: Service Uninstaller,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 기호 | 설명 | 대상 | 구분 | 연관 |
| /NT | [In] 네이버 툴즈인스톨러업데이터에서 호출하는 경우 넘김, 이 파라미터를 확인하여 설치/삭제시에 발생하는 오류 메시지를 AgentDll과 통신하도록 함 | SI, SD | Parameter |  |
| /M | [In] Silent(Mute) 진행 | SI, SD | Parameter |  |
| /M |
| /O | [In] 설치, 업데이트, 삭제 등만 진행하도록 하고 서비스에 연결된 프로세스 실행하지 않도록 함 | SI, SD | Parameter |  |
| /O |
| /D　 | [In] 설치 경로 설정한다. /M와 같이 사용된다. 　 | 　SI | Parameter　 |  |
| /D C:\NaverApp |

* 1. 설치 프로그램 인터페이스 정의
1. 설치프로그램 호출 시 인터페이스

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter 항목 | 기호 | 설명 | 비고 |  |
| Silent(Mute) | /M | UI는 보여주지 않고 Silent로 설치를 진행한다. (NSIS에서는 제공됨) | 기호 없는 경우 서비스의 기본 설치 진행 |  |
| Only Install | /O | 설치만 실행하도록 한다. 서비스를 실행하지 않음 | 기호 없는 경우 서비스의 기본 설치 진행 |  |
| 설치 폴더 결정 | /D | 설치 디렉토리 설정하는 파라미터이다. /M와 사용되는 옵션이다. | 기호 없는 경우 기본 경로로 설치하거나 사용자의 선택한 경로에 설치 진행 |  |
| 호출자: 네이버 툴즈인스톨러/업데이터 | /NT | 중복 설치 오류 등 설치 오류시 오류 메시지를 AgentDll에 전달한다. | 네이버 툴즈인스톨러/업데이터가 호출할 경우 항상 있음 |  |

* 1. 삭제 프로그램 인터페이스 정의
1. 삭제프로그램 호출 시 인터페이스

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter 항목 | 기호 | 설명 | 비고 |  |
| Silent(Mute) | /M | UI는 보여주지 않고 Silent로 삭제를 진행한다. | 기호 없는 경우 서비스의 기본 삭제 진행 |  |
| Only Delete | /O | 삭제만 실행하도록 한다. 기타 서비스 실행하지 않음 (삭제 이후의 웹페이지 띄우는 동작 등) | 기호 없는 경우 서비스의 기본 삭제 진행 |  |
| 호출자: 네이버 툴즈업데이터 | /NT | 중복 삭제 오류 등 삭제 오류시 오류 메시지를 AgentDll에 전달한다. | 네이버 툴즈업데이터에서 호출할 경우 항상 있음 |  |

* 1. 설치/삭제 프로그램 Exit Code 정의
1. NSIS에서 기본 적으로 제공하는 Exit Code는 다음과 같다. SetErrorLevel (NSIS 2.01이후 제공 함수) 등의 함수로 아래 기본값을 변경해서는 안 된다.
2. NSIS 프로그램이 아닌 설치, 삭제 프로그램도 아래 Exit 코드 규약을 따라야 한다.
* 0 - 보통 실행 (에러 없음)
* 1 - 사용자에 의해 중단 (취소 버튼)
* 2 - 스크립트에 의해 중단
	1. 설치/삭제 프로그램의 /M 처리

파라미터에 /M이 있는 경우 다음과 같은 스크립트를 작성하면 된다.

|  |
| --- |
| SetSilent Silent |

1. Status 정의
	* 1. 서비스 업데이트 종료 시 결과

|  |  |  |
| --- | --- | --- |
| 　결과 내용 | 설명 | 결과 번호 |
| 성공 | 업데이트 성공 | 0 |
| 업데이트 취소 | 사용자의 업데이트 취소 | 1 |
| 업데이트 오류 | 용량 부족이나 파일 복사오류에 의한 업데이트 오류 | 2 |
| ~~인터넷 오류~~ | ~~인터넷 연결 해제 등으로 업데이트 모듈을 다운로드 받지 못하여 발생한 오류~~ | ~~3~~ |
| 업데이트 중 | 업데이트 실행 중 | 3 |
| . |  |  |
| 기타 오류 | 프로세스가 중단되거나 시스템 다운등의 알 수 없는 오류 | 1000 |

