# Creating Baseball Card Templates for OOTP 11

**Written by BMW**

## Contents

Creating Baseball Card Templates for OOTP .......... 3
General Concepts .......... 3
Finding a Suitable Card Design .......... 4
Examples of 1960s cards – What to look for? .......... 4
Finding a Suitable Card Image .......... 7
Image size .......... 7
90 degree angles .......... 7
Image clarity .......... 7
Simplest form .......... 7
Altering the image .......... 8
Cutting out the picture .......... 8
Blanking out the shields .......... 9
Working on the card backs .......... 10
Creating the final template .......... 11
Where do the files go? .......... 12
Creating Layout XML .......... 13
How do we use XML? .......... 13
The required XML files .......... 13
The basics of OOTP Layout XML .......... 13
XML Declaration .......... 13
Picture tag .......... 13
Elements tag .......... 14
Element tags .......... 14
X, Y and Z coordinates .......... 14
Image Elements .......... 15
Text Elements .......... 16
Tweaking Layout XML placement .......... 19
Rule of Fernando Valenzuela .......... 19
Creating Variants .......... 20
Yearly Variants .......... 21
Token reference .......... 22
General Tokens .......... 22
League Tokens .......... 23
Team Tokens .......... 23
Person Tokens .......... 24
Statistical Tokens .......... 26
Image Tokens .......... 29

# Creating Baseball Card Templates for OOTP

OOTP 11 introduced the feature of Baseball Card templates. These templates allow our game to create baseball cards in response to certain events in the game or allow us to create our own cards manually. This gives we the opportunity to open up a new side of OOTP.

For instance, rather than look at a text log of all star selections, we can now quickly and attractively see the players that made our all star teams using a concept that is already familiar – a baseball card. We can see the team the player was on, how old he was (depending on our FaceGen settings) and his stats at the time that he was selected for the team.

This document will walk through the steps needed to create a card template – including both the conceptual and technical aspects of the process.

## General Concepts

Creating templates has six distinct phases:

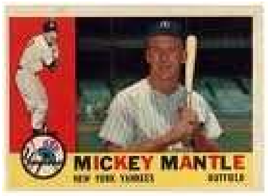
1. Finding a suitable card design, which can be skipped if creating from scratch.
2. Finding a suitable card image.
3. Altering the image so that it can be used as a template.
4. Creating the XML that will tell OOTP how to build the card.
5. Testing and tweaking the XML.
6. Creating variants (e.g. All Star, Hall of Fame).

## *Finding a Suitable Card Design*

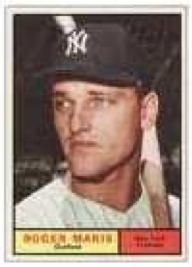
When creating a historical template, several designs can be eliminated off the bat knowing what the OOTP templates can and cannot do. Rather than go into detail about the XML and tokens at this point, we can walk through a decade of card designs and note what cards should work well with the system and which ones would be quite difficult.

Throughout the 1960s, there are 11 possible sets to make. We have included the 1961 Topps All Star variant (which isn't a set template by itself) since it happens to be drastically different from the standard 1961 design. Only seven of them are really viable candidates, and we'll step through each one by one.

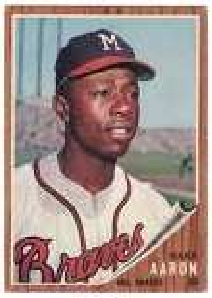
### Examples of 1960s cards – What to look for?



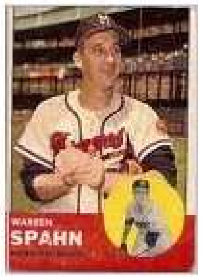
1960



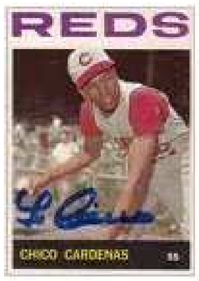
1961



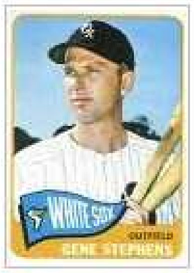
1962



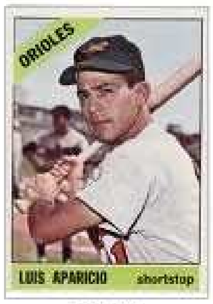
1963



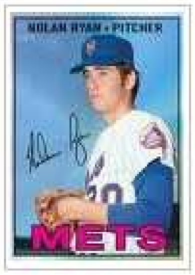
1964



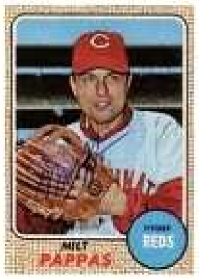
1965



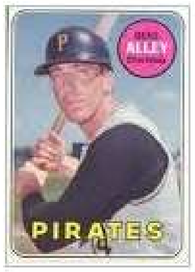
1966



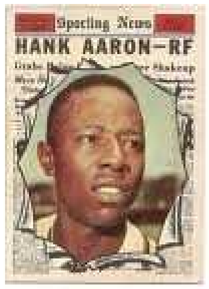
1967



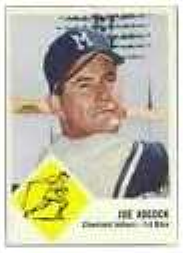
1968



1969



1961 Topps
All Star Variant
(would be included
in 1961 Topps)



1963 Fleer

**1960 Topps**

1960 Topps has two major challenges. One is the font color. The alternating blank/red letters really cannot be done for all teams (they could be done for all known major league teams, but the goal for my templates right now is to have these work with fictional teams as well).

We could easily just use black text however, and this probably does not destroy the spirit of the card. The dual photos, however, won't really work.

FaceGen, as the name alludes to, creates faces. The left hand photo is meant to be a head-to-toe image. Hence, this is not a great candidate for creating a card.

**1961 Topps**

1961 Topps has a good, very simple design that can be done with the card templates. The cards have varying colors for the text boxes at the bottom of the card that do not correspond to team colors.

However, the cards could be generated with static colored boxes or some tricks to change the colors for a few cards.

Plus, the All Star Variant is pretty cool, and very achievable with OOTP functionality.

**1962 Topps**

1962 Topps would be very easy to reproduce. Really zero complexity here.

**1963 Topps**

1963 Topps would be generally easy to reproduce, except for the second photo. It may be possible to have two photos on the card, but at best, they'd look so similar, it would likely just make for a silly looking card. Therefore, this year is probably not a great option nor a great candidate for creating a card.

**1963 Fleer**

This 1963 Fleer card set that spurred a lawsuit that gave Topps a monopoly for nearly two additional decades. This card would be very easy to reproduce.

**1964 Topps**

A 1964 Topps card would be easy to reproduce. The font at the top would probably be in a team color rather than a random color (i.e. the Reds really aren't associated with purple).

**1965 Topps**

1965 Topps is a nice looking card, but because the flag has a very odd typeface layout, it won't be easy to reproduce. we could conceivably create a more angular flag, but tapering the text as such may never be possible. Hence, this is not a great candidate for creating a card.

**1966 Topps**

The 1966 Topps card seems very simple, but the banner with the team name is at a diagonal. Currently, we cannot do this with the text. Hence, this is not a great candidate for creating a card.

**1967 Topps**

1967 Topps is a very simple design, which would be easy to do. The autograph probably wouldn't be carried over.

The 1974 Topps template does have a faux autograph, but there are two differences here. One, the autograph is on the front, not the back, making it more visible. Two, the autograph was individually placed by a Topps editor in some white-space on the photo (i.e. we wouldn't want the autograph going over the player's face).

