Application: gvSIG desktop - gvSIG bugs #4290
Can't autoidentify EPSG (WKT without the AUTHORITY tag)

07/14/2016 05:28 PM - Antonio Falciano

Status:	Closed	**% Done:**	0%
Priority:	Normal	**Spent time:**	0.00 hour
Assignee:	Joaquín del Cerro Murciano		
Category:	CRS		
Target version:	2.3.0-2445 (rev. org.gvsig.desktop-2.0.152)		
Severity:	Minor	**Add-on version:**	
gvSIG version:	2.3.0	**Add-on build:**	
gvSIG build:	2436	**Add-on resolve version:**	
Operative System:		**Add-on resolve build:**	
Keywords:	authority,prj,wkt,wkt_esri	**Proyecto:**	
Has patch:		**Hito:**	
Add-on name:	Unknown		

Description

Some projection files without the AUTHORITY tag can't be "AutoIdentifyEPSG"ed at the moment.
Some examples:

- EPSG:23032 (gdalsrsinfo EPSG:23032 -o wkt_esri --> ESRI WKT)

PROJCS["ED50_UTM_zone_32N",GEOGCS["GCS_European_1950",DATUM["D_European_1950",SPHEROID["International_1924",6378388,297]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",9],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["Meter",1]]

- EPSG:23032 (another ESRI WKT)

PROJCS["ED_1950_UTM_Zone_32N",GEOGCS["GCS_European_1950",DATUM["D_European_1950",SPHEROID["International_1924",6378388.0,297.0]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500000.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",9.0],PARAMETER["Scale_Factor",0.9996],PARAMETER["Latitude_Of_Origin",0.0],UNIT["Meter",1.0]]

- EPSG:3004 (ESRI WKT)

PROJCS["Monte Mario / Italy zone 2",GEOGCS["Monte Mario",DATUM["D_Monte_Mario",SPHEROID["International_1924",6378388,297]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",15],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",2520000],PARAMETER["false_northing",0],UNIT["Meter",1]]

- EPSG:3857 (ESRI WKT see #3954)

PROJCS["WGS_84_Pseudo_Mercator",GEOGCS["GCS_WGS_1984",DATUM["D_World Geodetic System 1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Popular Visualisation Pseudo Mercator"],PARAMETER["semi_minor",6378137.0],PARAMETER["latitude_of_origin",0.0],PARAMETER["central_meridian",0.0],PARAMETER["scale_factor",1.0],PARAMETER["false_easting",0.0],PARAMETER["false_northing",0.0],UNIT["m",1.0]]