* + 1. 쿼리 오류 정의

|  |  |  |
| --- | --- | --- |
| 쿼리 오류 ID | 내용 |  |
| NQUERY\_ERROR\_NONE | 오류 없음, 오류 없을 경우 <error> 엘리먼트 생성할 필요 없음 | 0 |
| NQUERY\_ERROR\_WRONGQUERY | 알 수 없는 쿼리이거나 잘못된 쿼리 | 1 |
| NQUERY\_ERROR\_NOTENOUGHBUF | Response 버퍼가 작아서 response data를 만들지 못할 경우 | 2 |
| NQUERY\_ERROR\_NETWORK | 네트워크 오류, 쿼리 진행 중 네트워크에 의해 진행되지 않을 경우 | 3 |
| NQUERY\_ERROR\_ALREADYRUNUPGRADER | 업그레이더가 이미 실행 중 이어서 업그레이더 실행에 실패한 경우 | 4 |
| NQUERY\_ERROR\_NOUPGRADERFILE | 업그레이더 파일 없어서 업그레이더 실행에 실패한 경우 | 5 |
| NQUERY\_ERROR\_ETC | 기타 오류, <message>엘리먼트에 사용자에게 보여줄 오류 메시지를 같이 만들어서 넘겨주어야 함 | 1000 |
|  |  |  |

* + 1. 서비스 설치 종료 시 결과

|  |  |  |
| --- | --- | --- |
| 　결과 내용 | 설명 | 결과 번호 |
| 성공 | 설치 성공 | 0 |
| 설치 취소 | 사용자의 설치 취소 | 1 |
| 설치 중단 | 스크립트에 의한 중단 | 2 |

* + 1. 서비스 삭제 종료 시 결과

|  |  |  |
| --- | --- | --- |
| 　결과 내용 | 설명 | 결과 번호 |
| 성공 | 삭제 성공 | 0 |
| 삭제 취소 | 사용자의 삭제 취소 | 1 |
| 설치 중단 | 스크립트에 의한 중단 | 2 |

1. 설치/업데이트/삭제 시 Agent DLL 사용 시나리오
	1. 설치 시 Agent DLL 사용 시나리오
		1. 통합인스톨러/통합업데이터 설치 시 Agent DLL 사용 시나리오
			1. 기본 구조 (설치)



1. Agent DLL 을 찾는다. : 지정된 위치의 레지스트리 검색 후, 실제 파일이 존재하는지 확인

2. Agent DLL의 Request( IsInstalled ) 를 호출한다. 인스톨 되었다면 TRUE, 아니면 FALSE를 리턴한다. 이미 설치 되었다면 설치 프로세스를 종료한다.

* + - 1. 기본 구조 (설치 오류)



1. 통합 인스톨러는 설치 프로그램을 사일런트로 실행한다.
2. 사일런트로 설치 실행 중 내부적인 오류 발생한다. (준비 모듈이 설치되어 있지 않거나 NSIS에서 필수적인 메시지를 띄워서 안내 후에 다시 진행해야 하는 경우)
3. NSIS 설치 프로그램에서 파일 또는 레지스트리에 오류 메시지를 기입한다.
4. 통합 인스톨러는 설치 프로세스가 종료된 이후에 AgentDll에 설치 오류 메시지가 있는지 확인한다.
5. AgentDll은 오류메시지를 파일 또는 레지스트리에서 읽어 온다.
6. 통합 인스톨러에 오류 코드 및 오류 메시지를 반환한다.
	1. 업그레이드 시 Agent DLL 사용 시나리오
		1. 통합인스톨러에 의한 업그레이드 시 Agent DLL 사용 시나리오
			1. 기본 구조



1. Request( isinstalled ) 함수를 호출해서 인스톨 되어있는지 체크한다.

2. 설치 되지 않았다면, 인스틀을 진행한다.

3. 설치 되었다면 Request(version. Ischanged) 함수를 사용해서 최신 버전인지 체크한다.

4. 최신 버전이라면 작업을 중단한다.

5. 구버전이라면 Request( upgrader.run : type )을 호출해서 업그레이드를 실시한다.

4. 업그레이드가 끝날 때 까지 Request( upgrader.status ) 함수를 지속적으로 호출해서 업그레이드 상태를 확인한다. 실패라면 실패로 이동

* + 1. 통합업데이터에 의한 업그레이드 시 Agent DLL 사용 시나리오
			1. 기본구조



1. Request( isinstalled ) 함수를 호출해서 인스톨 되어있는지 체크한다.

2. 설치 되지 않았다면, 인스틀을 진행한다.

3. 설치 되었다면 Request(version. Ischanged) 함수를 사용해서 최신 버전인지 체크한다.

4. 최신 버전이라면 작업을 중단한다.

5. 구버전이라면 먼저 Request(agreement.isshow) 를 이용하여 약관 표시 여부를 확인한다.

6. 약관을 표시 해야 한다면 개별 약관을 표시하고 Request( agreement.shown, true ) 를 호출해서 Agent DLL 에게 약관을 표시했다는 것을 알려준다.

