

# Compiler des projets Flex en ligne de commande

par Vincent Petithory ([Mes articles](#))

Date de publication : 21/03/2008

Dernière mise à jour :

Ce tutoriel vous expliquera comment compiler vos projets Flex en ligne de commande. Cette méthode vous permettra d'augmenter votre productivité tout en vous affranchissant de tout environnement de développement.

- I - Introduction
- II - Exploration ...
- III - Compilation simple ...
- IV - Compilation à l'aide d'un XML ...
- V - Vers l'automatisation ...
- VI - Conclusion

## I - Introduction

Vous ne vous êtes jamais dit que compiler vos projets Flex ou AIR sans vous reposer sur Flex Builder ou d'autre IDE serait plutôt pas mal ? Oui je vous vois venir, ça ne sert à rien, et pourtant :).

Cela permet de comprendre ce qui se passe dans les coulisses, et surtout, une fois maîtrisé, vous avez bien plus de contrôles sur tout ce que vous faites. C'est également une étape nécessaire pour flexouiller sans Flex Builder.

Bon bon. Oui ça peut paraître rébarbatif au début. Mais ça reste simple pour compiler un simple .as ou un MXML. Pour compiler tout un projet, ou un framework, cela demande un peu plus d'organisation, mais rien de bien compliqué.

Toutes les sources utilisées dans ce tutorial sont [ici](#) ( **miroir** ).

C'est parti :).

## II - Exploration ...

Un petit tour d'horizon de nos compileurs déjà. Ils se trouvent dans le dossier {pathFlexSDK3}\bin\. Ceux qui nous intéressent ici sont :

- Le compilateur mxmhc.exe (ainsi que le batch et le shell amxhc, qui configure mxmhc pour compiler du AIR)
- Le compilateur compc.exe (ainsi que le batch et le shell acompc, qui configure compc pour compiler une bibliothèque contenant des classes de AIR)
- Le compilateur asdoc.exe, qui permet de générer de la documentation.

Dans cet article, on ne verra que la compilation avec mxmhc. Des articles ultérieurs expliqueront comment compiler un ensemble de classes en une bibliothèque SWC, générer une documentation avec l'ASDOC, et compiler avec style vos applications AIR. Que du bonheur en perspective !

### III - Compilation simple ...

On supposera, dans la suite, qu'on se trouve dans un environnement Windows (des fichiers batch seront utilisés, il suffira de les transposer en shell pour les linuxiens).

Il faut garder à l'esprit qu'on cherche à être productif, nous allons donc utiliser des fichiers batch afin de faciliter l'appel des commandes.

Pour expérimenter, compilons un mxml simple. Le fichier batch à créer # dans le même répertoire pour cet exemple # nécessite de créer une variable mémorisant le chemin absolu ou relatif vers le compilateur mxmclc (préférez des chemins absolus pour la réutilisabilité).

```
@echo off

REM the path of the mxmclc compiler

SET mxmclcPath="C:\Program Files\Adobe\Flex SDK 3\bin\mxmclc.exe"

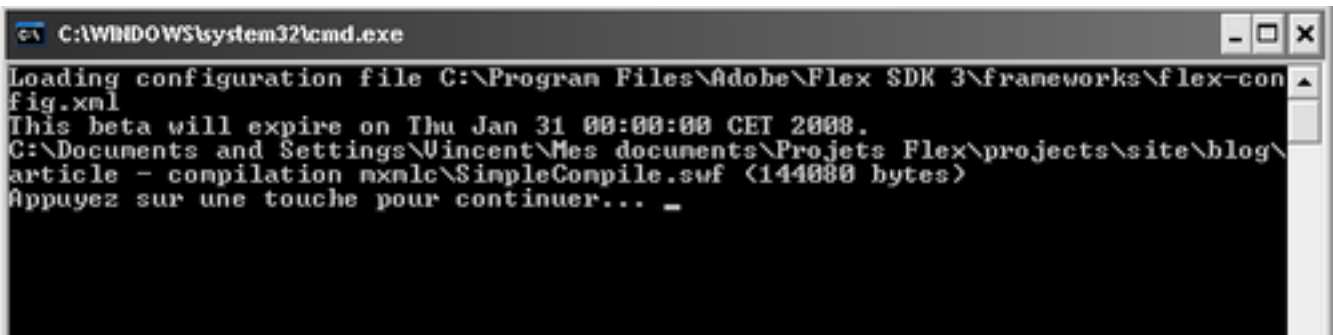
REM The mxml to compile

SET mxmlToCompile="SimpleCompile.mxml"

%mxmclcPath% %mxmlToCompile%

pause
```

