The Good vs bad Description

During Procedures, Changes, Documentation, SOPs and similar stuff (from now on refered to as "**description**"), I use a lot of descriptions that need to be followed by users, developers or sysadmins (from now on refered to as "**implementor**").

The one thing all there descriptions should have it common is:

- Giving the person the needed tools/guides to get from A to B
- Being clear and straight in all parts
- Not being aubigously, as this will lead to doubt/wrong implementation/ - hence errors

A lot of people write Procedures, Changes, Documentation, SOPs from either

- Their own perspective and knowledge level
- An expection of the implementor knowledge level

which is **the most common mistanke to make - relative to the target audience**.

As soon as the implementor reads the description, the path to misunderstading and doubt starts, typically because of a lower knowledge level, missing informations and faulte expectations from the authors side.

As always, there is a **target audience** for the description and the writer can definitely have an expectation towards the implementers skills and knowlegde level.

For Sysadmins writing SOPs to Sysadmins, You must expect that the Sysadmin to carry out the SOP, knows how to logon to a server (but do remember to write the FDQN of the server 🙂).

But for Sysadmins writing guides or manualt to end users or superusers, this cant be expected, and the logon procedure has to be explained.

Its better to pin out the details that at first looks stupid or tedious - than making the mistake of not doing it - its virtually impossible to do it to well (respecting and thinking of the target audience)

Wrong way

Log on to the database server and sudo to root

Right Way

Log on to the database server

```
ssh db01.mydomain.dk
```

After login, get root rights:

```
sudo -s
```

Comments

Wrong way

Run the sql file update.sql on the Market database

Right Way

Run the attached sql file update.sql on the Market database:

```
mysql -u marketuser -p market < update.sql
```

This must not give any errors, output should be like:

Comments

Output should be pasted into the Change

```
5 rows updated sucessfully
```

Wrong way

Deploy version 1.4.2 of the Marketplace software on the App servers and save the output

Check the output log for errors

Right Way

Login to each App server (app01.market.mydomain.dk and app02.market.mydomain.dk) and do

```
sudo -s
deploy.sh --version 1.4.2 --
summarize > /tmp/deploy.txt
```

examine if any errors occoured:

```
cat /tmp/deploy.txt | grep error
```

Comments

The **Wrong way** assumes the the implementor knows that he need to deploy version 1.4.0 and it does not state that "–summerize" gives the needed output or how to check for errors

Output should be pasted into the Change

Wrong way

Rollback to the previous version with the deploy script

Right Way

Login to each App server (app01.market.mydomain.dk and app02.market.mydomain.dk) and do

```
sudo -s
deploy.sh --version 1.4.0 --
reverse --summarize > /tmp
/deploy.txt
```

examine if any errors occoured:

```
cat /tmp/deploy.txt | grep error
```

Comments

The **Wrong way** assumes the the implementor knows that the previous version was 1.4.0 and it does not state anything more that the deploy scripts should be used.

This can be acceptable in a very controlled (puppet, chef etc) environment.

The Right does take it a step further because of:

In this case, the deployscript needs the "–revese" to rollback/downgrade as a fault precaution

The rollback can be done by pasting the exact command into a terminal

Output should be pasted into the Change

Wrong way

Take a backup of the config.properties file

Restore the backup of the config.properties file

Right Way

Take a backup of the config.properties file

```
cp /opt/tomcat/config.
properties /backup/config.
properties-Change7000
```

Restore the previous config.properties file

```
cp /backup/config.properties-
Change7000 /opt/tomcat/config.
properties
```

Comments

The **Wrong Way** states no path or filename for the backup, leaving it up to the implementor. If a need for rollback arises later, a problem can be actualling finding the backup that the (possible another) implementor did.

In the **Right Way**, the filename and path are stated, and the command part can be pasted into a terminal

Partly same as above