

[Version de ce document](#)

[Yag et LUA](#)

[Fonctionnement général](#)

[Les structures](#)

[Les feuilles de personnages \(cs\)](#)

[Les feuilles de pions \(ps\)](#)

[Les résultats de dés \(dr\)](#)

[Origine du jet de dés \(dro\)](#)

[L'objet yag \(LUA vers Yag\)](#)

[yag.LuaDir](#)

[yag.RollDice\(f1, f2, dro, us\)](#)

[yag.ClearDice\(\)](#)

[yag.PublicMessage\(s\)](#)

[yag.LocalMessage\(s\)](#)

[yag.GetPawnInfo\(id\)](#)

[yag.UpdatePawn\(ids, ps\)](#)

[Les fonctions LUA appelées par Yag \(Yag vers LUA\)](#)

[Yag\\_CharacterSheetUpdate\(sp, cs\)](#)

[Yag\\_RollDiceButton\(sp, cs, id, f1, f2, dro\)](#)

[Yag\\_RollDiceCallback\(sp, cs, id, dr1, dr2, ui, us\)](#)

## Version de ce document

Si vous avez trouvé ce document dans votre installation de Yag, vous aurez peut être besoin de récupérer la dernière version à l'adresse suivante: <http://yagame.fr/lua-api/>

## Yag et LUA

Un des buts de Yag est de rester aussi généraliste que possible en ce qui concerne les règles.

Ca soulève le besoin de le personnaliser en fonction du jeu visé.

Mais Yag est codé en c++ et son fonctionnement interne est inaccessible aux joueurs.

Une réponse courante à ce problème est d'embarquer dans le programme hôte (Yag) un langage de script comme LUA.

Pour échanger des données, certaines parties de Yag doivent être exposées à LUA:

- Quelques fonctions et variables doivent être accessibles pour les joueurs à travers LUA
- Yag doit pouvoir appeler des fonctions LUA

Ce document décrit les outils actuellement disponible pour ce faire.

La version de LUA embarquée est la v5.3.

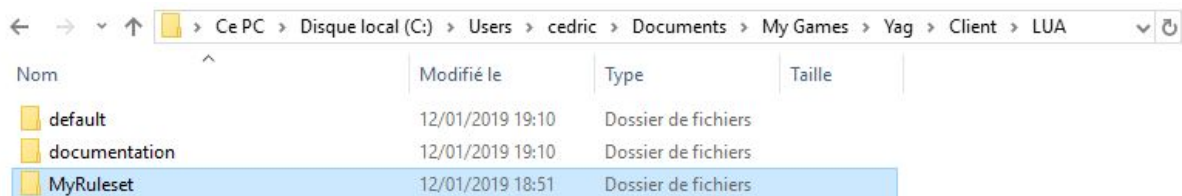
## Fonctionnement général

Pour appeler du code LUA, Yag utilise le contenu d'un répertoire qui doit:

- Se trouver dans [<my documents>/My Games/Yag/Client/LUA](#)
- Avoir un nom unique qui est spécifié dans le panneau LUA dans l'interface de Yag.

Par exemple si le nom du répertoire est [MyRuleSet](#), il doit être créé comme suit:

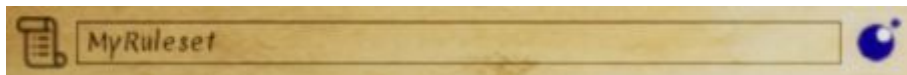
[<my documents>/My Games/Yag/Client/LUA/MyRuleSet](#)



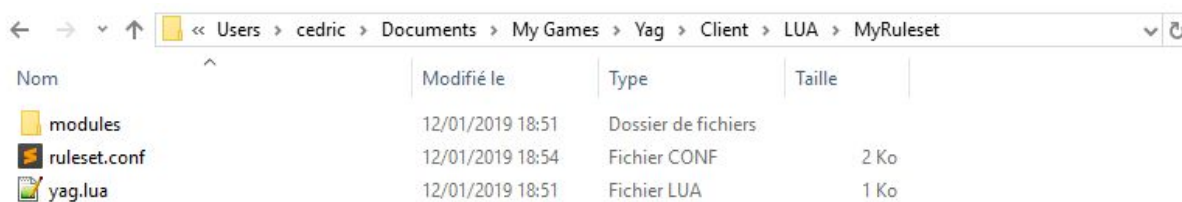
The screenshot shows a Windows Explorer window with the address bar set to "Ce PC > Disque local (C:) > Users > cedric > Documents > My Games > Yag > Client > LUA". The file list below shows three folders: "default", "documentation", and "MyRuleSet". The "MyRuleSet" folder is selected and highlighted in blue. The columns are "Nom", "Modifié le", "Type", and "Taille".

