```
	@Override
	public TableManager getSystemsTableManagerCustomizations(TableManager
tableManager, DatabaseProfile dbProfile) {
		super.getSystemsTableManagerCustomizations(tableManager, dbProfile);
		tableManager.submit("Systems", QueryType.update, new String[][] {
			{ "SystemCode", "N" },
			{ "Species", "|" + getSpeciesName() + "|" }
		});

		tableManager.submit("Systems", QueryType.update, new String[][] {
			{ "SystemCode", "N" },
			{ "Link",
"http://bacteria.ensembl.org/streptococcus_pneumoniae_tigr4/Gene/Summary?g=~" }
		});

		return tableManager;
	}

	  /**
	 * @see
edu.lmu.xmlpipedb.gmbuilder.databasetoolkit.profiles.UniProtSpeciesProfile#getSystemT
ableManagerCustomizations(edu.lmu.xmlpipedb.gmbuilder.databasetoolkit.tables.TableMan
ager,
	 *      edu.lmu.xmlpipedb.gmbuilder.databasetoolkit.tables.TableManager,
	 *      java.util.Date)
	 */
	@Override
	public TableManager getSystemTableManagerCustomizations(TableManager
tableManager, TableManager primarySystemTableManager, Date version) throws
SQLException, InvalidParameterException {
		// Start with the default OrderedLocusNames behavior.
		TableManager result = super.getSystemTableManagerCustomizations(tableManager,
primarySystemTableManager, version);

		// We want to grab all of the legal OrderedLocusNames Ids and
		// remove the '_', adding them to the OrderedLocusNames table
		final String vcID = "SP_*";
		String sqlQuery = "select d.entrytype_gene_hjid as hjid, c.value " + "from
genenametype c inner join genetype d " + "on (c.genetype_name_hjid = d.hjid) " +
"where (c.value similar to ?)" + "and type <> 'ordered locus names' " + "group by
d.entrytype_gene_hjid, c.value";

		String dateToday = GenMAPPBuilderUtilities.getSystemsDateString(version);
		Connection c = ConnectionManager.getRelationalDBConnection();
		PreparedStatement ps;
		ResultSet rs;
		try {
			// Query, iterate, add to table manager.
			ps = c.prepareStatement(sqlQuery);
			ps.setString(1, vcID);
			rs = ps.executeQuery();
			while (rs.next()) {
				String hjid = Long.valueOf(rs.getLong("hjid")).toString();

				// We want to remove the '_' here
```

```
                String id = rs.getString("value");

                String[] substrings = id.split("/");
                String new_id = null;
                for (int i = 0; i < substrings.length; i++) {

                    new_id = substrings[i].replace("_", "");

                    _Log.debug("Remove '_' from " + id + " to create: " + new_id + "
for surrogate " + hjid);
                    result.submit("OrderedLocusNames", QueryType.insert, new
String[][] { { "ID", new_id }, { "Species", "|" + getSpeciesName() + "|" }, {
"\"Date\"", dateToday }, { "UID", hjid } });
                }
            }
        } catch(SQLException sqlexc) {
            logSQLException(sqlexc, sqlQuery);
        }

        return result;
    }

    /**
     * Helper method for logging an SQL exception.
     */
    private void logSQLException(SQLException sqlexc, String sqlQuery) {
        _Log.error("Exception trying to execute query: " + sqlQuery);
        while (sqlexc != null) {
            _Log.error("Error code: [" + sqlexc.getErrorCode() + "]");
            _Log.error("Error message: [" + sqlexc.getMessage() + "]");
            _Log.error("Error SQL State: [" + sqlexc.getSQLState() + "]");
            sqlexc = sqlexc.getNextException();
        }
    }

    private static final Log _Log =
LogFactory.getLog(StreptococcusPneumoniaeTIGR4UniProtSpeciesProfile.class);
}
```