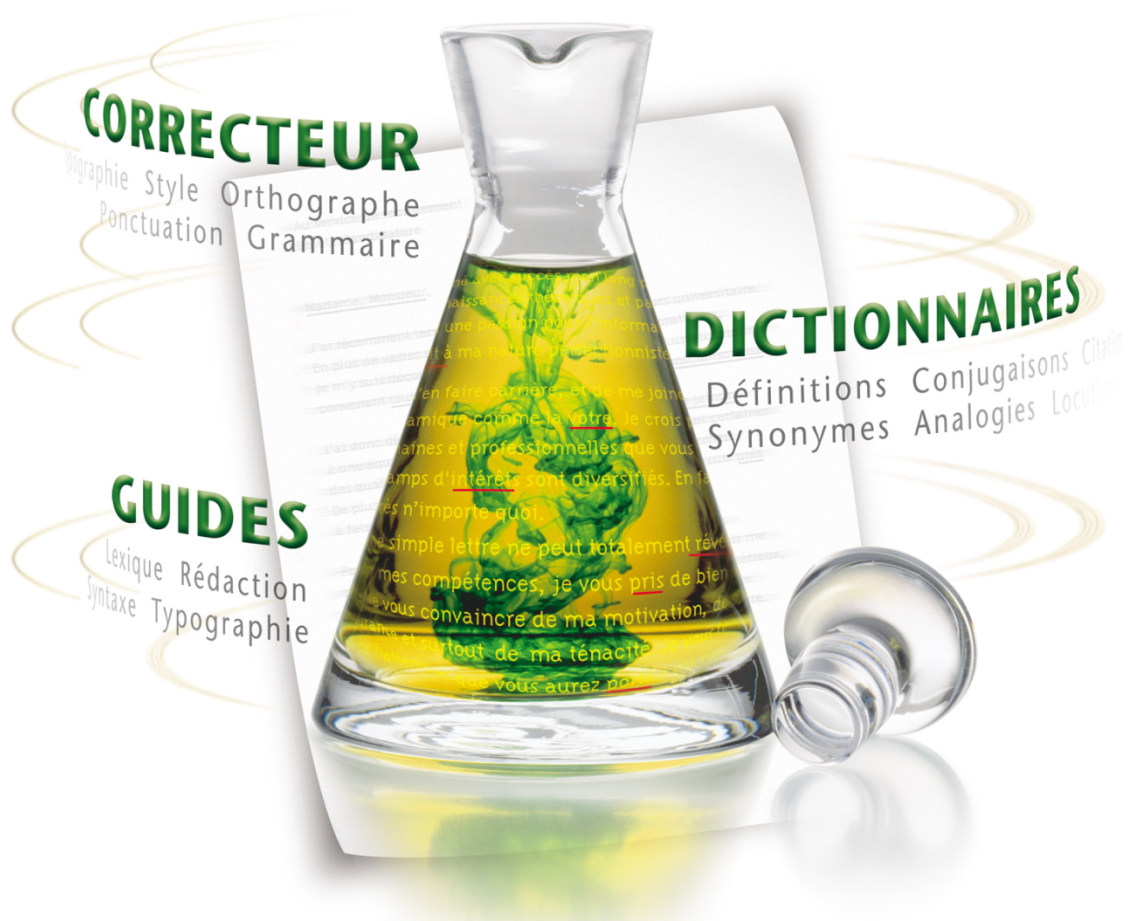


# API v2.0 pour Antidote

## Windows - COM

---



## Table des matières

### 1. Introduction

### 2. Principes de base

### 3. Estimation du temps et de la complexité

### 4. Description des interfaces COM

#### 4.1 Accès à l'interface COM d'Antidote

#### 4.2 Création de l'interface COM dans votre application

### 5. Communication

#### 5.1 Première étape: démarrage d'Antidote

#### 5.2 Deuxième étape: appel d'un outil d'Antidote

##### 5.2.1 Liste des méthodes de l'interface IApiOle d'Antidote

##### 5.2.2 Exemple en pseudocode pour appeler le correcteur

#### 5.3 Troisième étape: réponse aux demandes d'Antidote

##### 5.3.1 Identificateurs pour les documents et les zones de texte

##### 5.3.2 Liste des méthodes appelées par Antidote

### 6. Activation du journal

### 7. Exemple complet (C++ et C#)

### 8. Compatibilité des API

## Important

Si votre application implémente déjà une version de l'API d'Antidote, consultez la section 8.