We really won't have that kind of control. So we can do this, just lose the autograph.

The card front has text that could obscure part of the player photo. However, the text is so far at the bottom of the card and the photo takes up such a large part of the front that it is unlikely that any design would have the text encroaching on the face.

**1968 Topps**

1968 Topps would be very easy. The set does have about 4 different burlap textures and we may be able to achieve this with some tricks, but it's probably not that critical to get the feel for the card.

In fact, unless we actually had some 1968 Topps cards in our hands, we'd hardly notice that there was a difference from just Googling for images.

**1969 Topps**

1969 Topps is generally a pretty easy design. Using some tricks, we may be able to get a black border around the text. The circular shield in the upper right portion of the card could cover part of the player could be a little dangerous. But the positioning would, at worst, obscure a small portion of the ballcap and should be acceptable.

## *Finding a Suitable Card Image*

When we look for an image, we want to make sure we get one of sufficient quality. The top priorities here are:

### Image size

My preferred size is 311 pixels tall for a modern card. If we use this benchmark, we want to find a card image that is no smaller than 311 pixels. we can always find one that is large and shrink it down – if it has good clarity this will often be a good thing.

### 90 degree angles

Don’t take a card that is scanned in such a way that it’s crooked. The top and bottom of the card should look as close to a perfect horizontal line as possible. Likewise, the sides should be as close to vertical lines as possible. In general, we find a much greater loss of clarity if we need to rotate an image via an editing program than if we shrink it.

### Image clarity

We’ll want to have the best color possible and the sharpest picture possible in order to make a good template. Image sharpness is less of a concern if we be shrinking the card a great deal; blurriness is usually greatly reduced when we resize the image.

### Simplest form

If we have several choices, go for the option that has the least amount of work that we’d need to perform to get to our template. For instance, if we have a card that has an extra “Cy Young Award” banner that would be a pain to remove, take the extra time to look for another candidate.

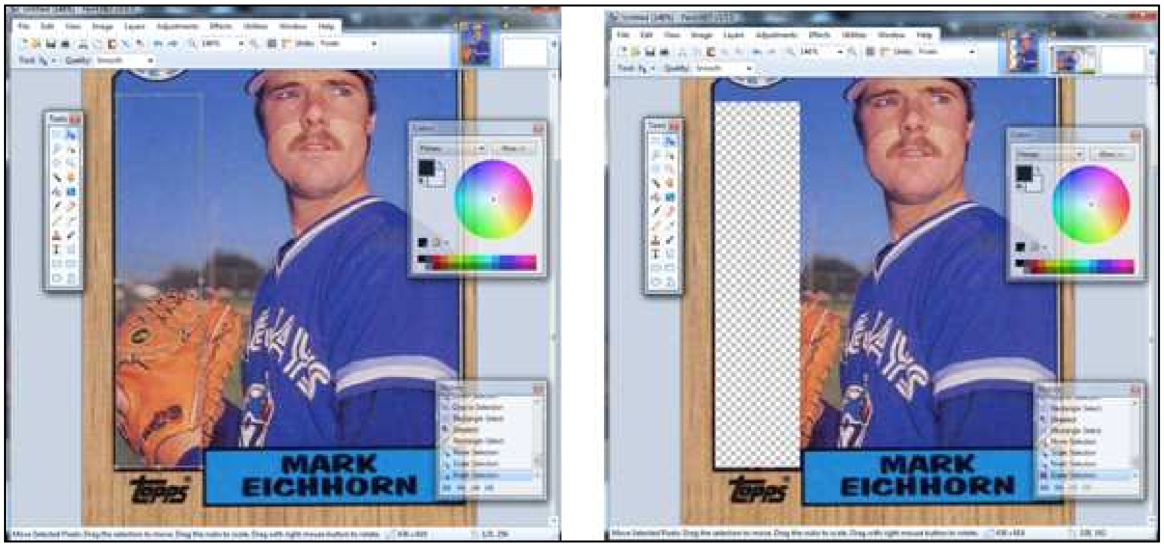
This is probably even more important for card backs. Since we be removing the stats to create a blank card back, look for a guy with less stats. It’s less work.

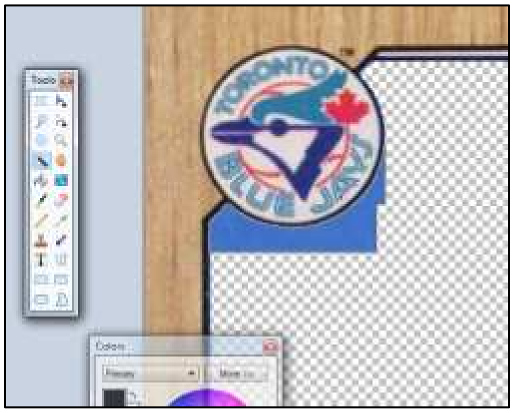
## *Altering the image*

In order to alter the image, we want to use a pretty robust image editing program. Using Paint is out, unless we are a masochist. Photoshop and Paint Shop Pro are probably both good, but they can cost quite a bit of money and would usually be overkill for the task.

There are two excellent image editing programs that are free, Paint.NET (www.getpaint.net) and The Gimp (www.gimp.org). I personally use Paint.NET for its ease of install and a keyboard/mouse interface that is simple and intuitive.

## Cutting out the picture



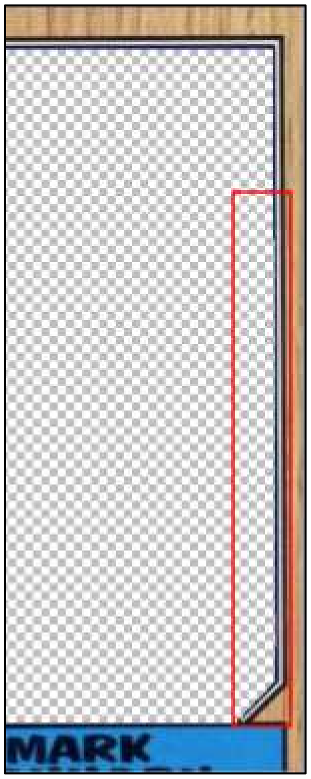


Using the selection tool of any program, draw rectangles to eat out portions of the photo as shown. our goal is basically to eat away all the portions of the picture.

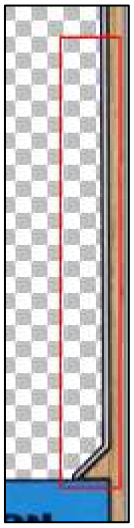
Most programs will display a checkerboard type pattern to indicate that we are viewing an empty space. This is a transparent portion of the layer that will allow a window for the OOTP player picture to show through.

In general, we want to use a lasso or magic lasso tool to handle some of the strangely shaped areas like this circular shield around the logo.

Do not be too seduced by the magic lasso; rookie users wonder how they can get the magic lasso to work better.  There is only so accurate that a tool that will automatically select an area that will ever be.



Very often, it will be more expedient to simply use the regular lasso tool or the eraser tool than to keep trying to get the magic lasso to do exactly what we want.  Don't let the size of the task fool we – unsuccessfully using the magic lasso can often  take *more* time than simply getting right to the point and manually cleaning up an image.



At this point, we can see that some of my lines are a little thin (look very carefully to see).  This can be easily fixed by simply using the line tool to create some new black lines.

**Important Note:** The black lines here aren't actually black.  No matter what kind of photo we're looking at, we almost never see true black in any kind of photo. (We will almost never see true white either, so this not is equally applicable to any white lines or areas.) Trying to make a true black line will often look jarring and stand out from the rest of the picture.

So in every case, prior to making the "black" line, find the thickest "black" line we can find on the image, zoom in, and use the eyedropper tool to grab the darkest part of that line.  we be surprised just how gray that "black" really was.