Related to #3954

Associated revisions

Revision 596 - 09/14/2016 05:11 PM - Joaquín del Cerro Murciano

refs #4290, patch gracias a Antonio Falciano

History

#1 - 07/14/2016 05:50 PM - Antonio Falciano

- File gvSIG-desktop-2.3.0-wkt2epsg-1.0.0-0-testing-all-all-j1_7.gvspkg added

A quick and dirty solution to the issue described above could be to search the candidate SRs in the EPSG Registry that matches with the source WKT in some way. More in detail, if the imported WKT can't be autoidentified as an EPSG code, if possible, we can search all SRs whose PROJ4 string matches to the original one (skipping the +towgs84 parameters) and then classify all the candidate SRs in reference to SR name similarity e.g using the Levenshtein distance, otherwise classify all the SRs in reference to name similarity only. So the best match/bet will be the EPSG code with minimum Levenshtein distance!
I attach a Jython script to show how all this works. Simply uncomment the wkt variable to test in main().

```
# encoding: utf-8

from es.idr.teledeteccion.connection import EpsgConnection, Query
from org.gdal import osr
from org.apache.commons.lang3 import StringUtils

from prettytable import PrettyTable

def getValues(sr):
    if sr.IsGeographic():
        kind = 'geographic 2D'
        attr = 'GEOGCS'
    elif sr.IsProjected():
        kind = 'projected'
        attr = 'PROJCS'
    return kind, attr

def searchCandidates(source_sr, bProj4 = True):
    '''Search for candidate SRs in the EPSG Registry'''

    kind, attr = getValues(source_sr)
    source_sr_name = source_sr.GetAttrValue(attr)

    # Connect with the EPSG Registry
    conn = EpsgConnection()
    conn.setConnectionEPSG()

    sql = '''SELECT DISTINCT coord_ref_sys_code FROM epsg_coordinatereferencesystem
        WHERE coord_ref_sys_kind = '%s' AND deprecated = 0''' % kind

    result = Query.select(sql, conn.getConnection())

    t = PrettyTable(["EPSG code", "SRS name", "Levenshtein Distance"])
    t.align["EPSG code"] = "r"
```

```
    t.align["SRS name"] = "l"
    t.align["Levenshtein Distance"] = "c"

    while result.next():
        crs_code = result.getInt("coord_ref_sys_code")
        try:
            target_sr = osr.SpatialReference()
            target_sr.ImportFromEPSG(crs_code)

            if bProj4 == True:
                source_proj4 = source_sr.ExportToProj4()
                target_sr.MorphToESRI() # skip the +towgs84 parameters
                target_proj4 = target_sr.ExportToProj4()
                if source_proj4 == target_proj4:
                    target_sr_name = target_sr.GetAttrValue(attr)
                    dist = StringUtils.getLevenshteinDistance(normName(source_sr_name), normName(target_sr_name))
                    t.add_row([crs_code, target_sr_name, dist])
            else:
                target_sr_name = target_sr.GetAttrValue(attr)
                dist = StringUtils.getLevenshteinDistance(normName(source_sr_name), normName(target_sr_name))
                t.add_row([crs_code, target_sr_name, dist])
        except:
            pass

    if len(t._rows) > 0:
        print "Candidate SRSs ordered by Levenshtein Distance (a shorter distance means a better match):"
        print t.get_string(sortby = "Levenshtein Distance")
    else:
        if bProj4 == True:
            print "No PROJ4 string matches..."
            print("Searching for candidate SRs in the EPSG Registry comparing SR names...")
            searchCandidates(source_sr, bProj4 = False)
        else:
            print "Sorry, there aren't candidate SRSs!"

    # Close the connection
    conn.close()

def normName(name):
    return StringUtils.replace(StringUtils.lowerCase(name), "_", " ")

def main():

    # EPSG:4326 (ESRI WKT)
    #wkt =
'"GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.01
"Degree",0.0174532925199433295]]"'
    # EPSG:23032 (gdalsrsinfo EPSG:23032 -o wkt_esri --> ESRI WKT)
    wkt =
'"PROJCS["ED50_UTM_zone_32N",GEOGCS["GCS_European_1950",DATUM["D_European_1950",SPHEROID["International_1924",6378388,297]],PRIM
8,297]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PAR
rigin",0],PARAMETER["central_meridian",9],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],U
northing",0],UNIT["Meter",1]]"'
```

```
    # EPSG:23032 (ESRI WKT)
    #wkt =
'"PROJCS["ED_1950_UTM_Zone_32N",GEOGCS["GCS_European_1950",DATUM["D_European_1950",SPHEROID["International_1924",6378388.0,297.0
8388.0,297.0]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500
_Easting",500000.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",9.0],PARAMETER["Scale_Factor",0.9996],PARAMETER["Latitu
ETER["Latitude_Of_Origin",0.0],UNIT["Meter",1.0]]"'
    # EPSG:23032 (OGC WKT)
    #wkt = '"PROJCS["ED50 / UTM zone 32N", GEOGCS["ED50", DATUM["European Datum 1950", SPHEROID["International 1924",
6378388.0, 297.0]], PRIMEM["Greenwich", 0.0], UNIT["degree", 0.0174532925199433]], PROJECTION["Transverse_Mercator"],