## 1. Introduction

L'API d'Antidote est une interface qui vous permet d'appeler les services d'Antidote depuis votre application Windows. Cette API est basée sur la technologie COM (Component Object Model) de Microsoft.

Dans le présent document, vous trouverez les informations nécessaires pour établir la communication entre votre application et Antidote.

## 2. Principes de base

- a) Votre application doit permettre l'accès au texte à l'aide des positions de caractères.  
Exemple: Antidote peut demander à votre application de lui retourner le texte qui se trouve entre les caractères 1023 et 1725.
- b) Votre infrastructure de programmation (langage, librairies, etc.) doit vous donner accès à l'interface COM d'Antidote.
- c) Vous devez pouvoir implanter une interface COM dans votre application.
- d) Antidote et votre application doivent s'exécuter en parallèle; l'API leur permet simplement de communiquer.

## 3. Estimation du temps et de la complexité

Complexité : facile si les principes ci-dessus sont déjà appliqués, moyen sinon.

Temps : 5 jours-personne si les principes ci-dessus sont déjà appliqués  
sinon, dépend de la complexité de l'application.

## 4. Description des interfaces COM

### 4.1 Accès à l'interface COM d'Antidote

Pour appeler les outils d'Antidote, vous devez utiliser l'interface **IApiOle** exportée par Antidote. Cette interface est la seule interface publique d'Antidote. Les autres interfaces sont pour usage interne de Druide et peuvent changer sans préavis.

L'interface **IApiOle** contient les méthodes suivantes. Ces méthodes sont décrites en détail à la section 5.2.1. Les autres méthodes non documentées de l'API existent pour rétrocompatibilité uniquement.

```
void LanceOutil2 (BSTR classe, BSTR outil, BSTR langue, BSTR versionAPI);
void LanceOutilDispatch2(IDispatch* disp, BSTR outil, BSTR langue, BSTR versionAPI);
```

### 4.2 Création de l'interface COM dans votre application

Pour qu'Antidote puisse communiquer avec votre application, vous devez y créer une interface COM. Cette interface doit être spécifique à Antidote.

L'interface que vous devez créer doit contenir les méthodes suivantes. Ces méthodes sont décrites en détail à la section 5.3.2.

```
HRESULT ActiveApplication(void);
HRESULT ActiveDocument([in] LONG idDoc);
HRESULT DonneDebutSelection([in] LONG idDoc, [in] LONG idZone, [out, retval] LONG *retour);
HRESULT DonneFinSelection([in] LONG idDoc, [in] LONG idZone, [out, retval] LONG *retour);
HRESULT DonneIdDocumentCourant([out, retval] LONG *retour);
HRESULT DonneIdZoneDeTexte([in] LONG idDoc, [in] LONG indice, [out, retval] LONG *retour);
HRESULT DonneIdZoneDeTexteCourante([in] LONG idDoc, [out, retval] LONG *retour);
HRESULT DonneIntervalle([in] LONG idDoc, [in] LONG idZone, [in] LONG leDebut, [in] LONG laFin, [out, retval] BSTR *retour);
HRESULT DonneLongueurZoneDeTexte([in] LONG idDoc, [in] LONG idZone, [out, retval] LONG *retour);
HRESULT DonneNbZonesDeTexte([in] LONG idDoc, [out, retval] LONG *retour);
HRESULT DonnePolice([in] LONG idDoc, [in] LONG idZone, [out, retval] BSTR *retour);
HRESULT DonneTitreDocCourant([out, retval] BSTR* retour);
HRESULT RemplaceIntervalle([in] LONG idDoc, [in] LONG idZone, [in] LONG leDebut, [in] LONG laFin, [in] BSTR laChaine);
HRESULT SelectionneIntervalle([in] LONG idDoc, [in] LONG idZone, [in] LONG leDebut, [in] LONG laFin);
```

## 5. Communication

### 5.1 Première étape: démarrage d'Antidote

Avant d'appeler un outil d'Antidote à partir de votre application, il faut s'assurer qu'Antidote est ouvert. Si ce n'est pas le cas, votre application doit démarrer Antidote.