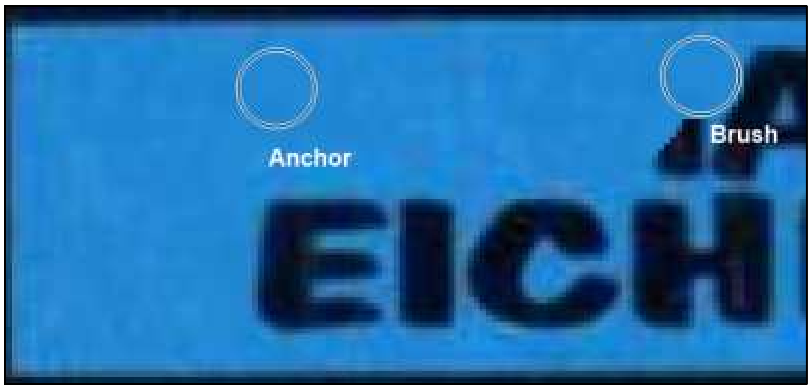
## Blanking out the shields

The shields on the card are the "backplates" for logos and text.  They can be a solid color, white or even more complex.  When we look at a 1987 Topps card, we see two shields on the front of the card.

One is the colored rectangle that holds the player name.  The second is the white circle that holds the team logo.  We'll want both of these to be blanked out so that the text and logo disappears, leaving the bare shield.

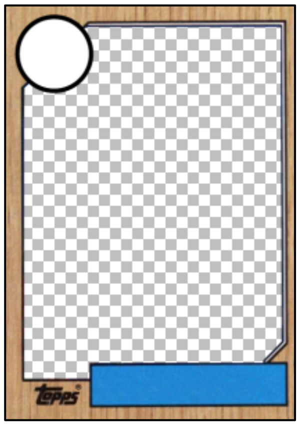
Every good paint program will have a clone tool.  This allows we to define point of the image that we want to copy (the Anchor) and will allow we to paint what is underneath that Anchor to another section of the image.

This method is superior to using just the paintbrush and a solid color, because it helps to



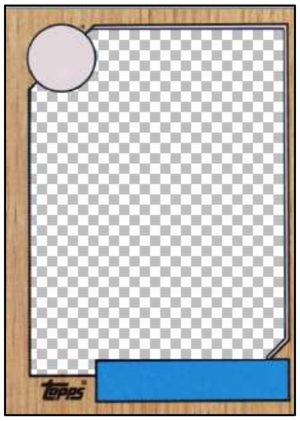
preserve the photographic nature of the card. In general, we either want a card to look 100% computer painted or 100% photographed.

We won't want 95% of the card to look like a photograph while the logo shield is so true white that it stands out like a sore thumb.



Notice here how the quickly generated white circle stands out from the rest of the photographic quality card template. It's too bright and too clean.

Note also, how the blue shield that we cleared of Mark Eichhorn's name has the same general photographic artifacts that help preserve the "suspension of disbelief."



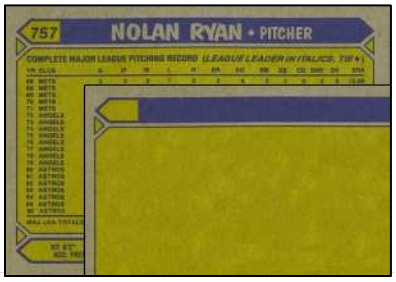
Once we apply the same clone technique to the logo shield, we will now have a basic card template front.

When we've completed the front, we will want to save the file in the native program format. For Paint.NET, this means .pdn and for Photoshop, this will be .psd.

We will want to use the native format, because it will always retain the most information should we need to go back to edit in the future.

Never, ever save it as a .jpg. This format lacks transparency capability and we will lose the blank area that we cleared from for the picture (it will be filled in with white pixels).

## Working on the card backs

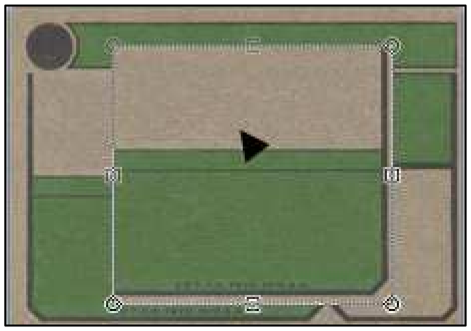


Creating the back will use the same methods. we will mainly need to use the clone tool to blank out the sections of the card.

The method is pretty straightforward. This Nolan Ryan back took under ten minutes to clear out. Also notice that the back is horizontal. This will be ideal for creating a card with a lot of statistics.

Some card backs may require to expand the stat area in order to get the amount of stats we want to have on the card.

In cases like this, we can copy and paste the stat boxes so that they extend to cover the other parts of the card back.

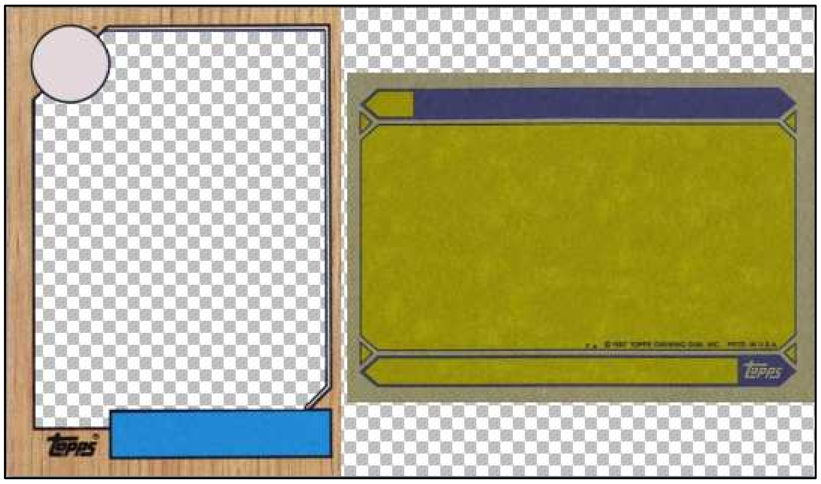


Notice here how we can copy the right hand side of the stat area and drag it to cover the trivia area, which we do not want to use at this time. We can also use this method to increase the height of the "green space" so that we will have room to fit ten seasons of data.

## Creating the final template

Once we have finished creating the front and the back, we will need to combine them both into a single template file.

In general, I recommend that the modern size card be about 311 pixels tall (and the horizontal back be about 311 pixels wide).

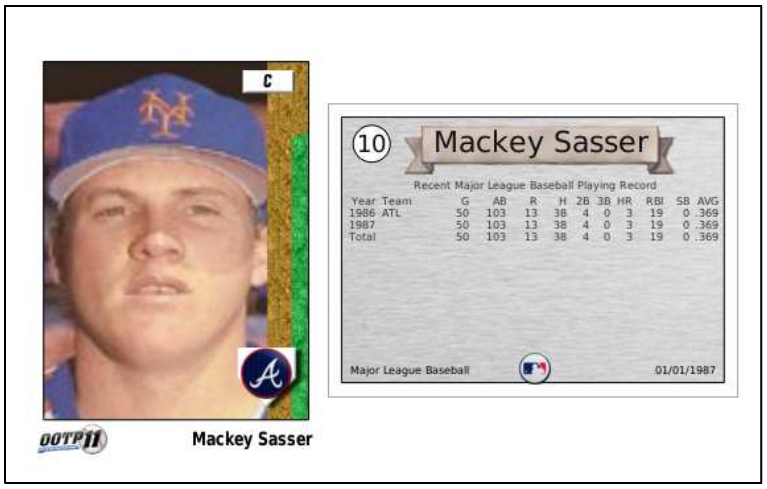


If our monitor is 1400x900, which is a common LCD monitor setting, we notice that the size of the card on the screen is roughly 2½ inches by 3½ inches – which is precisely the size of a modern baseball card.