7. Request( upgrader.run : type )을 호출해서 업그레이드를 실시한다.

4. 업그레이드가 끝날 때 까지 Request( upgrader.status ) 함수를 지속적으로 호출해서 업그레이드 상태를 확인한다. 실패라면 실패로 이동

* 1. 삭제 시 Agent DLL 사용 시나리오 시퀀스
		1. 기본 구조



1. Request(uninstaller.url) 를 호출한다. Uninstaller Path와 언인스톨러 Type을 가져온다.

2. 언인스톨러를 호출한다.

1. Agent DLL 배포 규약
2. Agent DLL 배포는 각 서비스의 배포 담당자가 담당한다. (CDN 업로드)
3. 배포시에는 AgentDll 파일과 AgentDll XML 파일을 같이 배포해야 한다.
4. AgentDll XML 형식 예제

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?><root> <agentdll> <!-- Agent DLL 버전 --> <version>0.0.0.1</version> <!-- Agent DLL 경로 --> <url>http://cdn.naver.com/naver/client/PhotoUpdateAgent.dll</url> </agentdll></root> |

1. XML 경로를 통합 인스톨러/업데이터 관리 페이지에 등록하여 다운로드 및 실행을 통합 인스톨러/업데이터에서 담당하도록 한다.
2. 각 서비스에서의 1차 리뷰 사항

기획적으로 정리되어야 할 사항은 기호 앞에 \* 또는 \*\*표시함
(\* : 각 서비스의 기획 이슈, \*\* : 통합 인스톨러의 기획이슈)

1. 툴바 / 데스크바 / 내PC검색 (담당자 : 김형래)

|  |  |  |
| --- | --- | --- |
|  | Question | Answer |
| 1-A | A.     AgentDLL 배포 주체 -> 이 부분은 어플에 defendency 한 부분 이므로 통합 인스톨러가 가져갈 필요는 없다고 생각됩니다. -> 최악의 경우에는 통합 인스톨러 팀 이슈로 어플이 업그레이드를 못할 수도….. -> 배포는 각 어플에서 하고 AgentDLL이 업그레이드 될 경우 역시 어플에서 수행하면 됩니다. ->만약 AgentDLL 이 존재하지 않는 경우가 크리티컬한 이슈라면 QA 테스트케이스에 포함하면 됩니다 | 인터페이스 구성도에 명기함- 배포의 주체 : 각 서비스- 설치의 주체 : 통합 업데이터/인스톨러 |
| 1-B | B. AgentDLL 비스타 권한 이슈 -> 비스타인 경우 AgentDLL을 로딩하는 프로세스가 어드민이냐 일반이냐에 따라 문제가 발생 할 수 있습니다. | AgentDll은 일반 모드로 실행함 |
| 1-C | C. /N 옵션 -> 툴바에서 이미 사용하는 옵션이므로 변경을 요청드립니다. | /O로 변경함 |

1. 포토 매니저 / 포토 뷰어 (담당자 : 김상욱)

|  |  |  |
| --- | --- | --- |
|  | Question | Answer |
| \*2-A | 통합 인스톨러에서 각 서비스의 약관 업데이트 지원 여부 | 기획적으로 시나리오 정리되어야 함 |

1. 네이버 폰 (담당자 : 김상은)

|  |  |  |
| --- | --- | --- |
|  | Question | Answer |
| \*\*3-A | 통합인스톨러에서 업데이트 목록을 보여주는 경우에 Interface를 통하지 않고 서버에서 받아오는것으로 보입니다. 이렇게 되면 실제 업데이트 관리와 다르게 이중으로 관리되어야 하는데요, 이렇게 2중으로 운영한다는 기준을 가지고 있는것이지요? | 운영정책에 따라 이중으로 관리될 수 있음.기획적으로 정리가 되어야 할 사항임 |
| \*3-B | RunUpgrader 실행시에 이미 어플이 떠있는 경우는 어떻게 동작해야 하는건가요? 어플이 종료하는 경우는 사용자 입력을 받아야 하는 경우도 있습니다. (검사치료중 또는 네이버폰 통화중 등등) | 기획적으로 정리가 되어야 할 사항임(특정 리턴 코드로 업그레이드 실패를 리턴하거나 강제 종료 후 업그레이드) |