On l'exécute...



```
C:\WINDOWS\system32\cmd.exe
Loading configuration file C:\Program Files\Adobe\Flex SDK 3\frameworks\flex-con
fig.xml
This beta will expire on Thu Jan 31 00:00:00 CET 2008.
C:\Documents and Settings\Vincent\Mes documents\Projets Flex\projects\site\blog\
article - compilation mxmclc\SimpleCompile.swf (144080 bytes)
Appuyez sur une touche pour continuer... _
```

Très bien, nous avons notre swf créée juste à côté de notre mxml et de son batch.

C'est bien pratique pour compiler simplement un petit mxml qu'on fait en test par exemple. Solution à ne pas comparer avec la version artillerie lourde, du style, Flex Builder -> New Project, et hop on se retrouve avec pas mal de fichiers pour... juste un test....

Ainsi, utiliser un simple batch a l'avantage d'être facile à mettre en place, et donc pratique pour des travaux de petite envergure.

## IV - Compilation à l'aide d'un XML ...

Si on veut aller plus loin, la méthode précédente ne suffit plus.

En effet, le compilateur prend en charge un nombre important d'options. La liste complète de ces options se trouve dans les sources jointes à ce tutoriel.

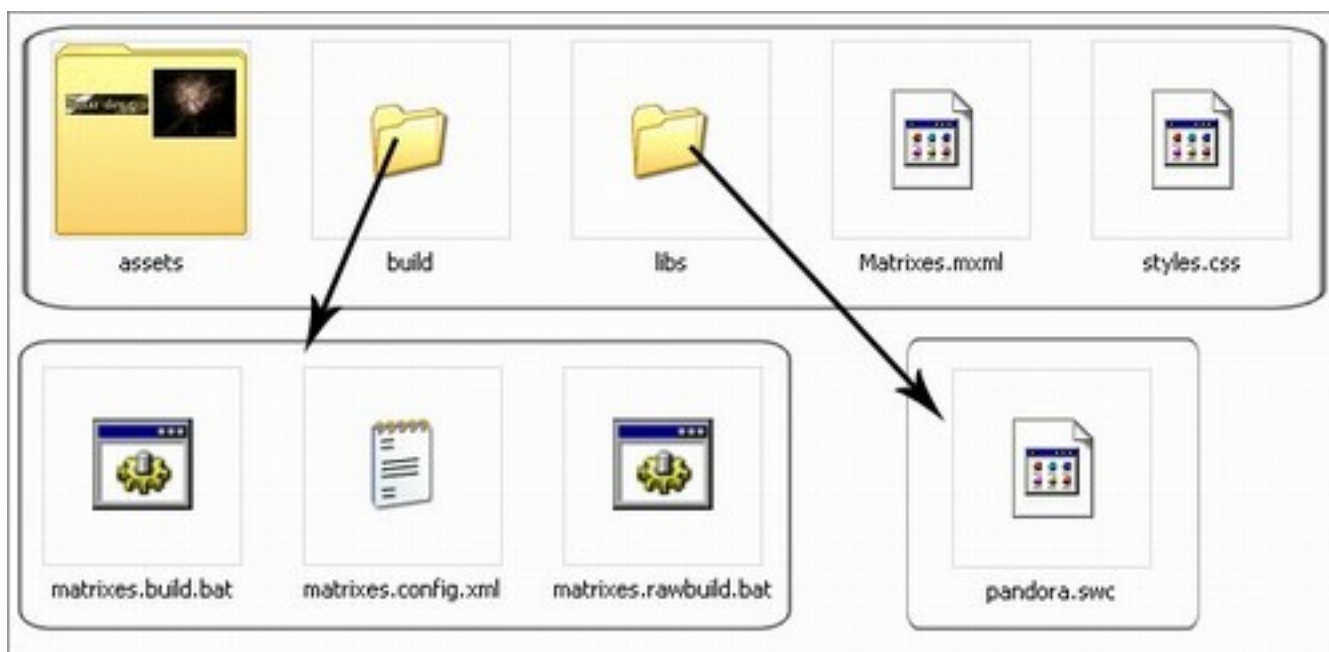
Dès lors, le batch n'est plus vraiment adapté à contenir toutes ces options. Au niveau maintenance et productivité, ce n'est plus gérable de créer une variable pour chaque option du compilateur, ou d'avoir une ligne de commandes inbuables. Nous allons donc utiliser un xml de configuration, qui sera spécifique à chaque projet, dans lequel les options du compilateur seront mémorisées.