In the case of my 1987 Topps card, the front is 221 pixels wide by 311 pixels tall. The back of the card, which has a horizontal layout, is 311 pixels wide by 221 pixels tall. When we create the final template, we will want to create a transparent canvas that is roughly 532 (221 for the front + 311 for the back) pixels wide by 311 pixels tall.

Then copy and paste the front and back onto the transparent canvas and save this as the final template.

If the card front or back has a white border, we may want to create a 1px wide grey line around the border so that the final product in OOTP can be distinguished from the program background, which will be white. Note in the example for the Mackey Sasser card that the card front looks odd while the back can be easily distinguished from OOTP's background.

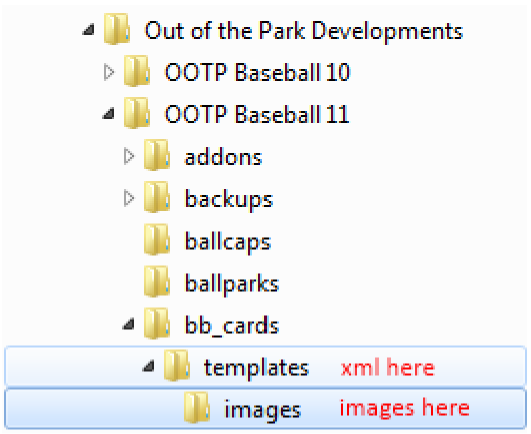


| Year | Team | G | AB | R | H | 2B | 3B | HR | RBI | SB | AVG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1986 | ATL | 50 | 103 | 13 | 38 | 4 | 0 | 3 | 19 | 0 | .369 |
| 1987 | | 50 | 103 | 13 | 38 | 4 | 0 | 3 | 19 | 0 | .369 |
| Total | | 50 | 103 | 13 | 38 | 4 | 0 | 3 | 19 | 0 | .369 |

When we are completed with creating the final template layout (make sure that we have the transparent space here, and not white/black pixels) save the template as a .png file. The .png format stands for Portable Network Graphics format. It is lossless, supports transparencies and, unlike gif files, are not patented and therefore do not require any licensing.

## Where do the files go?

The files are placed in the My Documents folder for OOTP. XML files must be placed in the *templates* folder while the images are placed in the *images* folder.



```
Out of the Park Developments
  OOTP Baseball 10
  OOTP Baseball 11
    addons
    backups
    ballcaps
    ballparks
    bb_cards
      templates    xml here
        images     images here
```

As a good practice, the image files should always include the template design name unless the image will be used for several designs.

For instance, do not name the card template *card.png*. Name it *1974_topps_ootp.png*. If there is a special component that will be used for a set, do not name the image *padres.png*, name it *1974_topps_Padres_stripe_1973.png*. This will help organize the images and allow for future enhancements where OOTP can automatically use different templates for different years.

## Creating Layout XML

### How do we use XML?

XML can be intimidating for non-programmers.  At a basic level, XML is nothing more than a method to store data so a program (like OOTP) can read it.  It can store any kind of data and it also stores the schema as well.

For instance, we can store “Closter”, “Ong’s Hat” and “Caldwell” as data, but XML will allow we to tell the reader (i.e. a program) that these are “Cities” in New Jersey via the schema.

Well, the best part about creating the Layout XML for OOTP is that we don’t really need to know how XML, in general, works.  Like most programs, OOTP has a set of predefined items that it looks for in the Layout XML.  we just need to know what OOTP is expecting.

### The required XML files

In order to make a card template, we will need a default.xml, an all_star.xml and a hof.xml.  These files will be used when manually creating a card or when the process is initiated by all star or hall of fame inductions.

### The basics of OOTP Layout XML

In a nutshell, the Layout XML is a map.  The XML will tell OOTP what we want to place (primarily text and images) and where we want to place it.  We can also potentially define other attributes, but these are the basics.

### XML Declaration

All XML documents must begin with a declaration of what type of XML it is.  This should remain constant, as this declaration is expected by OOTP.

**Example of XML Declaration tag**

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

### Picture tag

The PICTURE tag is acts as the canvas for our baseball card.  If we have images that are 1000 pixels wide but define the canvas as only 20 pixels wide, the final product will never show more than a 20 pixel wide strip of the card.

**Example of PICTURE tag**

```
<PICTURE width="537" height="311"></PICTURE>
```

Obviously, we do not want this so we should model the picture tag to have the same height and width largest part of our finished card – which will likely be the front and back of the card.

The picture tags will be the beginning and ending of our XML. All pieces of our card (known as elements) will be placed within these tags.

## Elements tag

The ELEMENTS tag is somewhat of a formality. A PICTURE has ELEMENTS and within the ELEMENTS tag are lots of little individual ELEMENT tags. We need to have only one, and it basically makes our opening XML look like this:

**Example of ELEMENTS tag**

```
<PICTURE width="537" height="311"><ELEMENTS></ELEMENTS></PICTURE>
```

## Element tags

The ELEMENT tags are placed between the single ELEMENTS tag. An Element will have a **type** attribute which is either an image or text. An Element will also have a **content** attribute which will hold the filename or the actual text, depending on the **type** of ELEMENT tag.

There is no real limit to the number of ELEMENT tags that we can place within the XML.

An ELEMENT tag can have a **filter** attribute. The filter ensure that the text or image will only be used when a card is generated by a specific event. The two events that can be used currently are *"all_star"* and *"induct_to_hof."* These filters should not be used in place of the all_star.xml and hof.xml templates.

**Example of ELEMENT tag**

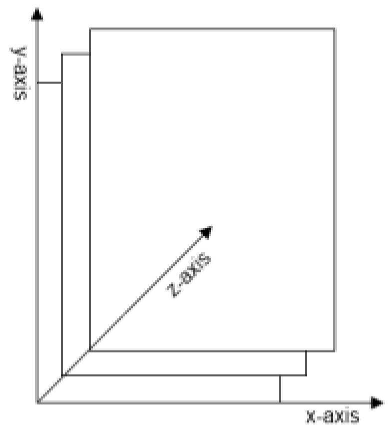
```
<ELEMENT type="image" x="164" y="226" zorder="50"
content="generic_star_all_star.png" />
```

## X, Y and Z coordinates

Each ELEMENT, regardless of the type, will include an **X**, **Y** and **Z_order** coordinate. These coordinates will tell OOTP where to place the image or text.

**Example of tag with coordinates in bold**

```
<ELEMENT type="image" x="164" y="226" zorder="50"
content="generic_star_all_star.png" />
```

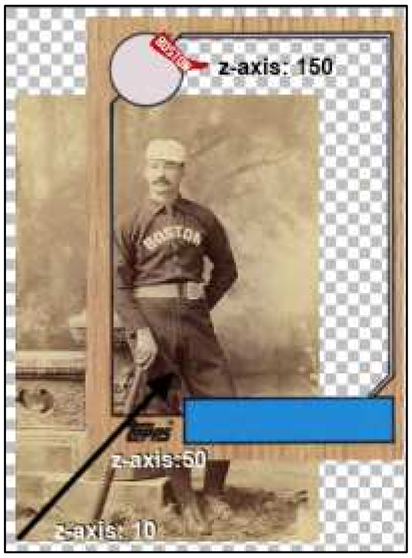


The **X** and **Y** coordinates control where the image or text will appear horizontally and vertically on the card.