1. PC 그린 (담당자 : 우병찬)

|  |  |  |
| --- | --- | --- |
|  | Question | Answer |
| 4-A | 1.     GetUpdateStatus 인터페이스네이버폰과 PC그린은 Agent내에 폴링쓰레드를 생성해서 상태를 체크하고 업그레이드가 종료시 ExitCode를 반환하는 방법으로 구현하기로 되어있는 상태입니다.참고 의견을 드리자면, 이 함수의 경우 통합인스톨러 프로세스(Agent.dll)와 각서비스프로세스간에 프로세스 통신이 이루어 져야 될거 같은데요 ShareMemory, Message, File, Registry..등을 사용해야 할거 같은데 각 서비스마다 알아서 처리해도 되지만 표준이 있으면 좋을 거 같습니다.또, 단순히 업데이트 상황을 감시하기 위한 함수 치고는 spec이 크다는 생각이 듭니다. 그래서 상태중에서 종료와 관련된 상태(취소종료, 성공종료, 오류종료)를 단순히 종료로 단순화 시키는 건 어떨까 합니다. | 1. 프로세스간의 표준 통신 방법 : 회의를 통해서 요구사항을 취합하여 결정2. 종료코드 단순화는 검토후에 결정 |
| \*4-B | 2. 최신업데이트 xml의 경우 2중(통합인스톨러, 서비스)으로 운영이 될거라 보여지는데 인터페이스 중 최신xml을 가져오는 부분이 있는데 어느쪽 xml로 접근하는 지에 대한 결정된 사항이 있는지 궁금합니다. | 기획적으로 정리가 되어야 할 사항임(네이버 폰의 첫번째 항목, 3-A와 같은 내용) |
| \*4-C | 3. 약관 및 RunUpgrade 인터페이스관련 내용은 아직 협의 가 완료 안된 상태라 이부분 스펙에 대한 공수는 현재로선 잡기가 어렵습니다.특히 이미 실행중이거나 사일런트 업그레이더시에는 각 서비스 마다 사용자 액션이 있을 수 있는데 이때에 대한 처리도 정리가 안되어있는 상태입니다. | 기획적으로 정리가 되어야 할 사항임(2-A, 3-B와 같은 내용) |

1. Appendix
	1. 공통 헤더 파일

|  |
| --- |
| /\*\*\* @enum NAVERAPP\_UPDATE\_TYPE\* @brief 업데이트 타입을 정의한 값이다.\*/enum NAVERAPP\_UPDATE\_TYPE{ NUPDATE\_TYPE\_NORMAL = 0, NUPDATE\_TYPE\_SILENT = 0x01, NUPDATE\_TYPE\_ONLYUPDATE = 0x02,};/\*\*\* @enum NAVERAPP\_UPDATE\_STATUS\* @brief 업데이트 상태를 정의한 값이다.\*/enum NAVERAPP\_UPDATE\_STATUS{ NUPDATE\_STATUS\_SUCCESS = 0, NUPDATE\_STATUS\_CANCEL, NUPDATE\_STATUS\_ERROR, NUPDATE\_STATUS\_STILL\_ACTIVE, NUPDATE\_STATUS\_ETC\_ERROR = 1000,};/\*\*\* @enum NAVERAPP\_ERROR\* @brief 각종쿼리의 오류를 정의한것이다.\*/enum NAVERAPP\_QUERY\_ERROR{ NQUERY\_ERROR\_NONE = 0, // 오류 없음NQUERY\_ERROR\_WRONGQUERY, // 잘못된 쿼리 임NQUERY\_ERROR\_NOTENOUGHBUF, // 버퍼가 부족함 NQUERY\_ERROR\_NETWORK, // 네트워크 오류 NQUERY\_ERROR\_ALREADYUPGRADERRUN, // 이미 업그레이더 실행 되어있음NQUERY\_ERROR\_NOUPGRADERFILE, // 업그레이더 파일 없음 // 추후 오류 더 추가될 수 있음 NQUERY\_ERROR\_ETC = 1000, // 기타 오류, 메시지필요함};/\*\*\* @enum NAVERAPP\_UNISTALLER\_TYPE\* @brief 언인스톨러타입을정의한값이다.\*/enum NAVERAPP\_UNISTALLER\_TYPE{ NUINSTALLER\_TYPE\_NSIS = 0, NUINSTALLER\_TYPE\_APP,};BOOL IsInstalled();BOOL GetServiceID(LPTSTR pGUID, int bufferLen);BOOL RunUpgrader(int runType);BOOL GetUpdateStatus(int& status);BOOL GetUnistallerPath(TCHAR\* uninstallerPath, int pathBufferLen);BOOL GetLocalVersion(TCHAR\* localVersion, int verBufferLen);BOOL GetRecentVersion(TCHAR\* recentVersion, int verBufferLen);BOOL IsVersionChanged(); |

