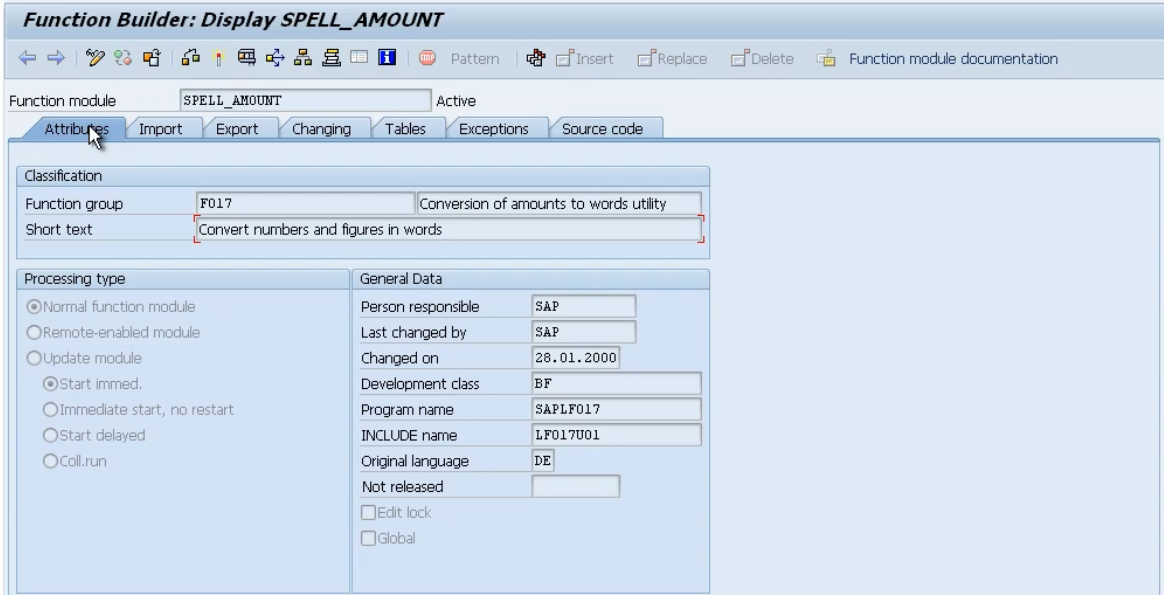
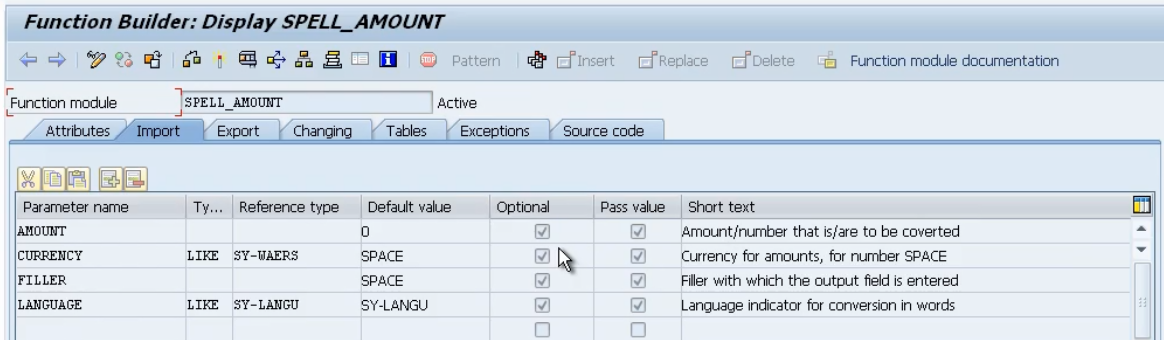


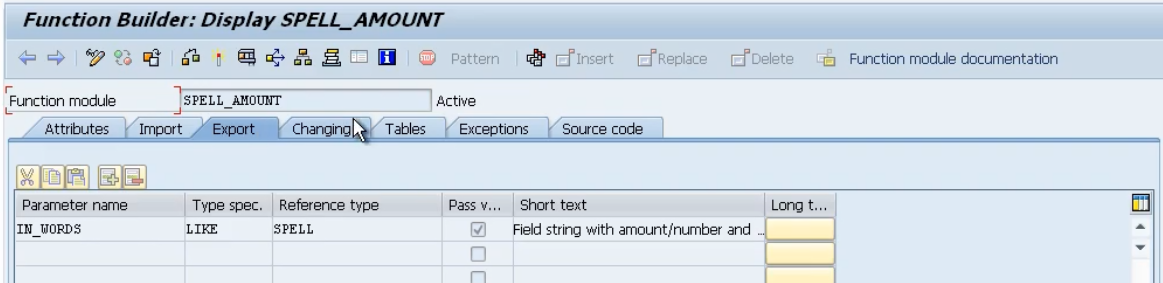
And all we need to do to have a look to see if these are what we want, is double click and we will get to see the code on the right-hand side. And just like any normal program, we can test out the function module.