The **Z** coordinate is a critical concept in making the cards. It controls what parts of our card appear on top of each other.

When a **Z** coordinate has a higher number, it will be layered closer to the top than something with a lower number.

Consider the card template with the photo of King Kelly. The card has three components that we want to use: Kelly's photo, the 1987 Topps border and the Boston Redstockings logo.



In order to layer the card properly, we'll want Kelly's photo on the bottom, the border with its transparent window should be in the middle and the logo should be on the top.

We can set the **z_order** for Kelly as 10, the border as 50 and the logo as 150.

The actual values do not matter, so long as Kelly is lowest, the logo is highest and the border is in the middle. The values could be 1, 2 and 3 or 100, 200 and 300.

As a precaution, we will want to use values with a decent about of space between them (like 10, 50, 150) in case we make any changes later to the card. The larger values give us more wiggle room to slot in new components.

## Image Elements

Image Elements are fairly straightforward. The **content** attribute will hold the image name, which can incorporate **tokens**. The images have **X, Y** and **Z_order** coordinates that will place them within the overall picture.

The Images Elements can also take **trialpha_red**, **trialpha_green** and **trialpha_blue** attributes. These attribute will take a standard HTML hex color in the syntax #rrggbb (#FFFFFF = white, #000000 = black). The three attributes can also take a **color token**. Currently, in order to change an image to a specific color (ex. Team Color), we will need to pass the same **color token** to each of the trialpha attributes.

When using the trialpha attributes, this will essentially "overwrite" the image with the solid color, so these attributes are best used to create team-colored backgrounds on cards.

In the latest version, we can force FaceGen to generate a photo at a certain **zoom** or **angle** so that it will work independently of the normal FaceGen settings as seen in the game. The choices we have are the same and have the same behavior as they do within the game. The **facegen_zoom** attribute will force the card to use one of the following zoom settings: in, normal, out or random. The **facegen_angle** attribute will force the card to use one of the following angle settings: from_left, from_right, straight or random.

**Example of Image ELEMENT tag**

```
<ELEMENT type="image" x="164" y="226" zorder="50"
content="generic_star_all_star.png" />
```

**Example of Image ELEMENT tag with trialpha attributes**

```
<ELEMENT type="image" x="0" y="0" zorder="7"
trialpha_red="[%TEAM_BG_COLOR]" trialpha_green="[%TEAM_BG_COLOR]"
trialpha_blue="[%TEAM_BG_COLOR]"
content="1974_topps_original_flag.png" />
```

**Example of Image ELEMENT tag with facegen attributes**

```
<ELEMENT type="image" x="0" y="0" zorder="7" facegen_zoom = "out"
facegen_angle = "from_left"
content="1974_topps_original_flag.png" />
```

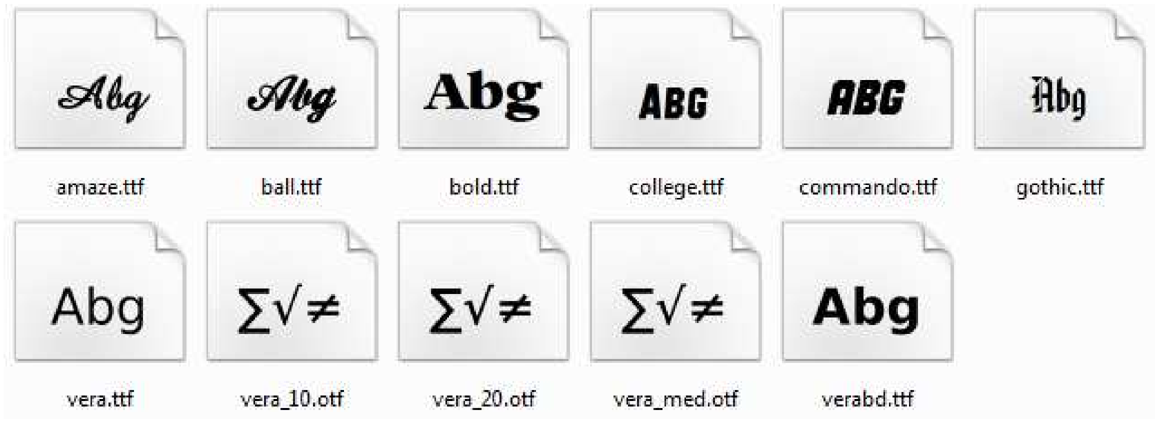
## Text Elements

Text Elements are have more default options than Image Elements. The **content** attribute will hold the actual text that is written on the card, which can incorporate **tokens**. We will need to define a **font_name**, **font_x_size**, **font_y_size** and **font_color** for a Text Element.

The **font_x_size** and **font_y_size** allow us to control the height and width of the font separately, meaning that if we want the font to be slightly fatter or thinner we can define the font as 12 pixels tall and 8 pixels wide (thinner) or 8 pixels tall and 12 pixels wide (fatter).

**Font_color** takes a standard HTML hex color in the syntax #rrggbb (#FFFFFF = white, #000000 = black). The Text Element can also take a **font_bg_color** takes a standard HTML hex color, like **font_color**. Both attributes can also take a **color token**.

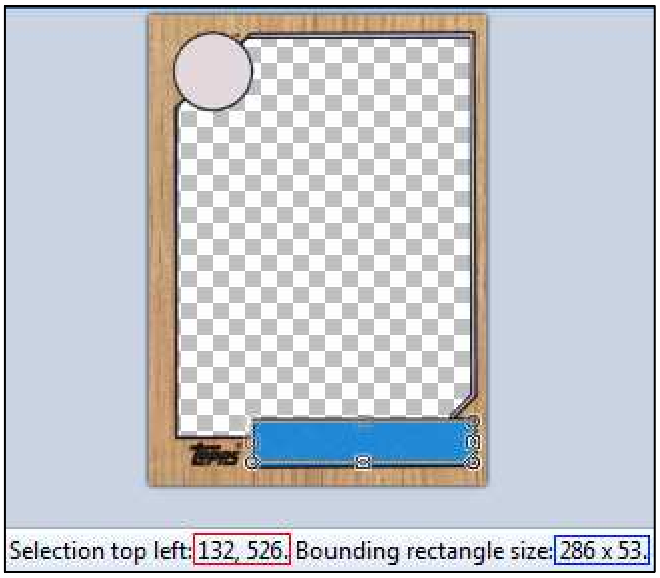
**Font_name** can currently take only fonts that are part of the OOTP game. Adding fonts to the OOTP data/fonts folder will not expand the list of available fonts. When using the **font_name** attribute, the extension (.ttf) is not required.



The Text Elements have **X, Y** and **Z_order** coordinates that will place them within the overall picture. These are accompanied by **width** and **height** attributes that, along with the **X** and **Y** coordinates, will define the text box.

A rough example of the text box can be seen in this image. The box is roughly in the place we would want to put text for the player's name.

Note that Paint.NET is invaluable since selecting an area displays all the values that we need to enter in **X**, **Y**, **width** and **height**. The numbers in the red outline above are **X** and **Y** respectively. The numbers in the blue outline are the **width** and **height** respectively.


Selection top left: 132, 526. Bounding rectangle size: 286 x 53.

The last major attributes for a Text Element tag are **align** and **valign**.