* 1. NSIS 프로그램에서 실행 파라미터 확인 방법

|  |
| --- |
| Call GetParameterspop $0MessageBox MB\_OK $0 |

* 1. Agetn Dll XML 형식

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?><root><agentdll><version>0.0.0.1</version><url>http://cdn.naver.com/naver/client/PhotoUpdateAgent.dll</url></agentdll></root> |

* 1. Agent Dll샘플 구현 예제
		1. Type A – 포토매니저

\*\* GetLocalPath - Export되지 않는 내부 함수

|  |
| --- |
| BOOL PhotoViewerAgent::GetLocalPath(TCHAR\* localPath, int pathBufferLen){ TCHAR szValue[MAX\_PATH]; ULONG nBufLen = MAX\_PATH; BOOL bInstall = FALSE; CRegKey reg; LONG lResult = reg.Open(HKEY\_LOCAL\_MACHINE, PHOTOMANAGER\_KEYNAME); if(ERROR\_SUCCESS == lResult) { lResult = reg.QueryStringValue(PHOTOMANAGER\_INSVALUE, szValue, &nBufLen); if(lResult == ERROR\_SUCCESS) { \_tcscpy\_s(localPath, pathBufferLen, szValue); }  } reg.Close(); return (lResult == ERROR\_SUCCESS);} |

1. IsInstalled - Export되지 않는 내부 함수

|  |
| --- |
| BOOL PhotoViewerAgent::IsInstalled(){ TCHAR szValue[MAX\_PATH] = {0, }; if(GetLocalPath(szValue, MAX\_PATH)) { return TRUE; } else { return FALSE; }} |

1. GetServiceID - Export되지 않는 내부 함수

|  |
| --- |
| int PhotoViewerAgent::GetServiceID(LPTSTR pGUID, int bufferLen){ TCHAR szValue[MAX\_PATH] = {0, }; if(GetLocalPath(szValue, MAX\_PATH)) { int guidLen = \_tcslen(GUID\_PHOTOMANAGER); if (pGUID) { errno\_t err = \_tcscpy\_s(pGUID, bufferLen, GUID\_PHOTOMANAGER); if (err) guidLen = -1; } return guidLen; } else { return -1; }} |

1. RunUpgrader - Export되지 않는 내부 함수

현재는 runType를 실행할 때 파라미터등으로 주는 인터페이스가 되어 있지 않아서 런처를 실행하는 것으로 구현되어 있음

실제 구현 시에는 runType에 맞게 업그레이더를 실행할 수 있어야 함, 또한 Type A계열은 업그레이더의 업데이트 이슈가 있기 때문에 런처를 실행하여 업그레이더를 실행하기 때문에 런처를 실행하도록 구현해 놓음.

|  |
| --- |
| BOOL PhotoViewerAgent::RunUpgrader(int runType){ TCHAR installPath[MAX\_PATH] = {0, }; if(GetLocalPath(installPath, MAX\_PATH)) { ATLPath::Append(installPath, \_T("\\")); ATLPath::Append(installPath, PHOTOMANAGER\_LAUNCHER);// runType에 따라서 실행방법을 달리해야함 ShellExecute(NULL, NULL, installPath, NULL, NULL, SW\_SHOW); m\_IsRunUpgrader = TRUE; return TRUE; } else { return FALSE; }} |

1. GetUpdateStatus - Export되지 않는 내부 함수

현재는 프로세스가 있는지 없는지 만 체크하여 업데이트 중, 업데이트 종료의 상태를 알려주도록 되어 있는데 실제 구현 시에는 진행 중, 업데이트 성공, 업데이트 취소, 업데이트 오류 등의 상태를 알 수 있어야 함

업그레이더 실행 시 런처를 실행하여 업그레이더를 실행하도록 구현해 놓았기 때문에 런처와 업그레이더의 프로세스를 같이 체크하도록 구현해 놓음

|  |
| --- |
| BOOL PhotoViewerAgent::GetUpdateStatus(int& status){ DWORD launcherProcessId = FindProcessId(PHOTOMANAGER\_LAUNCHER); DWORD upgraderProcessId = FindProcessId(PHOTOMANAGER\_UPGRADER); if (m\_IsRunUpgrader)  { if (launcherProcessId == DWORD(-1) && upgraderProcessId == DWORD(-1)) { status = RESULT\_SUCCESS; m\_IsRunUpgrader = FALSE; } else { status = RESULT\_STILL\_ACTIVE; } return TRUE; } else { return FALSE; }} |