first, we have the function module attributes. Here you can see we have the function group that this function module belongs to, and we have some text to describe the function module and group. And we have various options further down. Processing type defines the actual type of function module. This one is defined as normal, but we can have remote enabled which means it can get triggered from an external system. And then the update module, you can see. This is set to start immediate, and we have some various other options, such as start delayed. And then just basic information about when the function module was created.

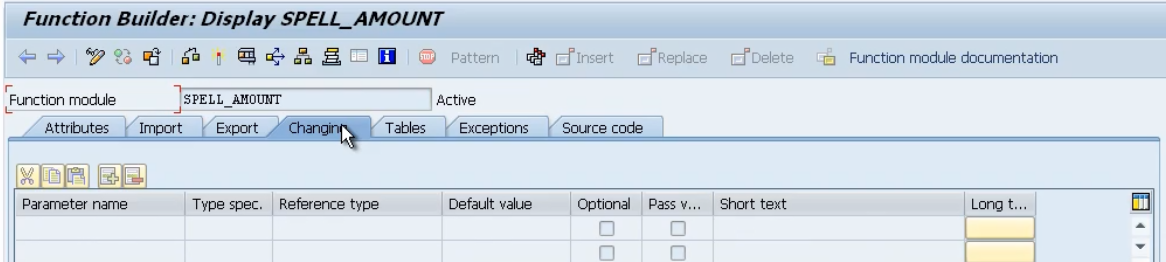


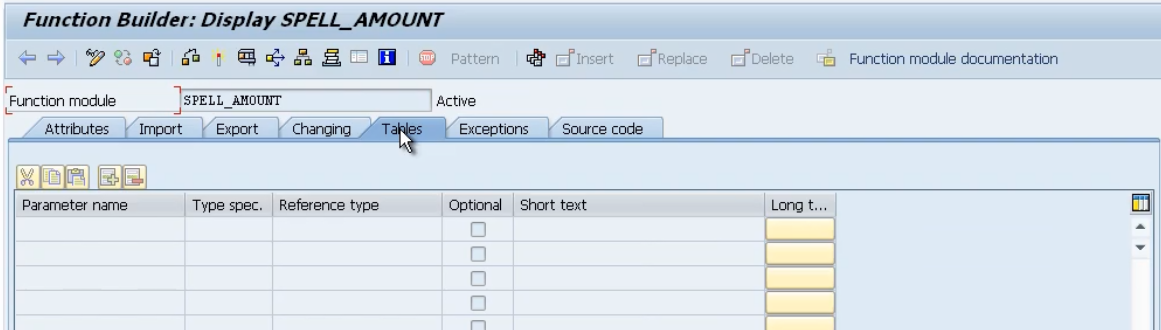
Let's move on to the import tab. Now, this lists the individual fields we will use for our data interface for data we pass into the function module. So, if we look at this from a function module perspective, it's going to be importing information that it can use during the processing of its code.

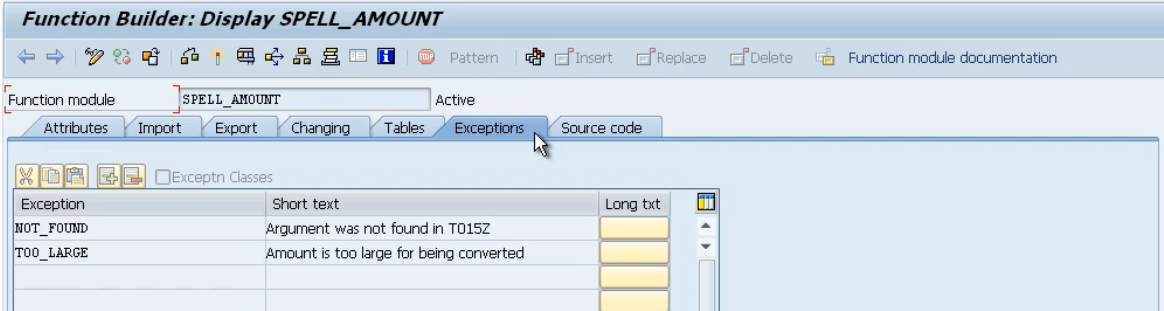
And in this case, we got four fields, an amount, a currency, a filler, and a language. And we have some text on the right-hand side to give some information about what the actual field is. And we have a couple of check boxes. The first one is optional. That means with this function module, all the fields that get imported, they're all optional fields. We don't have to fill them in. When we call this function module from our program, often, you will get some mandatory fields. But this one, every field is optional. And then we have pass by value. Which means it's going to pass the actual value into the function module.