Le dossier où est installé Antidote est enregistré dans les registres de Windows dans la clé suivante :

HKEY\_LOCAL\_MACHINE\SOFTWARE\Druide informatique inc.\Antidote

La valeur à consulter est :

DossierAntidote

Votre application doit consulter cette valeur pour connaître l'emplacement d'Antidote et l'exécuter avec le paramètre « -activex ». Ensuite, votre application doit attendre qu'Antidote soit complètement ouvert avant de poursuivre. Pour savoir si Antidote est ouvert, votre application peut tester l'existence de la fenêtre d'Antidote.

Exemple : Consultez la méthode `LancerAntidote()` du fichier `AntidoteApiOle.h` pour connaître les détails d'implémentation (voir la section 7).

Une version de pont doit être fournie à Antidote. Ce numéro de version vous permet d'interagir avec différentes éditions d'Antidote (différentes versions de l'API).

Vous devez indiquer la version de l'API le plus récent supporté à la fois par Antidote et votre application. La version supportée par Antidote se trouve dans les registres dans la même clé que mentionnée précédemment. La valeur à consulter est :

VersionAPI

Par exemple :

si vous implémentez l'API version 2.0 et 2.1 dans votre logiciel  
et qu'Antidote implémente l'API 2.2  
alors vous devez instancier l'API 2.1 dans votre application et passer 2.1  
comme version dans `LanceOutil`.

### 5.2 Deuxième étape: appel d'un outil d'Antidote

Votre application utilisera les méthodes disponibles dans l'interface **IApiOle** pour demander l'ouverture d'un outil d'Antidote.

Le progId de l'interface **IApiOle** est : `Antidote.ApiOle`

Important : Il faut trouver le serveur COM d'Antidote dans la "Running object table" (ROT) plutôt que d'utiliser `::CoCreateInstance()`. La méthode `TrouverServeurCOMDansLaROT()` du fichier `AntidoteApiOle.h` démontre comment y parvenir (voir la section 7).

### 5.2.1 Liste des méthodes de l'interface IApiOle d'Antidote

void **LanceOutil2** (BSTR\* szClientProgId, BSTR\* szOutil, BSTR\* langue, BSTR\* versionAPI);

*Description:* Demande l'ouverture d'un outil (correcteur, dictionnaires, guides).

*Paramètres:* szClientProgId : le ProgID de l'interface que vous avez créée pour Antidote (voir la section 4.2). Cette méthode nécessite que votre serveur COM soit un singleton, car Antidote utilisera ::CoCreateInstance() sur ce progId.  
szOutil : le nom de l'outil désiré. Consultez les constantes dans le namespace CstApiAntidote du fichier AntidoteApiOle.h pour connaître les valeurs possibles.  
langue: la langue dans laquelle vous souhaitez effectuer la recherche. Consultez les constantes dans le namespace CstApiAntidote du fichier AntidoteApiOle.h pour connaître les valeurs possibles.  
versionAPI: la version de l'API implémenté par votre texteur (2.0 pour cette documentation).

void **LanceOutilDispatch2** (IDispatch\* p\_dispatch, BSTR\* szOutil, BSTR\* langue, BSTR\* versionAPI);

*Description:* Méthode similaire à LanceOutil, mais qui permet de passer à Antidote le pointeur IDispatch qui servira à rappeler votre serveur.

Exemple : Consultez la classe AntidoteApiOle LancerAntidote() du fichier AntidoteApiOle.h pour connaître les détails d'implémentation (voir la section 7).

### 5.2.2 Exemple en pseudocode pour appeler le correcteur

(voir aussi la voir la section 7 pour un exemple en C++ et en C#)

```

si(AntidoteDejaLancé() OU LancerAntidote())
{
    AttendreQueAntidoteSoitPret();

    ServeurApiDeAntidote ant;
    si(TrouverServeurCOMDansLaROT(ant)) // Running Object Table
    {
        ant.LanceOutilDispatch(
            pointeurSurMonPropreServeurCom,
            CstApiAntidote::kModuleCorrecteur,
            languePourCetteRecherche,
            versionDeAPI,
        );
    }
}

```

### 5.3 Troisième étape: réponse aux demandes d'Antidote

Pour ouvrir un outil, Antidote doit faire certaines demandes à votre application. Antidote appellera les méthodes qui se trouvent dans l'interface COM que vous avez créée dans votre application pour Antidote (voir la section 4.2).