1. GetUnistallerInfo - Export되지 않는 내부 함수

|  |
| --- |
| int PhotoViewerAgent::GetUnistallerInfo(TCHAR\* unistallerPath, int pathBufferLen, int& unistallerType){ CRegKey reg; LONG lResult; TCHAR szValue[MAX\_PATH] = {0, }; ULONG valueLen = MAX\_PATH; lResult = reg.Open(HKEY\_LOCAL\_MACHINE, UNINSTALL\_PHOTOMANAGER\_KEYNAME); if(ERROR\_SUCCESS == lResult) { lResult = reg.QueryStringValue(UNINSTALL\_VALUE, szValue, &valueLen); if(ERROR\_SUCCESS == lResult && unistallerPath) { \_tcscpy\_s(unistallerPath, pathBufferLen, szValue); unistallerType = NUINSTALLER\_TYPE\_NSIS; } } reg.Close(); if (ERROR\_SUCCESS == lResult) return valueLen; else return -1;} |

1. GetLocalVersion - Export되지 않는 내부 함수

임의로 ‘mainver’ 엘리먼트 값을 리턴하도록 해 놓음, 서비스 마다 대표적인 로컬 버전을 리턴하도록 해야 함

|  |
| --- |
| int PhotoViewerAgent::GetLocalVersion(TCHAR\* localVersion, int verBufferLen){ TCHAR localPath[MAX\_PATH] = {0 , }; if(!GetLocalPath(localPath, MAX\_PATH)) return -1; ATLPath::Append(localPath, \_T("\\")); ATLPath::Append(localPath, PM\_LAUNCHAR\_CONFIG\_XML); TiXmlDocument doc; char \*pFilePath = XMLUtil::Encode\_utf8(localPath); bool isLoadFile = doc.LoadFile(pFilePath); delete pFilePath; if (!isLoadFile) return -1; TiXmlElement\* pElement = doc.FirstChildElement(("root")); if(pElement == NULL) return -1; pElement = pElement->FirstChildElement("Info"); if(pElement == NULL) return -1; pElement = pElement->FirstChildElement("mainver"); if(pElement == NULL) return -1; TCHAR szTemp[MAX\_PATH] = {0, }; if (localVersion == NULL) { localVersion = szTemp; verBufferLen = MAX\_PATH; } if(XMLUtil::GetValue(pElement, localVersion, verBufferLen)) { return \_tcslen(localVersion); } return -1;} |

1. GetRecentVersion - Export되지 않는 내부 함수

로컬 버전과 마찬가지로 대표적인 최신 버전을 리턴하도록 해야 함

|  |
| --- |
| int PhotoViewerAgent::GetRecentVersion(TCHAR\* recentVersion, int verBufferLen){ TCHAR serviceInfoURL[MAX\_PATH] = {0, }; if(!GetServiceInfoURL(serviceInfoURL, MAX\_PATH)) return -1; tipbase::GenericHTTPClient httpClient; if(!httpClient.Request(NULL, serviceInfoURL)) return -1; TiXmlDocument serviceInfoDoc; serviceInfoDoc.Parse((char\*)httpClient.QueryHTTPResponse()); TiXmlElement\* pElement = serviceInfoDoc.FirstChildElement("root"); if(pElement == NULL) return -1; pElement = pElement->FirstChildElement("info"); if(pElement == NULL) return -1; pElement = pElement->FirstChildElement("mainver"); if(pElement == NULL) return -1; TCHAR szTemp[MAX\_PATH] = {0, }; if (recentVersion == NULL) { recentVersion = szTemp; verBufferLen = MAX\_PATH; } if(XMLUtil::GetValue(pElement, recentVersion, verBufferLen)) { return \_tcslen(recentVersion); } return -1;} |

1. IsVersionChanged - Export되지 않는 내부 함수

버전 변경은 문자열 비교로 구현해 놓았는데 실제 구현 시에는 서비스마다 메이저 업데이트를 판별할 수 있는 로직이 포함되어야 함

|  |
| --- |
| BOOL PhotoViewerAgent::IsVersionChanged(){ TCHAR localVer[MAX\_PATH] = {0 ,}; TCHAR recentVer[MAX\_PATH] = {0, }; if(!GetLocalVersion(localVer, MAX\_PATH)) return FALSE; if(!GetRecentVersion(recentVer, MAX\_PATH)) return FALSE; if(\_tcscmp(localVer, recentVer) == 0) { return FALSE; } else { return TRUE; }} |

* + 1. Type B - PC그린
1. IsInstalled – Type A 참고
2. GetServiceID – Type A 참고
3. RunUpgrader- Export되지 않는 내부 함수

Type B는 업그레이더가 런처의 기능도 있기 때문에 업그레이더를 실행 시킴

|  |
| --- |
| BOOL PCGreenAgent::RunUpgrader(int runType){ TCHAR upgraderPath[MAX\_PATH] = {0, }; if(GetLocalPath(upgraderPath, MAX\_PATH)) { ATLPath::Append(upgraderPath, \_T("\\")); ATLPath::Append(upgraderPath, PCGREEN\_UPGRADER); TCHAR szParam[] = \_T("/show"); // runType에 따라서 실행방법을 달리해야함 ShellExecute(NULL, NULL, upgraderPath, NULL, NULL, SW\_SHOW); m\_IsRunUpgrader = TRUE; return TRUE; } else { return FALSE; }} |

1. GetUpdateStatus - Export되지 않는 내부 함수

Type B는 업그레이더가 업그레이더를 temp에 복사해놓고 실행하므로 업그레이더 프로세스만 감시하면 되어서 다음과 같이 구현함.