Avant de se lancer, quelque rappels et conseils pratiques.

La bonne habitude à prendre dès lors qu'on travaille en POO est d'organiser son travail sous forme de packages. Notre projet a une racine, contenant nos packages, et d'autres éléments. Ce n'est normalement pas nouveau pour vous (j'espère). La bonne pratique est d'ajouter à cette racine d'autres dossiers types, comme par exemple un dossier 'docs', contenant notre documentation, un dossier 'bin' ou 'deploy' pour notre sortie, etc..

Une bonne idée est de faire un dossier 'build', qui contiendra tous nos fichiers de configuration pour compiler notre projet, et d'ajouter des raccourcis des batch où bon vous semble, si vous jugez que cela vous fait gagner du temps.

Je vais prendre comme exemple une mini-application que j'ai réalisé il y a quelques temps, et qui illustre l'utilisation de la classe Matrix... Un truc bateau en somme, qui n'utilise qu'un simple MXML et une bibliothèque (d'une seule classe, pour l'exemple).



On désire compiler Matrixes.mxml, mais avec certaines options. Parmi elles, le nom du swf produit, les source-path, des library à inclure, le frame-rate, des metadata, etc.

Notre batch, avec les options du compilateur devient :

```
@echo off

REM the path of the mxmmlc compiler

SET mxmmlcPath="C:\Program Files\Adobe\Flex SDK 3\bin\mxmmlc.exe"

%mxmmlcPath% -sp ../ -include-libraries ../libs/pandora.swc -use-network=false -default-size 1024
768 -default-frame-rate 25
-default-background-color #FFFFFF -o ../matrixes_BasicBuild.swf -file-specs ../Matrixes.mxml

pause
```

Assez rébarbatif n'est-ce pas ?

Voyons l'équivalent avec le xml. Voici celui qui va paramétrer les options de notre application, matrixes.config.xml :

```
<flex-config>

  <!--Compiler options -->
  <compiler>
    <source-path append="true">
<path-element>../</path-element>
    </source-path>
    <library-path append="true">
<path-element>../libs/pandora.swc</path-element>
    </library-path>
    <debug>>true</debug>
  </compiler>

  <!--Files to compile -->
  <file-specs>
<path-element>../Matrixes.mxml</path-element>
  </file-specs>

  <!--Output file -->
  <o>../matrixes.swf</o>

  <!--Other params -->
  <default-background-color>#FFFFFF</default-background-color>
  <default-frame-rate>25</default-frame-rate>

  <default-size>
    <width>1024</width>
    <height>768</height>
  </default-size>

  <!-- Enables SWFs to access the network. -->
  <use-network>>false</use-network>

</flex-config>
```

C'est déjà mieux. Certes plus 'bavard', mais bien mieux organisé.

Pour savoir comment obtenir le nom de la balise qui va se relier à la bonne option, il faut respecter toujours la même logique.

Si on regarde la doc des options du compilateur (lien de téléchargement donné plus haut), nous avons par exemple : -compiler.source-path [path-element], d'alias -sp.

Le raisonnement est le même que pour des packages. `-source-path` est une commande de compiler. Ainsi, nous le regroupons dans le XML dans une balise .

En revanche, `-o` est l'alias de la commande top-level 'output'. La balise est donc à placer directement sous la racine.

La logique est la même pour les autres commandes.

Un point important à noter est que les chemins relatifs spécifiés sont pris par rapport à l'emplacement du xml.