Nom	Modifié le	Type	Taille
default	12/01/2019 19:10	Dossier de fichiers	
documentation	12/01/2019 19:10	Dossier de fichiers	
MyRuleset	12/01/2019 18:51	Dossier de fichiers	

Et être renseigné comme suit dans Yag:



Ce répertoire doit obligatoirement contenir un fichier [ruleset.conf](#) à sa racine.



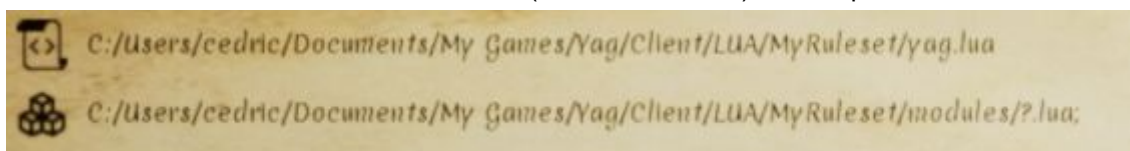
The screenshot shows a Windows Explorer window with the address bar set to "Users > cedric > Documents > My Games > Yag > Client > LUA > MyRuleset". The file list below shows three items: "modules" folder, "ruleset.conf" file, and "yag.lua" file. The columns are "Nom", "Modifié le", "Type", and "Taille".

Nom	Modifié le	Type	Taille
modules	12/01/2019 18:51	Dossier de fichiers	
ruleset.conf	12/01/2019 18:54	Fichier CONF	2 Ko
yag.lua	12/01/2019 18:51	Fichier LUA	1 Ko

Ce fichier contient 2 variables obligatoires:

- [LUA\\_FILE](#) qui contient le chemin relatif du fichier que Yag lancera pour exécuter du code LUA. Ce chemin doit être donné relativement au répertoire [<my documents>/My Games/Yag/Client/LUA/MyRuleSet](#).
- [LUA\\_PATH](#) qui contient une liste de chemins, également relatifs à [<my documents>/My Games/Yag/Client/LUA/MyRuleSet](#) et séparés par un ";". Cette variable est utilisée pour créer la variable d'environnement `LUA_PATH` (dans laquelle les chemins relatifs seront transformés par Yag en chemins absolus) qui informe LUA des chemins où il doit chercher les modules.

Le résultat de ces deux variables est affiché (et non modifiable) dans le panneau LUA:



Si vous avez installé des jeux de dés personnalisés, le fichier [ruleset.conf](#) contient également 3 variables optionnelles ([CUSTOM\\_DICE\\_01](#), [CUSTOM\\_DICE\\_02](#), [CUSTOM\\_DICE\\_03](#)) qui permettent de spécifier le nom de 3 jeux de dés.

Attention les jeux de dés spécifiés doivent être installés indépendamment du répertoire LUA. Pour plus de détails, vous pouvez regarder la vidéo sur les dés personnalisables:

<https://www.youtube.com/watch?v=baECyr8j0Qw>

Chaque fois qu'on fait appel à un mécanisme LUA dans Yag, Yag exécute le fichier `LUA_FILE` qui doit donc contenir les fonctions de l'API dédiés à la communication entre Yag et LUA (voir plus bas).

Par défaut un répertoire `default` est créé et écrasé chaque fois qu'on lance Yag.

Ce répertoire contient un fichier `ruleset.conf` ainsi qu'un fichier `yag.lua` et un module `default.lua`, qui définissent ensemble le comportement par défaut de Yag.

Ce répertoire est un exemple qui peut être dupliqué et renommé comme point de départ d'un ruleset personnalisé.