PARAMETER["latitude_of_origin", 0.0], PARAMETER["central_meridian", 9.0], PARAMETER["scale_factor", 0.9996],
PARAMETER["false_easting", 500000.0], PARAMETER["false_northing", 0.0], UNIT["metre", 1.0], AUTHORITY["EPSG", 23032]]"'
    # EPSG:3004 (ESRI WKT)
    wkt = '"PROJCS["Monte Mario / Italy zone 2",GEOGCS["Monte
Mario",DATUM["D_Monte_Mario",SPHEROID["International_1924",6378388,297]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433]],PROJ
3295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",15],PARAMETER["scale_factor",0.999
factor",0.9996],PARAMETER["false_easting",2520000],PARAMETER["false_northing",0],UNIT["Meter",1]]"'
    # EPSG:3857 (gdalsrsinfo EPSG:3857 -o wkt --> OGC WKT)
    #wkt = '"PROJCS["WGS 84 / Pseudo-Mercator",GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["de
1"]],UNIT["degree",0.0174532925199433,AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]],PROJECTION["Mercator_1SP"],PARAMETER["cent
METER["central_meridian",0],PARAMETER["scale_factor",1],PARAMETER["false_easting",0],PARAMETER["false_northing",0],UNIT["metre",1,AUTHORIT
1,AUTHORITY["EPSG","9001"]],AXIS["X",EAST],AXIS["Y",NORTH],EXTENSION["PROJ4","+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0
+lon_0=0.0 +x_0=0.0 +y_0=0 +k=1.0 +units=m +nadgrids=@null +wktext +no_defs"],AUTHORITY["EPSG","3857"]]"'
    # EPSG:3857 (OGC WKT)
    #wkt = '"PROJCS["WGS 84 / Pseudo-Mercator", GEOGCS["WGS 84", DATUM["World Geodetic System 1984", SPHEROID["WGS 84",
6378137.0, 298.257223563, AUTHORITY["EPSG","7030"]], AUTHORITY["EPSG","6326"]], PRIMEM["Greenwich", 0.0,
AUTHORITY["EPSG","8901"]], UNIT["degree", 0.0174532925199433], AXIS["Geodetic longitude", EAST], AXIS["Geodetic latitude",
NORTH], AUTHORITY["EPSG","4326"]], PROJECTION["Popular Visualisation Pseudo Mercator", AUTHORITY["EPSG","1024"]],
PARAMETER["semi_minor", 6378137.0], PARAMETER["latitude_of_origin", 0.0], PARAMETER["central_meridian", 0.0],
PARAMETER["scale_factor", 1.0], PARAMETER["false_easting", 0.0], PARAMETER["false_northing", 0.0], UNIT["m", 1.0], AXIS["Easting",
EAST], AXIS["Northing", NORTH], AUTHORITY["EPSG","3857"]]"'
    # EPSG:3857 (ESRI WKT see #3954)
    #wkt = '"PROJCS["WGS_84_Pseudo_Mercator",GEOGCS["GCS_WGS_1984",DATUM["D_World Geodetic System
1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Popular
ON["Popular Visualisation Pseudo
Mercator"],PARAMETER["semi_minor",6378137.0],PARAMETER["latitude_of_origin",0.0],PARAMETER["central_meridian",0.0],PARAMETER["scale_factor
["scale_factor",1.0],PARAMETER["false_easting",0.0],PARAMETER["false_northing",0.0],UNIT["m",1.0]]"'
    # EPSG:3857 (gdalsrsinfo EPSG:3857 -o wkt_esri --> ESRI WKT)
    #wkt =
'"PROJCS["WGS_1984_Web_Mercator_Auxiliary_Sphere",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,29
6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Mercator_Auxiliary_Sphere"],PARAMETER[
ARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",0.0],PARAMETER["Standard_Parallel_1",0.0],PAR
l_1",0.0],PARAMETER["Auxiliary_Sphere_Type",0.0],UNIT["Meter",1.0]]"'
    # EPSG:3857 (gdalsrsinfo EPSG:3857 -o wkt_noct --> OGC WKT no CT)
    #wkt = '"PROJCS["WGS 84 / Pseudo-Mercator",GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
84",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["degree",0.0174532925199433]],PROJECTION["Mercator_1SP"],PARAMETER["central_meri
["central_meridian",0],PARAMETER["scale_factor",1],PARAMETER["false_easting",0],PARAMETER["false_northing",0],UNIT["metre",1]]"'
    # Corrupt WKT
    #wkt = '"foo"'

    print wkt

    source_sr = osr.SpatialReference()
```

```
    try:
        source_sr.ImportFromWkt(wkt)
        #source_sr.MorphToESRI() # allow to skip the +towgs84 parameters
        source_sr.MorphFromESRI() # it could be an ESRI WKT in the worst case
    except: #RuntimeException:
        print "Check the WKT, it may be corrupt!"
        return

    # 1. Autoidentify EPSG
    try:
        source_sr.AutoIdentifyEPSG()
        crs_code = source_sr.GetAuthorityCode(None)
        print("OGR can autoidentify the EPSG code!")
        print ">>> EPSG:%s <<<" % crs_code
        return
    except:
        print("OGR can't autoidentify the EPSG code...")

        # 2. Searching for candidate SRSs comparing proj4 strings and SRS names
        try:
            source_proj4 = source_sr.ExportToProj4()
            print("OGR can export the source SR to PROJ4...")
            #print source_proj4
            print("Searching for candidate SRs in the EPSG Registry comparing PROJ4 strings and SR names...")
            searchCandidates(source_sr, bProj4 = True)

        # 3. Searching for candidate SRSs comparing SRS names only
        except:
            print("OGR can't export the source SR to PROJ4...")
            print("Searching for candidate SRs in the EPSG Registry comparing SR names...")
            searchCandidates(source_sr, bProj4 = False)
```