#### 5.3.1 Identificateurs pour les documents et les zones de texte

Dans le cas où vos documents sont divisés en plusieurs zones de texte (ex.: entête, note de bas de page, etc.), Antidote peut les corriger en même temps que le reste du document. Votre application doit alors gérer une liste des zones de texte pour chacun des documents. Antidote appellera la méthode `DonneIdZoneDeTexte(LONG idDocument, LONG indiceZoneDeTexte)` pour chacune des zones de texte du document en passant l'indice de la zone de texte comme deuxième paramètre. L'indice débute à 1 et correspond au rang de la zone de texte dans votre liste. Vous devez fournir un identificateur unique pour le document ainsi que pour chacune des zones de texte. Ces identificateurs seront utilisés par Antidote pour faire des requêtes à une zone de texte en particulier dans un document en particulier.

Dans le cas où vos documents ne contiennent qu'une seule zone de texte, vous devez fournir un identificateur unique pour le document et un autre pour la zone de texte.

#### 5.3.2 Liste des méthodes appelées par Antidote

`HRESULT ActiveApplication(void);`

*Description:* Active votre application pour la mettre en avant-plan.

`HRESULT ActiveDocument([in] LONG idDoc);`

*Paramètre:* idDoc : Identificateur de document.

*Description:* Active le document spécifié pour le mettre devant les autres documents de votre application sans mettre votre application en avant-plan.

`HRESULT DonneDebutSelection([in] LONG idDoc, [in] LONG idZone, [out, retval] LONG *retour);`

*Paramètres:* idDoc : Identificateur de document.

idZone : Identificateur de la zone de texte.

*Description:* Retourne la position du début de la sélection pour la zone de texte **idZone** du document **idDoc**. La position est calculée en terme de caractères par rapport au début de la zone de texte.

`HRESULT DonneFinSelection([in] LONG idDoc, [in] LONG idZone, [out, retval] LONG *retour);`

*Paramètres:* idDoc : Identificateur de document.

idZone : Identificateur de la zone de texte.

*Description:* Retourne la position de la fin de la sélection pour la zone de texte **idZone** du document **idDoc**. La position est calculée en terme de caractères par rapport au début de la zone de texte.

`HRESULT DonneIdDocumentCourant([out, retval] LONG *retour);`

*Description:* Retourne l'identificateur du document courant (le document en avant-plan dans votre application).



HRESULT DonneIdZoneDeTexte([in] LONG idDoc, [in] LONG indice, [out, retval] LONG \*retour);

*Paramètres:* idDoc : Identificateur de document.  
Indice : Indice de la zone de texte dans le document. Les indices débutent à 1.  
*Description:* Retourne l'identificateur de la zone de texte correspondant à l'indice **indice** dans le document **idDoc**.

HRESULT DonneIdZoneDeTexteCourante([in] LONG idDoc, [out, retval] LONG \*retour);

*Paramètre:* idDoc : Identificateur de document.  
*Description:* Retourne l'identificateur de la zone de texte courante (la zone de texte qui contient le curseur).

HRESULT DonneIntervalle([in] LONG idDoc, [in] LONG idZone, [in] LONG leDebut, [in] LONG laFin, [out, retval] BSTR \*retour);

*Paramètres:* idDoc : Identificateur de document.  
idZone : Identificateur de la zone de texte.  
leDebut : La position du début de l'intervalle par rapport au début de la zone de texte (en caractères).  
laFin : La position de fin de l'intervalle par rapport au début de la zone de texte (en caractères).  
*Description:* Retourne le texte de la zone de texte **idZone**, de la position **leDebut** jusqu'à la position **laFin**, dans le document **idDoc**.

HRESULT DonneLongueurZoneDeTexte([in] LONG idDoc, [in] LONG idZone, [out, retval] LONG \*retour);

*Paramètres:* idDoc : Identificateur de document.  
idZone : Identificateur de la zone de texte.  
*Description:* Retourne la longueur de la zone de texte **idZone** dans le document **idDoc**.

HRESULT DonneNbZonesDeTexte([in] LONG idDoc, [out, retval] LONG \*retour);

*Paramètre:* idDoc : Identificateur de document.  
*Description:* Retourne le nombre de zones de texte dans le document spécifié.

HRESULT DonnePolice([in] LONG idDoc, [in] LONG idZone, [out, retval] BSTR \*retour);

*Paramètres:* idDoc : Identificateur de document.  
idZone : Identificateur de la zone de texte.  
*Description:* Méthode obsolète présente pour rétrocompatibilité uniquement. Retourner une chaîne vide.