**IMPORTANT: Yag ne partage aucun mécanisme LUA entre le serveur et les clients.**

**Chaque script LUA tourne localement, chaque fonction est appelée localement.**

**Ca signifie que chaque joueur connecté doit installer le répertoire qui sera utilisé par sa copie locale de Yag.**

**En général ça revient simplement à dézipper une archive préparée par un des joueurs et renseigner son nom dans le panneau LUA de Yag.**

Pour lancer du LUA, Yag procède de la façon suivante:

- Il lance toujours le script `LUA_FILE` complet pour identifier et charger les objets LUA (fonctions...): si on met du code en dehors des fonctions, il sera exécuté
- Il lance optionnellement une fonction LUA de l'api Yag, en fonction du contexte.

Par exemple:

- Le bouton LUA dans le panneau LUA lance simplement le script et n'appelle aucune fonction. Ca peut servir à vérifier si le script et les modules sont présents et accessible, et, si le script fait quelque chose, ça peut transformer Yag en interpréteur LUA.
- Quand on clique sur le bouton LUA dans le panneau des feuilles de perso, Yag lance le script puis appelle la fonction `Yag_CharacterSheetUpdate`, qui doit être fournie par le joueur dans le script.
- Quand on clique sur un bouton de dés, Yag lance le script puis appelle la fonction `Yag_RollDiceButton` qui doit être fournie par le joueur. Si les dés sont lancés, après avoir lu les résultats, Yag les envoie au joueur via la fonction `Yag_RollDiceCallback` qui doit être fournie par le joueur.

Plus de détails sur ces 3 fonctions plus bas.

Dans le script LUA, Yag est accessible par l'objet `yag` qui contient divers objets et fonctions.

Par exemple:

- `yag.LuaDir` contient le chemin complet du répertoire `<my documents>/My Games/Yag/Client/LUA`.
- `yag.PublicMessage()` écrit un message public dans le journal

## Les structures

Les entrées et sorties des fonctions ont des structures spécifiques qui sont décrites ici.

## Les feuilles de personnages (cs)

cs = character sheet = feuille de perso en anglais

Les feuilles de perso dans LUA sont des tables avec la structure suivante:

```
{
  name = "my cs name",
  maxhp = 27,
  init = "1d20+1",
  lines =
  {
    id = { mod = true, extra = "whatever", string1 = "type", string2 = "name", formula1 = "1d20", formula2 = "1d8" },
    id = { mod = true, extra = "whatever", string1 = "type", string2 = "name", formula1 = "1d20", formula2 = "1d8" },
    id = { mod = true, extra = "whatever", string1 = "type", string2 = "name", formula1 = "1d20", formula2 = "1d8" },
    ...
  },
  ids = { 1 = "ID1", 2 = "ID2", ... }
}
```

ou plus court:

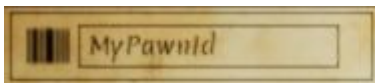
```
{ name, maxhp, init, lines { id {mod, extra, string1, string2, formula1, formula2} ... }, ids }
```

**maxhp** est un nombre (points de vie max de la feuille de perso)

**mod** est un booléen

Tous les autres champs sont des strings.

**ids** est une table contenant les identifiants des pions attachés à la feuille de perso. L'**id** d'un pion est celui qu'on trouve sur la feuille des pions avec l'icone de code-barre:



Attention à ne pas confondre les ids de lignes avec les ids de pions:

- `cs.lines["LineID"]` = la ligne (de la feuille de perso) identifiée par "LineID"
- `cs.ids[1]` = l'id du premier pion attaché à la feuille de perso

Chaque **id** (aussi une string) devrait être unique. Notez que, pour laisser une liberté complète au joueur, Yag ne s'occupe pas de l'unicité des **ids**.

Les objets (pions sur la map ou lignes au sein d'une feuille de perso) avec des ids identiques peuvent créer des problèmes (disparition d'objets ou mauvaises mises à jour).

Aussi bien pour les id de lignes que de pions, Yag génère un **id** unique par défaut avec les lettres "ID" suivies d'un nombre séquentiel.