#2 - 07/14/2016 06:00 PM - Antonio Falciano

For instance, this is the result of the above script related to one of the **EPSG:23032 WKT** examples:

```
Running script wkt2epsg.
PROJCS["ED_1950_UTM_Zone_32N",GEOGCS["GCS_European_1950",DATUM["D_European_1950",SPHEROID["International_1924",6378388.0,297.0]
388.0,297.0]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["Transverse_Mercator"],PARAMETER["False_Easting",500
Easting",500000.0],PARAMETER["False_Northing",0.0],PARAMETER["Central_Meridian",9.0],PARAMETER["Scale_Factor",0.9996],PARAMETER["Latitud
TER["Latitude_Of_Origin",0.0],UNIT["Meter",1.0]]
OGR can't autoidentify the EPSG code...
OGR can export the source SR to PROJ4...
Searching for candidate SRs in the EPSG Registry comparing PROJ4 strings and SR names...
Candidate SRSs ordered by Levenshtein Distance (a shorter distance means a better match):
+-----------+----------------------+----------------------+
| EPSG code | SRS name             | Levenshtein Distance |
+-----------+----------------------+----------------------+
|     23032 | ED50_UTM_zone_32N    |          3           |
|      2077 | ELD79_UTM_zone_32N   |          5           |
```

```
|     3199 | LGD2006_UTM_zone_32N |         7          |
+----------+----------------------+--------------------+
Script wkt2epsg terminated.
```

...for **EPSG:3004 WKT** example:

```
Running script wkt2epsg.
PROJCS["Monte Mario / Italy zone 2",GEOGCS["Monte
Mario",DATUM["D_Monte_Mario",SPHEROID["International_1924",6378388,297]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]],PROJ
3295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",15],PARAMETER["scale_factor",0.999
factor",0.9996],PARAMETER["false_easting",2520000],PARAMETER["false_northing",0],UNIT["Meter",1]]
OGR can't autoidentify the EPSG code...
OGR can export the source SR to PROJ4...
Searching for candidate SRs in the EPSG Registry comparing PROJ4 strings and SR names...
Candidate SRSs ordered by Levenshtein Distance (a shorter distance means a better match):
+----------+--------------------------+----------------------+
| EPSG code | SRS name                 | Levenshtein Distance |
+----------+--------------------------+----------------------+
|     3004 | Monte_Mario_Italy_zone_2 |         2          |
+----------+--------------------------+----------------------+
Script wkt2epsg terminated.
```

...and finally for the **EPSG:3857 WKT** example:

```
Running script wkt2epsg.
PROJCS["WGS_84_Pseudo_Mercator",GEOGCS["GCS_WGS_1984",DATUM["D_World Geodetic System
1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433295]],PROJECTION["Popular
ON["Popular Visualisation Pseudo
Mercator"],PARAMETER["semi_minor",6378137.0],PARAMETER["latitude_of_origin",0.0],PARAMETER["central_meridian",0.0],PARAMETER["scale_factor
["scale_factor",1.0],PARAMETER["false_easting",0.0],PARAMETER["false_northing",0.0],UNIT["m",1.0]]
OGR can't autoidentify the EPSG code...
OGR can't export the source SR to PROJ4...
Searching for candidate SRs in the EPSG Registry comparing SR names...
Candidate SRSs ordered by Levenshtein Distance (a shorter distance means a better match):
+----------+-------------------------------------------------------------------------+----------------------+
| EPSG code | SRS name                                                                | Levenshtein Distance |
+----------+-------------------------------------------------------------------------+----------------------+
|     3857 | WGS 84 / Pseudo-Mercator                                                |         3          |
|     3832 | WGS 84 / PDC Mercator                                                   |         5          |
|     3395 | WGS 84 / World Mercator                                                 |         7          |
|     3994 | WGS 84 / Mercator 41                                                    |         9          |

|     7613 | NAD83(2011) / WISCRS Kenosha, Milwaukee, Ozaukee and Racine (ftUS)      |         56         |
|     7535 | NAD83(2011) / WISCRS Calumet, Fond du Lac, Outagamie and Winnebago (m)  |         60         |
|     7594 | NAD83(2011) / WISCRS Calumet, Fond du Lac, Outagamie and Winnebago (ftUS) |         63         |
+----------+-------------------------------------------------------------------------+----------------------+
```

