The following change to EditField::Insert(int, cost WString&) will fix the problem. I believe this should be regarded as a library bug and should be fixed to honor EditField's promise to work with customized Convert.

```
int EditField::Insert(int pos, const WString& itext)
{

 // omitted many unchanged lines
 //
 // the following test might need to be rewritten
 // taking the Format/Scan effect into account
 //
 if(ins.GetCount() + text.GetCount() > maxlen) {
  BeepExclamation();
  return 0;
 }

 //changes =====================
 int len=text.GetLength();
 text.Insert(pos, ins);
 SetData(GetData()); // might be smarter way to achieve it
 len-=text.GetLength();
 Update();
 return -len;
}
```

After insert action, the text the EditField held in its internal storage becomes different from what it displays (because of the customized Format). This should not happen. Above code did the synchronization. I feel that it's not the best place to do such synchronization. Mirek will certainly be able to find the best and most economic way to achieve such synchronization, but one is indeed needed.