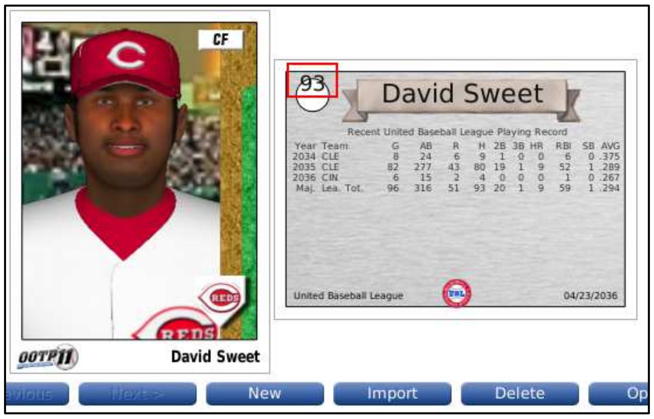
The **valign** attribute will vertically align the text within the defined text box and can take the property of *"top," "middle"* or *"bottom."* If no **valign** attribute is given, the default is *"middle."*

The **align** attribute will horizontally align the text within the defined text box and can take the property of *"left," "center," "right," "fit"* and *"shrink."* The *"fit"* property will shrink the text if it is too large to horizontally fit within the text area. However it will also allow grow vertically outside the text box, so it should be avoided. The *"shrink"* property will shrink the text if it is too large to horizontally or vertically fit within the text area. If no **align** attribute is given, the default is *"center."*

**Example of Image ELEMENT tag**

```
<ELEMENT type="text" x="247" y="62" zorder="20" width="23"
height="23" align="shrink" valign="middle" font_name="vera"
font_color="#111111" font_x_size="20" font_y_size="20"
content="[%UNIFORM_NUMBER]" />
```

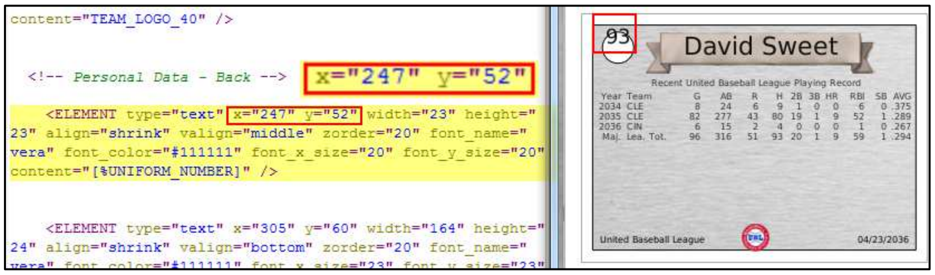
## *Tweaking Layout XML placement*



```
93          David Sweet
      Recent United Baseball League Playing Record
Year  Team          G    AB    R    H  2B  3B  HR  RBI  SB  AVG
2034  CLE           8    24    6    9   1   0   0    6   0 .375
2035  CLE          82   277   43   80  19   1   9   52   1 .289
2036  CIN           6    15    2    4   0   0   0    1   0 .267
 Maj.  Lea. Tot.   96   316   51   93  20   1   9   59   1 .294

United Baseball League                            04/23/2036
```

While Paint.NET can give us the X/Y coordinates to use for placement of our images and text boxes, we'll find that they aren't perfect. Note here that the number outlined in red isn't within the circular shield.

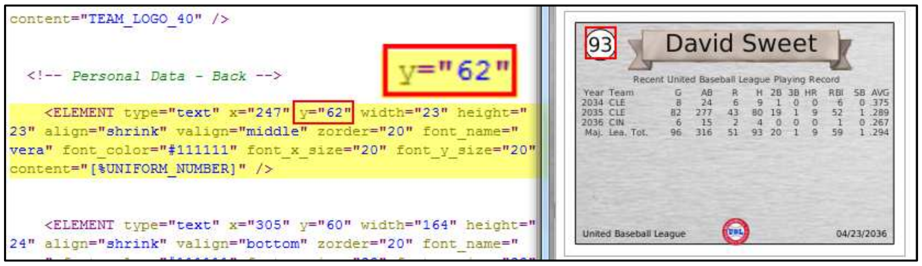
Fixing this can be done by tweaking the XML on the fly. We just need the **default.xml** file open in a text editor and OOTP 11 open at the same time. Navigate to a player and access the Baseball Card menu option and create a **New** card. We won't bother entering a title, as this will just be time consuming since we'll be creating (and replacing) dozens and dozens of cards as part of this process.



```
content="TEAM_LOGO_40" />


  <!-- Personal Data - Back -->     x="247" y="52"

    <ELEMENT type="text" x="247" y="52" width="23" height="
23" align="shrink" valign="middle" zorder="20" font_name="
vera" font_color="#111111" font_x_size="20" font_y_size="20"
content="[%UNIFORM_NUMBER]" />


    <ELEMENT type="text" x="305" y="60" width="164" height="
24" align="shrink" valign="bottom" zorder="20" font_name="
```

```
93          David Sweet
      Recent United Baseball League Playing Record
Year  Team          G    AB    R    H  2B  3B  HR  RBI  SB  AVG
2034  CLE           8    24    6    9   1   0   0    6   0 .375
2035  CLE          82   277   43   80  19   1   9   52   1 .289
2036  CIN           6    15    2    4   0   0   0    1   0 .267
 Maj.  Lea. Tot.   96   316   51   93  20   1   9   59   1 .294

United Baseball League                            04/23/2036
```

While looking at the code, we can alter the Y value since the Uniform Number is too high. Horizontally, the Uniform Number looks fine, so we can leave this alone. If we change this to 62, save **default.xml** and create a new card, we can see that this worked.



```
content="TEAM_LOGO_40" />

                                          y="62"
  <!-- Personal Data - Back -->

    <ELEMENT type="text" x="247" y="62" width="23" height="
23" align="shrink" valign="middle" zorder="20" font_name="
vera" font_color="#111111" font_x_size="20" font_y_size="20"
content="[%UNIFORM_NUMBER]" />


    <ELEMENT type="text" x="305" y="60" width="164" height="
24" align="shrink" valign="bottom" zorder="20" font_name="
```

```
93          David Sweet
      Recent United Baseball League Playing Record
Year  Team          G    AB    R    H  2B  3B  HR  RBI  SB  AVG
2034  CLE           8    24    6    9   1   0   0    6   0 .375
2035  CLE          82   277   43   80  19   1   9   52   1 .289
2036  CIN           6    15    2    4   0   0   0    1   0 .267
 Maj.  Lea. Tot.   96   316   51   93  20   1   9   59   1 .294

United Baseball League                            04/23/2036
```

This should be repeated for each element and image on the card that is not to our liking. This can be time consuming but it is a significant portion of the work required to create a good looking template.

## Rule of Fernando Valenzuela

When we test Layout XML, look for players and teams with unusually long or short names. We may not be able to prevent the text from shrinking, but we should prevent the text from spilling over the borders we've set for it.

This applies to stats as well. Test the design with a rookie as well as someone with 1,000 stolen bases or 12,000 at bats to make certain that columns won't overlap each other.

## *Creating Variants*

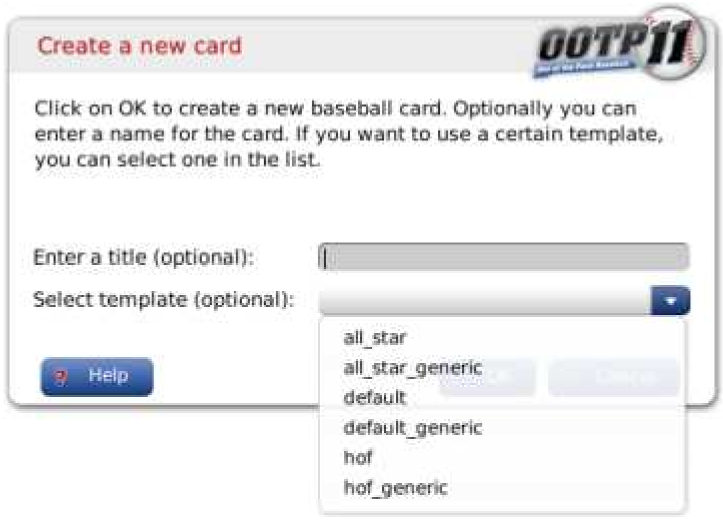
OOTP currently has several variant cards in addition to the default template:

1. Pitcher Award cards
2. Hitter Award cards
3. Rookie Award cards
4. Defense Award cards
5. Hall of Fame induction cards
6. All Star cards

These cards use the following xml files:

1. *<award_name>.xml (examples: cy_young_award.xml, mvp.xml)*
2. *pitcher_award.xml*
3. *hitter_award.xml*
4. *rookie_award.xml*
5. *defense_award.xml*
6. *hof.xml*
7. *all_star.xml*

While we have filters that will show or hide elements or images, the best method is defining a separate *allstar.xml* and *hof.xml* templates, since these are required by the game.