#3 - 07/19/2016 11:55 AM - Antonio Falciano

- File provincias_andalucia.zip added

I attach a shapefile (provincias_andalucia.shp) whose projection file (**EPSG:23030**) is not autoidentified. For instance, try to load it in a view defined in EPSG:4326. Its prj file (ESRI WKT, so no main AUTHORITY tag) was generated by GDAL 2.1.0.
Instead, this is the output of the wkt2epsg.py script:

```
Running script wkt2epsg.
PROJCS["ED50_UTM_zone_30N",GEOGCS["GCS_European_1950",DATUM["D_European_1950",SPHEROID["International_1924",6378388,297]],PRIME
,297]],PRIMEM["Greenwich",0],UNIT["Degree",0.0174532925199433295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARA
igin",0],PARAMETER["central_meridian",-3],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],U
northing",0],UNIT["Meter",1]]
OGR can't autoidentify the EPSG code...
OGR can export the source SR to PROJ4...
Searching for candidate SRs in the EPSG Registry comparing PROJ4 strings and SR names...
Candidate SRSs ordered by Levenshtein Distance (a shorter distance means a better match):
+-----------+------------------+----------------------+
| EPSG code | SRS name         | Levenshtein Distance |
+-----------+------------------+----------------------+
|     23030 | ED50_UTM_zone_30N |          0           |
+-----------+------------------+----------------------+
Script wkt2epsg terminated.
```

#4 - 07/28/2016 01:00 PM - Antonio Falciano

- File jcrs_fix_#3954_#4178_#4288_#4290.patch added

I attach a cumulative patch that fixes #3954, #4178, #4288 and finally also #4290.
So when the prj (ESRI WKT) is not autoidentified, the best matching EPSG code is assigned.
I think that it can be further improved, but it should work.

Has patch: Yes

#5 - 07/29/2016 05:59 PM - Antonio Falciano

- File gvSIG-desktop-2.3.0-org.gvsig.projection.app.jcrs-2.1.51-snapshot-2105-testing-all-all-j1_7.gvspkg added

For testing purpose, I attach the gvspkg of org.gvsig.projection.app.jcrs too. It works fine, doing more or less what prj2epsg does. A defect is that it's not very quick, especially when it's used the first time and the EPSG Registry has not been initialized (#4025).

#6 - 08/29/2016 01:08 PM - Joaquín del Cerro Murciano

- Target version changed from 2.3.0-2447-final (rev. org.gvsig.desktop-2.0.153) to 2.4.0-2850-final (rev. org.gvsig.desktop-2.0.220)

#7 - 09/02/2016 11:38 AM - Antonio Falciano

I've just tested the patch in RC4 portable and it works like a charm... Why is the target version changed to 2.4?

#8 - 09/13/2016 06:53 PM - Joaquín del Cerro Murciano

- Target version changed from 2.4.0-2850-final (rev. org.gvsig.desktop-2.0.220) to 2.3.0-2447-final (rev. org.gvsig.desktop-2.0.153)

#9 - 09/14/2016 05:12 PM - Joaquín del Cerro Murciano

- Assignee set to Joaquín del Cerro Murciano

- Status changed from New to Fixed

#10 - 09/15/2016 09:41 AM - Joaquín del Cerro Murciano

- Target version changed from 2.3.0-2447-final (rev. org.gvsig.desktop-2.0.153) to 2.3.0-2445 (rev. org.gvsig.desktop-2.0.152)

#11 - 09/17/2016 05:58 PM - Antonio Falciano

- Status changed from Fixed to Closed

Files

gvSIG-desktop-2.3.0-wkt2epsg-1.0.0-0-testing-all-all-j1_7.gvspkg	42.9 KB	07/14/2016	Antonio Falciano
provincias_andalucia.zip	156 KB	07/19/2016	Antonio Falciano
jcrs_fix_#3954_#4178_#4288_#4290.patch	9.06 KB	07/28/2016	Antonio Falciano
gvSIG-desktop-2.3.0-org.gvsig.projection.app.jcrs-2.1.51-snapshot-21[illegible]g-all-all-j7_7.gvspkg	753 KB	07/29/2016	Antonio Falciano