Pour les feuilles de perso, ce nombre est réinitialisé quand la feuille ne contient aucune ligne.

Pour les pions, ce nombre est réinitialisé quand la map ne contient plus aucun pion.

Les champs correspondent à ceux situés dans la ligne cochée ici:

<input checked="" type="checkbox"/>	ability	extra	string1	string2	formula1	formula2
<input type="checkbox"/>	dex		ability	dexterity	5	
<input type="checkbox"/>	str		ability	Force	5	
<input type="checkbox"/>	int		ability	intelligence	17	
<input type="checkbox"/>	w3	bow	combat	bow of forgiveness (di.	1d20-3	1d6-3
<input type="checkbox"/>	w2	hammer	combat	hammer of compassion	1d20-3	1d10-3
<input type="checkbox"/>	w1	sword	combat	sword of love (close)	1d20-3	1d8-3

`extra` est un champ qui n'est pas utilisé par Yag, il est donné au joueur pour n'importe quelle utilisation où quelque chose doit être utilisé par LUA et stocké dans Yag: variables, conditions, listes de propriétés... C'est juste une string gratuite.

La dernière table peut contenir autant de lignes que nécessaire.

Par exemple, si la feuille de perso ci dessus est stockée dans la variable `cs`:

- `cs.lines["w3"].formula1` va retourner "1d20-3"
- `cs.lines["dex"].string1` va retourner "ability"
- `cs.lines["w2"].extra` va retourner "hammer"
- `cs.lines["str"].mod` va retourner `false`

## Les feuilles de pions (ps)

`ps` = pawn sheet = feuille de pion en anglais

Une feuille de pion est une table contenant des information sur un pion.

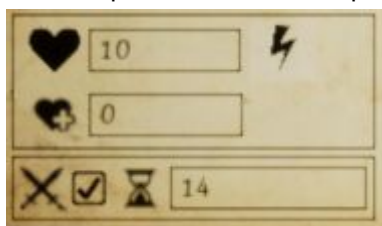
Voici la structure d'une `ps`:

```
{
  health = 10,
  initiative = 14,
  round = true
}
```

`health` et `initiative` sont des entiers.

`round` est un booléen spécifiant si le pion participe au round courant.

Ils correspondent à ces champs dans le panneau des pions:



## Les résultats de dés (dr)

`dr` = dice result = résultat de dés en anglais

Les résultats de dés contiennent beaucoup d'informations: la formule d'origine, la valeur, l'étiquette et la couleur de chaque dé lancé.

Voici la structure d'un jet complet "`3d6 + 1d4 + 7`":

```
{
  1 =
  {
    formula = "+bonus",
    color = "wood",
    result = { 1 = { value = 7, label = "7" } }
  },
  2 =
  {
    formula = "3d6",
    color = "white",
    result =
    { 2 = { value = 3, label = "three" }, 3 = { value = 6, label = "six" }, 1 = { value = 5, label = "five" } }
  },
  3 =
```

```

{
  formula = "1d4",
  color = "black",
  result =
  { 1 = { value = 4, label = "four" } }
}

```

ou plus court:

```
{ { formula, color, result { {value, label} ... } }, ... }
```

La clé pour chaque formule est un nombre

La formule est une string

La clé de chaque `line { value, label, color }` est l'ordre des dés dans chaque formule: dans "3d6", une clé "2" signifie que 3 était le deuxième dé lancé. Dans l'exemple ci dessus, dans le résultat "3d6", 5 était le premier dé, 3 le deuxième, et 6 le troisième.

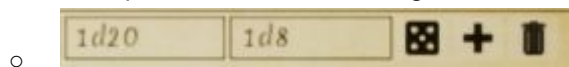
Notez que LUA ne garantit pas l'ordre de ses tables, donc ces clés sont le seul moyen d'être certain de l'ordre des dés lancés.