실제 구현 시에는 진행 중, 업데이트 성공, 업데이트 취소, 업데이트 오류 등의 상태를 알 수 있어야 함

|  |
| --- |
| BOOL PCGreenAgent::GetUpdateStatus(int& status){ if (m\_IsRunUpgrader)  { DWORD upgraderProcessId = FindProcessId(PCGREEN\_UPGRADER); if (upgraderProcessId == DWORD(-1)) { status = RESULT\_SUCCESS; m\_IsRunUpgrader = FALSE; return TRUE; } status = RESULT\_STILL\_ACTIVE; return TRUE; } else { return FALSE; }} |

1. GetUnistallerPath – Type A 참고
2. GetLocalVersion - Export되지 않는 내부 함수

임의로 ‘upgrade\_desc’ 엘리먼트 값을 메인 버전으로 리턴하도록 해 놓음. 실제 구현 시에는 대표적인 버전을 리턴하도록 해야 함

|  |
| --- |
| int PCGreenAgent::GetLocalVersion(TCHAR\* localVersion, int verBufferLen){ TCHAR xmlPath[MAX\_PATH] = {0, }; if(!GetLocalPath(xmlPath, MAX\_PATH)) return -1; ATLPath::Append(xmlPath, \_T("\\")); ATLPath::Append(xmlPath, PCG\_UPGRADE\_XML);  TiXmlDocument doc; char \*pFilePath = XMLUtil::Encode\_utf8(xmlPath); bool isLoaded = doc.LoadFile(pFilePath); delete pFilePath; if (!isLoaded) return -1; TiXmlElement\* pElement = doc.FirstChildElement(("application")); if(pElement == NULL) return -1; pElement = pElement->FirstChildElement("upgrade\_desc"); if(pElement == NULL) return -1; TCHAR szTemp[MAX\_PATH] = {0, }; if (localVersion == NULL) { localVersion = szTemp; verBufferLen = MAX\_PATH; } if(XMLUtil::GetValue(pElement, localVersion, verBufferLen)) { return \_tcslen(localVersion); } return -1;} |

1. GetRecentVersion - Export되지 않는 내부 함수

|  |
| --- |
| BOOL PCGreenAgent::GetRecentVersion(TCHAR\* recentVer, int verBufferLen){ TCHAR serverXMLPath[MAX\_PATH] = {0, }; if(!GetURL\_ServerXML(serverXMLPath, MAX\_PATH)) return FALSE; tipbase::GenericHTTPClient httpClient; if(!httpClient.Request(NULL, serverXMLPath)) { return FALSE; } TiXmlDocument doc; doc.Parse((char\*)httpClient.QueryHTTPResponse()); TiXmlElement\* pElement = doc.FirstChildElement("application"); if(pElement == NULL) return FALSE; pElement = pElement->FirstChildElement("upgrade\_desc"); if(pElement == NULL) return FALSE; if(!XMLUtil::GetValue(pElement, recentVer, verBufferLen)) return FALSE; return TRUE;} |

1. IsVersionChanged – Type A 참고
	* 1. Type C – 툴바
2. IsInstalled – Type A 참고
3. GetServiceID – Type A 참고
4. RunUpgrader - Export되지 않는 내부 함수