Create a new card

Click on OK to create a new baseball card. Optionally you can enter a name for the card. If you want to use a certain template, you can select one in the list.

Enter a title (optional):

Select template (optional):

While testing, we can force OOTP to generate an All Star, Hall of Fame card or other type of card by using the Select Template dropdown when creating a new card for a player.

This will display the templates that are currently stored in \Out of the Park Developments\OOTP Baseball 11\bb_cards\templates.

## Yearly Variants

OOTP will allow yearly variants – that is we can have multiple designs within our templates folder that will be accessed if your current sim is in a certain year or decade. Because of this, a good tip to follow when creating any graphic files is to give them names that attribute them to a specific set.

For example, if we have a file named “border.png” that is used for the 1974 Topps set, do not name it “border.png” rather name it “border_topps_1974.png” so that it will be less likely to conflict with other designs that we make or that other people create.

OOTP will use the following hierarchy when determining the template that should be used to generate a card:

1. [trigger]_[year].xml
   *example: all_star_1970.xml*
   *OOTP will use this template for all stars only when the game year is 1970.*

2. [trigger]_[decade].xml
   *example: all_star_195.xml*
   *OOTP will use this template for all stars only when the game years begin with 195x, unless a more specific template exists for an all star for a year within that decade, such as allstar_1952.xml.*

3. [trigger].xml
   *example: all_star.xml*
   *OOTP will use this template for all stars once it has determined that no templates exist for the specific game year or decade.*

4. default_[year].xml
   *example: default_1970.xml*
   *OOTP will use this template only when the game year is 1970, provided that the card has not been generated using a trigger.*

5. default_[decade].xml
   *example: default_195.xml*
   *OOTP will use this template only when the game years begin with 195x, unless a more specific template exists for a year within that decade, such as default_1952.xml, provided that the card has not been generated using a trigger.*

6. default.xml
   *example: default.xml*
   *OOTP will use this template once it has determined that no templates exist for the specific game year or decade, provided that the card has not been generated using a trigger.*

## Token reference

Tokens represent the variable text that can be displayed on a card. If a token is not valid for the card (pitcher stats on a hitter card or hitter stats on a pitcher card), the entire **Text Element** will be hidden.

### *General Tokens*

**\n**

Creates a line break (carriage return). If there are two consecutive line breaks (example: "\n\n"), this will be rendered as a single line break. A space between the line breaks (example: "\n \n") will cause two line breaks to be generated.

If a token generates a blank value (example: [%SEASON9_W] for a rookie pitcher), this will not generate a space, allowing "\n[%SEASON9_W]\n" to be represented as a single line break. This is an important concept as it allows the creation of stat columns that will shrink and expand as needed.

**[%DATE]**

Game date, in mm/dd/yyyy format.
*Example: 01/01/1970*

**[%YEAR]**

Game date year, in yyyy format.
*Example: 1970*

**[%YEAR2]**

Game date year, in yy format.
*Example: 70*

**[%RANDOM_NUMBER]**

Returns 0 to 9. Using multiple times in a row will allow the generation of a two or three (or more) digit number.

Be aware that using this method will generate leading zeros. A Zip Code is an example of a number with leading zeros: MANATI, PR has a Zip Code of 00674. Using "674" as the Zip Code would not be understood by the postal service. In the same fashion, if we had the random number generation look up a file called "00674.jpg" OOTP will not find the file if the file was named "674.jpg."
*Example: 5*

## League Tokens

**[%LEAGUE_NAME]**
Full name of league.
*Example: Federal League*

**[%LEAGUE_ABBR]**
Abbreviation of league.
*Example: FL*

**[%LEAGUE_TEXT_COLOR]**
HTML hex color code for league's text color.
*Example: #000000*

**[%LEAGUE_BG_COLOR]**
HTML hex color code for league's text background color.
*Example: #ffffff*

**[%SUB_LEAGUE_NAME]**
Full name of sub league.
*Example: American League*

**[%SUB_LEAGUE_ABBR]**
Abbreviation of sub league.
*Example: AL*

## Team Tokens

**[%TEAM_ID]**
Numeric ID of team within OOTP.
*Example: 12*

**[%TEAM_NAME]**
City name of team.
*Example: New York*

**[%TEAM_ABBR]**
Abbreviation of team.
*Example: NY*

**[%TEAM_NICK]**
Nickname of team.
*Example: Highlanders*

**[%TEAM_NAME_NICK]**

City and nickname of team.
*Example: New York Highlanders*

**[%TEAM_TEXT_COLOR]**
HTML hex color code for team's text color.
*Example: #000000*

**[%TEAM_BG_COLOR]**
HTML hex color code for team's text background color.
*Example: #ffffff*

**[%TEAM_JERSEY_MAIN_COLOR]**
HTML hex color code for team's main jersey color.
*Example: #000000*

**[%TEAM_JERSEY_SECONDARY_COLOR]**
HTML hex color code for team's secondary jersey color.
*Example: #000000*

**[%TEAM_JERSEY_PIN_STRIPES_COLOR]**
HTML hex color code for team's jersey pin stripe color.
*Example: #000000*

**[%TEAM_BALLCAPS_MAIN_COLOR]**
HTML hex color code for team's main cap color.
*Example: #000000*

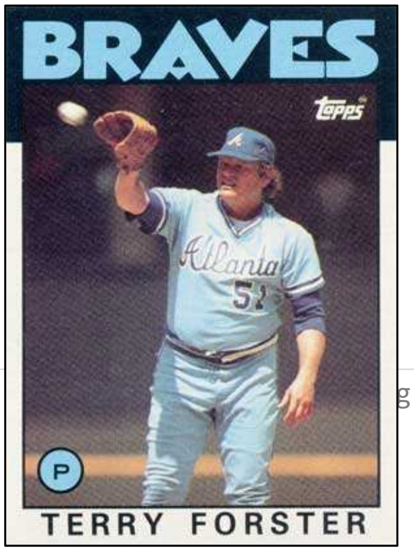
**[%TEAM_BALLCAPS_VISOR_COLOR]**
HTML hex color code for team's cap visor color.
*Example: #000000*

## *Person Tokens*

**[%PERSON_FIRST_NAME]**
First Name of person.
*Example: Terry*

**[%PERSON_LAST_NAME]**
Last Name of person.
*Example: Forster*

**[%PERSON_NICK_NAME]**
Nickname of person.
*Example: Fat Tub of Goo*



**[%PERSON_NAME]**
Full name of person.
*Example: Terry Forster*

**[%PERSON_NAME_NICK]**
Full name of person, including nickname.
*Example: Terry Fat Tub of Goo Forster*

**[%POSITION]**
Full position of player.
*Examples: Pitcher, Catcher, First Base, Second Base, Third Base, Shortstop, Left Field, Center Field, Right Field, Designated Hitter*