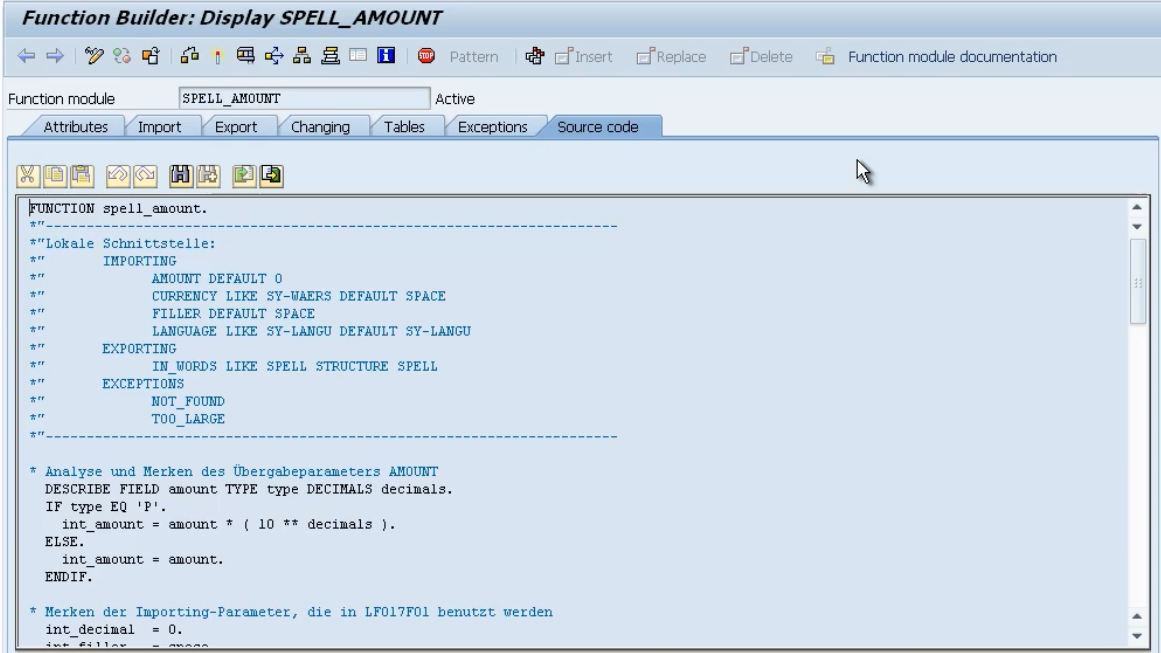
Let's move onto the export tab. Now once the function module does its work, the fields defined in the export tab are what get sent back to our calling program. So again, from a function module perspective, it exports this information out to whoever called it.

Then we have a changing tab. And as you might guess, this just lists fields that would be changed by the function module. That is, we can identify fields that we send into the function module, the function module may then change the values, and these fields are then returned to our calling program. So, where the fields on the import tab, the functional module only receives in. The export, the functional module only sends them back out to whoever called. The changing, it receives data in, changes it and then sends it back out. So, it's a combination of the two, really.

Then we have tables. And just like when we looked at subroutines, we're not restricted to just passing in individual fields. We can pass in internal tables as well. And this is where internal tables would be set up.



Now the exceptions tab. Every function module can pass back exception information to indicate if the function module was executed successfully or not. And we're not just limited to finding out if it was successful. But we can also determine specific error messages. So, we could have a whole list of exceptions here. And as you can see with this function module, we have not NOT\_FOUND, we have TOO\_LARGE, and we could have a whole host of other messages that get passed back to our program, to help us determine why the function module didn't execute as we wanted it to. These exceptions are not fixed. Every function module defines its own exceptions. It's just like passing messages back to the calling program just to indicate if worked, or not. And, if it didn't, why didn't it work? So that the calling program can then use its own logic to determine what it should do next.



And the last tab is the source code itself.