Type C는 메인 업그레이드는 설치 파일을 다운로드 받아서 실행하는 것이므로 설치 파일을 다운로드하여 실행하는 것으로 구현 함

|  |
| --- |
| BOOL ToolbarAgent::RunUpgrader(int runType){ tipbase::GenericHTTPClient httpClient; if(!httpClient.Request(NULL, URL\_DLL\_UPDATE)) { return FALSE; }  TiXmlDocument upgradeInfoDoc; upgradeInfoDoc.Parse((char\*)httpClient.QueryHTTPResponse()); TiXmlElement\* pNode = upgradeInfoDoc.FirstChildElement("naver\_toolbar\_info"); if(pNode == NULL) return FALSE; pNode = pNode->FirstChildElement("SETUP"); if(pNode == NULL) return FALSE; TCHAR Server\_SetupPath[MAX\_PATH] = {0, }; if(!XMLUtil::GetValue(pNode, Server\_SetupPath, MAX\_PATH)) return FALSE; TCHAR Local\_SetupPath[MAX\_PATH] = {0, }; \_tcscpy\_s(Local\_SetupPath, MAX\_PATH, ::PathFindFileName(Server\_SetupPath) ); TCHAR szTempFileName[MAX\_PATH] = {0, };  ::GetTempPath(COUNTOF(szTempFileName), szTempFileName); ::PathAppend(szTempFileName, Local\_SetupPath); ::DeleteUrlCacheEntry(Server\_SetupPath); if(!httpClient.Request(NULL, Server\_SetupPath, tipbase::GenericHTTPClient::RequestGetMethod, szTempFileName)) { return FALSE; } ::ShellExecute(NULL,\_T("open"),szTempFileName,\_T(""),\_T(""),SW\_SHOW); m\_IsRunUpgrader = TRUE; return TRUE;} |

1. GetUpdateStatus – Type C 참고

프로세스 검사하는 식으로 구현해 놓았지만 실제 구현 시에는 설치 중, 설치 완료, 설치 취소, 설치 오류 등의 상태를 리턴해야 함

1. GetUnistallerPath – Type A 참고
2. GetLocalVersion - Export되지 않는 내부 함수

DLL 버전을 리턴 함

|  |
| --- |
| BOOL ToolbarAgent::GetLocalVersion(TCHAR\* localVersion, int verBufferLen){ TCHAR localPath[MAX\_PATH] = {0 , }; if(!GetLocalPath(localPath, MAX\_PATH)) return FALSE; ATLPath::Append(localPath, \_T("\\")); ATLPath::Append(localPath, TOOLBAR\_TBINFO\_INI); CIni ini(localPath); return ini.GetString(\_T("DLL"), \_T("version"), localVersion, verBufferLen);} |

1. GetRecentVersion - Export되지 않는 내부 함수

|  |
| --- |
| BOOL ToolbarAgent::GetRecentVersion(TCHAR\* recentVersion, int verBufferLen){ tipbase::GenericHTTPClient httpClient; if(!httpClient.Request(NULL, URL\_DLL\_UPDATE)) { return FALSE; } TiXmlDocument upgradeInfoDoc; upgradeInfoDoc.Parse((char\*)httpClient.QueryHTTPResponse()); TiXmlElement\* pElement = upgradeInfoDoc.FirstChildElement("naver\_toolbar\_info"); if(pElement == NULL) return -1; pElement = pElement->FirstChildElement("VERSION"); if(pElement == NULL) return -1; TCHAR szTemp[MAX\_PATH] = {0, }; if (recentVersion == NULL) { recentVersion = szTemp; verBufferLen = MAX\_PATH; } if(XMLUtil::GetValue(pElement, recentVersion, verBufferLen)) { return \_tcslen(recentVersion); } return -1;} |

1. IsVersionChanged – Type A 참고
2. FAQ

|  |  |
| --- | --- |
| 질문  | 답변 |
| Agreement.isshow 와 Agreement.shown 의 동작은 정확히 어떻게 되는 건가요? 연속적으로 두 함수가 호출될 수 있나요? 그러면 그 순서는 어떻게 되나요? | Agreement.isshow 는 통합인스톨러/통합업데이터가 현재 약관이 변경된 사항이 있어서 사용자에게 약관을 표시해야 되는지를 AgentDLL에게 물어보는 함수입니다. 만약 약관을 보여줘야 해서, 통합업데이터 등에서 먼저 약관을 보여주게 되면, 해당 프로그램에서 다시 약관을 보여줄 필요가 없다는 것을 알려주기 위해서 agreement.shown 을 이용합니다. |
| NUPDATE\_STATUS\_ERROR, NUPDATE\_STATUS\_ETC\_ERROR 는 어떤 차이가 있나요? | 정상적인 진행과정에서 에러가 발생하면 NUPDATE\_STATUS\_ERROR 를 발생하고, 그 외에는 전부 NUPDATE\_STATUS\_ETC\_ERROR를 리턴합니다. Ex) AgentDLL의 upgrader.run 을 호출하기 전에 upgrader.status 를 호출하게 되면 NUPDATE\_STATUS\_ETC\_ERROR, 진행중의 하드 디스크나 메모리 부족은 NUPDATE\_NUPDATE\_STATUS\_ERROR 가 된다. |
| Upgrader.status 는 얼마나 자주 호출되나요? | 아직 정확하게 정해진 시간 스펙은 없습니다만, 최소 1초에 한번 이상 호출됩니다. |