## Origine du jet de dés (dro)

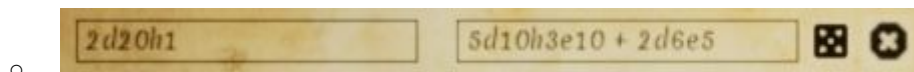
dro = dice roll origin = origine du jet de dés en anglais

**dro** (dice roll origin) est un nombre de 1 à 4 qui spécifie l'origine du jet de dés

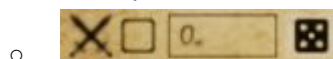
- 1 lancé en cliquant sur un bouton de feuille de perso:



- 2: lancé en cliquant sur le bouton du panneau des dés



- 3: lancé en cliquant sur le bouton d'init:



## L'objet yag (LUA vers Yag)

Depuis LUA on peut accéder à l'objet **yag** qui nous permettra d'échanger avec Yag.

Il a les propriétés suivantes.

### yag.LuaDir

- Le répertoire `<my documents>/My Games/Yag/Client/LUA`
- Utile si on distribue du contenu LUA et qu'on a besoin d'un répertoire reconnu sur tous les ordinateurs.

### yag.RollDice(f1, f2, dro, us)

- Demande à Yag de lancer les 2 formules **f1** et **f2**

- `f1` et `f2` sont des strings
- `ui` (user integer) est un nombre libre que vous pouvez utiliser comme bon vous semble: il sera passé tel quel à la fonction `Yag_RollDiceCallback`. Vous pouvez l'utiliser pour propager `dro` de `Yag_RollDiceButton` vers `Yag_RollDiceCallback`.
- `us` (user string) est une string libre que vous pouvez utiliser comme bon vous semble: elle sera passée telle quelle à la fonction `Yag_RollDiceCallback`.
- `yag.RollDice(f1, f2)` ne renvoie rien
- Après que les dés aient été lancés et les résultats lus, Yag appellera la fonction callback `Yag_RollDiceCallback`, voir plus bas pour les détails.

## `yag.ClearDice()`

- Arrête tout jet de dés en cours et efface tous les dés.
- `yag.ClearDice()` ne prend pas d'argument et ne renvoie rien

## `yag.PublicMessage(s)`

- Écrit la string `s` dans le journal.
- Tous les joueurs verront le message: peut être utilisé pour les résultats de dés
- `s` est une string
- `yag.PublicMessage(s)` ne renvoie rien

## `yag.LocalMessage(s)`

- Écrit la string `s` dans le journal.
- Seul le joueur appelant verra le message: peut être utilisé pour des logs personnelles ou des jets de dés privés.
- `s` est une string
- `yag.LocalMessage(s)` ne renvoie rien

## `yag.GetPawnInfo(id)`

- Fournit des informations sur le pion identifié par `id`
- `id` est une string qui doit être l'identifiant d'un pion
- `yag.GetPawnInfo` renvoie dans l'ordre:
  - une structure `ps` qui est la feuille de pion du pion `id`
  - une structure `cs` qui est la feuille de perso du pion `id` (peut être nil si le pion n'est attaché à aucune feuille de perso)
- ex: `MyPawnPs, MyPawnCs = yag.GetPawnInfo("MyPawnId")`

## `yag.UpdatePawn(ids, ps)`

- Met à jour tous les pions identifiés dans la table `ids` avec la structure `ps`
- `ids` est une table de strings contenant les `ids` des pions: `{ "ID1", "ID24", "MyID", ... }`
- `ps` est une structure `ps`
- `yag.UpdatePawn` ne renvoie rien

- Attention: la même structure `ps` (avec toutes ses propriétés) sera appliquée à tous les pions dans `ids`, il est donc parfois plus prudent de mettre à jours vos pions un par un en utilisant une table contenant un seul `id`:
  - `yag.UpdatePawn( { "MyID" }, MyPs )`

## Les fonctions LUA appelées par Yag (Yag vers LUA)

Quand on a besoin de demander à Yag d'effectuer une tâche LUA, comme mettre à jour une feuille de perso ou personnaliser un résultat de dé, on clique des boutons dans l'interface Yag.

Ces boutons appellent des fonctions LUA dans le script LUA du joueur.

Ces fonctions doivent être fournies par le joueur dans son script, et sont décrites ici.