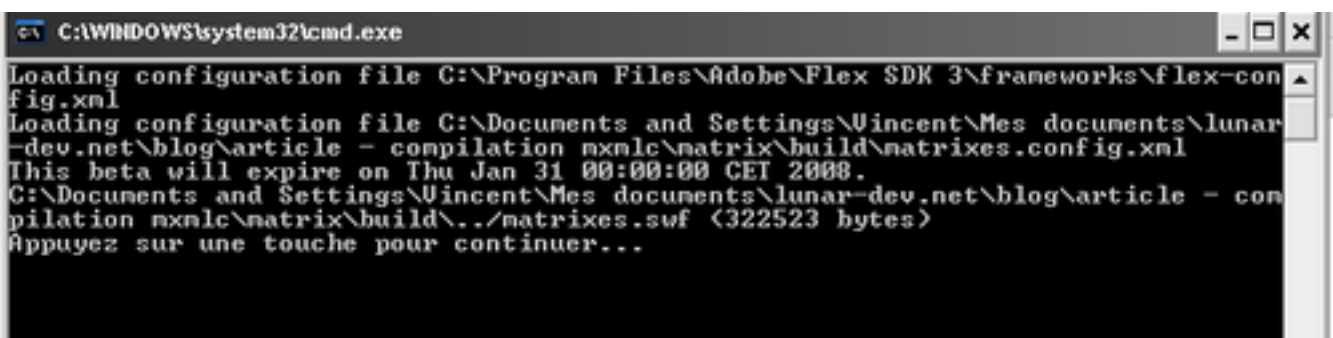
Une autre remarque : j'ai pu constater que la commande `-l`, (library-path dans le xml) utilisée en dehors du xml de configuration semble neutraliser l'import des bibliothèques chargés par le fichier de configuration par défaut (flex-config.xml). En revanche, utilisée dans un autre xml de configuration, les bibliothèques s'ajoutent normalement aux autres sans les neutraliser. Utilisez donc `-include-libraries` en ligne de commande, qui ne produit pas cette erreur. En passant, la différence entre ces deux commandes : `-include-libraries` inclut dans le swf la totalité de la bibliothèque, alors que `-l` (library-path) inclut uniquement les classes requises par l'application.

Maintenant, nous devons modifier le batch pour qu'il charge notre fichier de configuration.

Pour cela, nous utilisons la commande `-load-config`.

```
@echo off
REM the path of the mxmmlc compiler
SET mxmmlcPath="C:\Program Files\Adobe\Flex SDK 3\bin\mxmmlc.exe"
REM the path of your xml configuration
SET xmlConfigPath=matrixes.config.xml
%mxmmlcPath% -load-config+=%xmlConfigPath%
pause
```

Il n'y a plus qu'à lancer le batch.



```
C:\WINDOWS\system32\cmd.exe
Loading configuration file C:\Program Files\Adobe\Flex SDK 3\frameworks\flex-con
fig.xml
Loading configuration file C:\Documents and Settings\Vincent\Mes documents\lunar
-dev.net\blog\article - compilation mxmmlc\matrix\build\matrixes.config.xml
This beta will expire on Thu Jan 31 00:00:00 CET 2008.
C:\Documents and Settings\Vincent\Mes documents\lunar-dev.net\blog\article - con
pilation mxmmlc\matrix\build\..\matrixes.swf (322523 bytes)
Appuyez sur une touche pour continuer...
```

Remarquez la ligne supplémentaire indiquant que le compilateur a bien chargé notre xml de configuration (en plus du xml de configuration de flex par défaut).

Pour les développeurs AIR, tout ceci est valable, à ceci près qu'il faut invoquer `amxmlc.bat` au lieu de `mxmmlc.exe`. `amxmlc` charge `air-config.xml` au lieu de `flex-config.xml` afin d'inclure les classes propres à AIR. J'en dirai plus sur la compilation de projets AIR dans un autre article.

## V - Vers l'automatisation ...

Réécrire un xml de configuration à chaque projet serait fastidieux. J'utilise donc un template de xml qui convient pour 95% des projets. Je l'accompagne aussi d'un template de batch, où seule une ligne est à modifier. On gagne un temps précieux, puisqu'1 minute suffit à tout mettre en place.

De plus, on voit vite un avantage se profiler : la portabilité du xml et son implémentation dans bon nombre de langages (Java, PHP, etc.), offrent un bon panel de possibilités de gestion. Pourquoi pas une petite appli en AIR qui créerait automatiquement vos xml de configuration par exemple ?

C'est tout pour cette fois-ci ! Toute remarque est la bienvenue.

## VI - Conclusion

Dans cet article, nous avons vu comment compiler des projets Flex à l'aide des compilateurs en ligne de commande. Nous avons pu approfondir en mettant en place une méthode afin d'améliorer la rapidité de mise en service et notre productivité, tout cela sans l'utilisation d'environnements de développement. Une étape indispensable pour tout développeur Flex et Air voulant s'orienter vers des outils libres.