**[%POSITION_SHORT]**
Abbreviated position of player.
*Examples: P ,C, 1B, 2B, 3B, SS, LF, CF, RF, DH*

**[%UNIFORM_NUMBER]**
Uniform number of player.
*Example: 51*

**[%AGE]**
Current age of player.
*Example: 34*

**[%PERSON_THROWS]**
Throwing handedness in lowercase.
*Example: left, right*

**[%PERSON_BATS]**
Batting handedness in lowercase.
*Example: left, right, switch*

**[%PERSON_THROWS_U]**
Throwing handedness in proper case.
*Example: Left, Right*

**[%PERSON_BATS_U]**
Batting handedness in proper case.
*Example: Left, Right, Switch*

**[%PERSON_WEIGHT]**
Weight of player.

*Example: 200 lbs, 95 kg*

**[%PERSON_HEIGHT]**
Height of player.
*Example: 6'0", 72 inches*

**[%PERSON_CITY_OF_BIRTH]**
Player's city of birth.
*Example: Fort Gaines*

**[%PERSON_DOB]**
Player's date of birth in mm/dd/yyyy format.
*Example: 01/01/1970*

**[%PERSON_ID]**
Player's OOTP ID.
*Example: 13423*

## Statistical Tokens

Each season statistical token can be used with [%SEASON_ or [%SEASON***x***_ where ***x*** is the number of seasons prior to the current season. The statistics can display up to nine seasons prior to the current season. The statistical tokens will display only the level (ML, AAA, etc.) that the player is in at the time the card is generated. [%CAREER_ will display the career total for the statistic at the player's current level.

**[%SEASON_YEAR], [%SEASON1_YEAR] ... [%SEASON29_YEAR]**
Statistical year in yyyy format.
*Example: 1974*

**[%SEASON_YEAR2], [%SEASON1_YEAR2] ... [%SEASON29_YEAR2]**
Statistical year in yy format.
*Example: 74*

**[%SEASON_TEAM], [%SEASON1_TEAM] ... [%SEASON29_TEAM]**
Abbreviation of team, will list multiple teams, separated by commas.
*Example: OAK, NYA*

**[%SEASON_G], [%SEASON1_G] ... [%SEASON29_G], [%CAREER_G]**
Games played by batter.
*Example: 34*

**[%SEASON_AB], [%SEASON1_AB] ... [%SEASON29_AB], [%CAREER_AB]**
At bats by batter.

*Example: 34*

**[%SEASON_H], [%SEASON1_H] ... [%SEASON29_H], [%CAREER_H]**
Hits by batter.
*Example: 34*

**[%SEASON_2B], [%SEASON1_2B] ... [%SEASON29_2B], [%CAREER_2B]**
Doubles by batter.
*Example: 34*

**[%SEASON_3B], [%SEASON1_3B] ... [%SEASON29_3B], [%CAREER_3B]**
Triples by batter.
*Example: 34*

**[%SEASON_HR], [%SEASON1_HR] ... [%SEASON29_HR], [%CAREER_HR]**
Home runs by batter.
*Example: 34*

**[%SEASON_R], [%SEASON1_R] ... [%SEASON29_R], [%CAREER_R]**
Runs scored by batter.
*Example: 34*

**[%SEASON_RBI], [%SEASON1_RBI] ... [%SEASON29_RBI], [%CAREER_RBI]**
Runs batted in by batter.
*Example: 34*

**[%SEASON_SB], [%SEASON1_SB] ... [%SEASON29_SB], [%CAREER_SB]**
Stolen bases by batter.
*Example: 34*

**[%SEASON_HITTER_BB], [%SEASON1_HITTER_BB] ... [%SEASON29_HITTER_BB], [%CAREER_HITTER_BB]**
Walks by batter.
*Example: 34*

**[%SEASON_HITTER_K], [%SEASON1_HITTER_K] ... [%SEASON29_HITTER_K], [%CAREER_HITTER_K]**
Strikeouts by batter.
*Example: 34*

**[%SEASON_AVG], [%SEASON1_AVG] ... [%SEASON29_AVG], [%CAREER_AVG]**
Batting average by batter.
*Example: .266*

**[%SEASON_OBA], [%SEASON1_OBA] ... [%SEASON29_OBA], [%CAREER_OBA]**
On base average by batter.
*Example: .326*

**[%SEASON_SLG], [%SEASON1_SLG] ... [%SEASON29_SLG], [%CAREER_SLG]**
Slugging percentage by batter.
*Example: .367*

**[%SEASON_OPS], [%SEASON1_OPS] ... [%SEASON29_OPS], [%CAREER_OPS]**
On base plus slugging by batter.
*Example: .689*

**[%SEASON_E], [%SEASON1_E] ... [%SEASON29_E], [%CAREER_E]**
Errors by player.
*Example: 34*

**[%SEASON_GP], [%SEASON1_GP] ... [%SEASON29_GP], [%CAREER_GP]**
Games pitched by pitcher.
*Example: 34*

**[%SEASON_GS], [%SEASON1_GS] ... [%SEASON29_GS], [%CAREER_GS]**
Games started by pitcher.
*Example: 34*

**[%SEASON_W], [%SEASON1_W] ... [%SEASON29_W], [%CAREER_W]**
Wins by pitcher.
*Example: 34*

**[%SEASON_L], [%SEASON1_L] ... [%SEASON29_L], [%CAREER_L]**
Losses by pitcher.
*Example: 34*

**[%SEASON_SV], [%SEASON1_SV] ... [%SEASON29_SV], [%CAREER_SV]**
Saves by pitcher.
*Example: 34*

**[%SEASON_IP], [%SEASON1_IP] ... [%SEASON29_IP], [%CAREER_IP]**
Innings pitched by pitcher.
*Example: 72.1*

**[%SEASON_HA], [%SEASON1_HA] ... [%SEASON29_HA], [%CAREER_HA]**
Hits allowed by pitcher.
*Example: 34*

**[%SEASON_HRA] [%SEASON1_HRA] ... [%SEASON29_HRA], [%CAREER_HRA]**
Home runs allowed by pitcher.
*Example: 34*

**[%SEASON_BB], [%SEASON1_BB] ... [%SEASON29_BB], [%CAREER_BB]**
Walks allowed by pitcher.
*Example: 34*

**[%SEASON_K], [%SEASON1_K] ... [%SEASON29_K], [%CAREER_K]**
Strikeouts by pitcher.
*Example: 34*
**[%SEASON_ERA], [%SEASON1_ERA] ... [%SEASON29_ERA], [%CAREER_ERA]**
Earned run average by pitcher.
*Example: 3.47*

**[%SEASON_WHIP], [%SEASON1_WHIP]…[%SEASON29_WHIP], [%CAREER_WHIP]**
Walks + hits per inning pitched by pitcher.
*Example: 1.22*

## *Image Tokens*

Image tokens can be used in place of a filename in the **content** attribute of an **Image Element.** OOTP will dynamically display the team or league logos as the image rather than an external file.

**PICTURE**
FaceGen or photo of player.

**TEAM_LOGO**
Team logo 150 pixels x 150 pixels.

**TEAM_LOGO_*n***
Team logo ***n*** pixels x ***n*** pixels. ***n*** must be one of the following numbers: 25, 40, 50, 60, 90 or 110.

**JERSEY_LOGO**
Team jersey logo in native sze.

**CAP_LOGO**
Team cap logo in native sze.

**LEAGUE_LOGO**
League logo 150 pixels x 150 pixels.

**LEAGUE_LOGO_n (see TEAM_LOGO_n)**

League logo ***n*** pixels x ***n*** pixels. ***n*** must be one of the following numbers: 25, 40, 50, 60, 90 or 110.