### Yag\_CharacterSheetUpdate(sp, cs)

- Cette fonction est appelée quand on modifie une feuille de perso (en anglais "feuille de perso" se dit "character sheet", d'où le nom "cs").
- `sp` (selected pawns) est une table contenant les `id` (string) de tous les pions sélectionnés.
- `cs` a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).
- Cette fonction prend une `cs` en argument et renvoie une `cs`.
- Voici la forme minimale de cette fonction qui doit apparaître dans le script LUA appelé par Yag (elle ne fait rien puisque la `cs` est renvoyée non modifiée):

```
function Yag_CharacterSheetUpdate(sp, cs)
  return cs
end
```

Quand c'est demandé, Yag envoie la feuille de perso courante en argument de cette fonction.

Quand Yag reçoit la `cs` retournée par la fonction, Yag va mettre à jour la feuille de perso dans l'interface en conséquence.

### Yag\_RollDiceButton(sp, cs, id, f1, f2, dro)

- Cette fonction est appelée quand on lance des dés (en cliquant un bouton dans l'interface).
- `sp` (selected pawns) est une table contenant les `id` (string) de tous les pions sélectionnés.
- `cs` a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).
- `id` est une string contenant l'id unique de la ligne à laquelle `f1` et `f2` appartiennent.
- `f1` et `f2` sont deux strings contenant les 2 formules à lancer.
- `dro` (dice roll origin) est un nombre donnant l'origine du jet de dé (voir le chapitre des structures plus haut). Vous avez la possibilité de le propager à la fonction `Yag_RollDiceCallback` avec le paramètre `ui` de la fonction `Yag_RollDice`.
- `Yag_RollDiceButton` ne renvoie rien mais quand le jet est terminé et que les résultats ont été lus par Yag, la fonction `Yag_RollDiceCallback` est appelée par Yag pour permettre au joueur de faire ce qu'il veut avec les résultats des dés.
- Voici la forme minimale de cette fonction qui doit apparaître dans le script LUA appelé par Yag, elle ne fait rien (les dés ne seront même pas lancés):

```
function Yag_RollDiceButton(sp, cs, id, f1, f2, dro)
end
```



Notez que lorsque l'appel est fait depuis une feuille de perso, `f1` et `f2` ne sont pas nécessaires puisqu'ils peuvent être retrouvés avec `cs` et `id` comme suit:

- `f1 = cs.lines[id].formula1`
- `f2 = cs.lines[id].formula2`

Mais parfois il n'y a ni `cs` ni `id` (quand un jet est demandé directement depuis le panneau des dés par exemple), donc ils sont fournis en entrée.

Notez également que cette fonction ne fait rien par défaut: si on le souhaite, il faut explicitement lui demander de nettoyer les dés et de les lancer avec les fonctions `yag.ClearDice` and `yag.RollDice`. Il est donc possible de ne pas utiliser du tout les dés de Yag: il suffit de ne jamais appeler la fonction `yag.RollDice` et de gérer les dés nous même avec des nombres aléatoires et les règles qu'on veut.

### Yag\_RollDiceCallback(sp, cs, id, dr1, dr2, ui, us)

- C'est la fonction callback appelé après un jet de dés, automatique (appel par Yag de `Yag_RollDiceButton(sp, cs, id, f1, f2)`) ou manuel (appel par le joueur de `yag.RollDice(f1, f2)`).
- `sp` (selected pawns) est une table contenant les `id` (string) de tous les pions sélectionnés.
- `cs` a la structure d'une feuille de perso LUA (voir le chapitre des structures plus haut).
- `id` est une string contenant l'id unique de la ligne à laquelle `f1` et `f2` appartiennent.
- `dr1` et `dr2` ont la structure d'un résultat de dés ("dice result" en anglais, d'où le nom "dr") (voir le chapitre des structures plus haut)
- `dr1` et `dr2` sont les résultats respectifs de `f1` et `f2`.
- `ui` (user integer) est le paramètre `ui` passé par `yag.RollDice`.
- `us` (user string) est la string `us` passée par `yag.RollDice`.
- `Yag_RollDiceCallback(sp, cs, id, dr1, dr2)` ne renvoie rien.
- Voici la forme minimale de la fonction qui doit apparaître dans le script LUA lancé par Yag, elle ne fait rien:

```
function Yag_RollDiceCallback(sp, cs, id, dr1, dr2)
end
```