HRESULT DonneTitreDocCourant([out, retval] BSTR\* retour);

*Description:* Retourne le titre du document courant.

HRESULT RemplaceIntervalle([in] LONG idDoc, [in] LONG idZone, [in] LONG leDebut, [in] LONG laFin, [in] BSTR laChaine);

*Paramètres:* idDoc : Identificateur de document.  
idZone : Identificateur de la zone de texte.  
leDebut : La position du début de l'intervalle par rapport au début de la zone de texte (en caractères).  
laFin : La position de fin de l'intervalle par rapport au début de la zone de texte (en caractères).  
laChaine : La chaîne de caractères de remplacement.

*Description:* Remplace le texte de la zone de texte **idZone** de la position **leDebut** jusqu'à la position **laFin** par **laChaine** dans le document **idDoc**.

HRESULT SelectionneIntervalle([in] LONG idDoc, [in] LONG idZone, [in] LONG leDebut, [in] LONG laFin);

*Paramètres:*

- idDoc : Identificateur de document.
- idZone : Identificateur de la zone de texte.
- leDebut : La position du début de l'intervalle par rapport au début de la zone de texte (en caractères).
- laFin : La position de fin de l'intervalle par rapport au début de la zone de texte (en caractères).

*Description:* Sélectionne le texte de la zone de texte **idZone** de la position **leDebut** jusqu'à la position **laFin** dans le document **idDoc** (pour illustrer la progression d'Antidote).

## 6. Activation du journal

Pour faciliter le développement de votre serveur COM, il est possible d'activer un journal qui contiendra une trace de tous les appels entre Antidote et votre application. Le journal contiendra aussi les exceptions (erreurs COM), le cas échéant.

**Note :** Le journal ralentit considérablement la communication entre les 2 applications et ne devrait être utilisé que lors du développement.

Pour activer le journal :

1. Ouvrir le fichier `Config.xml` avec un éditeur de texte supportant les retours de charriot Linux (par exemple Wordpad).  
Le chemin par défaut de ce fichier est :  
`C:\Programmes\Druides\Connectix 10\Application\Config`
2. Ajouter l'entrée suivante dans la section `<Choix>`.  
**Attention**, les entrées de ce fichier sont sensibles à la case.  

```
<Choix>
<JournalApiCOM>1</JournalApiCOM>
</Choix>
```
3. Quitter puis relancer l'AgentConnectix.
4. Le journal sera créé lors de la prochaine interaction avec votre serveur COM. Son emplacement sera :  
`C:\Users\[nom de l'utilisateur]\AppData\Roaming\Druides\Connectix\JournalApiCOM.txt`

## 7. Exemple complet (C++ et C#)

Le dossier compressé contenant ce PDF contient aussi deux exemples de texteur générique dans lesquels s'intègre Antidote. L'un est en C++ et l'autre en C# (code source et exécutable). Ces exemples fournissent un bon point de départ pour intégrer Antidote dans votre application. Vous pouvez copier-coller une bonne part de l'exemple directement dans votre application.

Pour plus de détails, consultez les documents "**Lisez-moi.txt**" dans le dossier des exemples.

## 8. Compatibilité des API

Puisque les API COM d'Antidote sont rétrocompatibles, il est possible d'écrire votre application pour qu'elle soit compatible à plusieurs éditions d'Antidote en utilisant l'API correspondant à la version la plus ancienne.

Le premier API COM était disponible dans Antidote Prisme (5).  
L'API DDE fournie avec Antidote MP (4) n'est plus supportée.

Nouveauté de l'API d'Antidote 9 (API 2.0) par rapport à l'API d'Antidote 8 et HD (7) :

- La version de l'API est fournie afin de pouvoir cibler plusieurs versions de l'API
- Il est possible de demander l'ouverture des dictionnaires et guides sur un mot dans une langue spécifique

Nouveauté par rapport à l'API d'Antidote RX (6) :

- Il faut maintenant lancer Antidote avec le paramètre « -activex ».

Nouveauté par rapport à l'API d'Antidote Prisme :

- Il existe maintenant une seule méthode LanceOutil plutôt que 5 (correcteur, dictionnaire, synonymes, conjugaison, grammaire).
- Il faut maintenant se connecter à Antidote par la "Running object table" (ROT) plutôt que d'utiliser ::CoCreateInstance().
- Il est recommandé d'utiliser les variantes avec Dispatch des méthodes d'appel à Antidote.